POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Versatile Strategy for Comprehensive Data Collection and Retention in Embedded SoC Memories

Original
A Versatile Strategy for Comprehensive Data Collection and Retention in Embedded SoC Memories / Bernardi, P., Insinga, G., Battilana, M., Beer, P., Carnevale, G., Coppetta, M., Mautone, N., Repele, A., Scaramuzza, P., Ullmann, R.. - In: ACM TRANSACTIONS ON EMBEDDED COMPUTING SYSTEMS. - ISSN 1539-9087. - 25:1(2026), pp. 1-15. [10.1145/3766550]

Availability:
This version is available at: 11583/3006890 since: 2026-01-23T12:24:34Z

Publisher:
ACM

Published
DOI:10.1145/3766550

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

20 November 2026

Check for updates

ACM DIGITAL LIBRARY | Association for Computing Machinery | acm open

Latest updates: https://dl.acm.org/doi/10.1145/3766550

RESEARCH-ARTICLE

A Versatile Strategy for Comprehensive Data Collection and Retention in Embedded SoC Memories

PAOLO BERNARDI, Polytechnic of Turin, Turin, TO, Italy

GIORGIO INSINGA, Polytechnic of Turin, Turin, TO, Italy

MATTEO BATTILANA, Infineon Technologies AG, Neubiberg, Germany

PETER H BEER, Infineon Technologies AG, Neubiberg, Germany

GIAMBATTISTA CARNEVALE, Infineon Technologies AG, Neubiberg, Germany

MATTEO COPPETTA, Infineon Technologies AG, Neubiberg, Germany

View all

Open Access Support provided by:

Infineon Technologies AG

Polytechnic of Turin

PDF Download
3766550.pdf
23 January 2026
Total Citations: 0
Total Downloads: 81

Published: 06 January 2026
Online AM: 10 October 2025
Accepted: 22 August 2025
Revised: 14 July 2025
Received: 13 December 2024

Citation in BibTeX format

A Versatile Strategy for Comprehensive Data Collection and Retention in Embedded SoC Memories

PAOLO BERNARDI, Polytechnic of Turin, Torino, Italy
GIORGIO INSINGA, Polytechnic of Turin, Torino, Italy
MATTEO BATTILANA, Infineon Technologies, Padova, Italy
PETER BEER, Infineon Technologies AG, Neubiberg, Germany
GIAMBATTISTA CARNEVALE, Infineon Technologies, Padova, Italy
MATTEO COPPETTA, Infineon Technologies, Padova, Italy
NELLINA MAUTONE, Infineon Technologies AG, Neubiberg, Germany
ALBERTO REPELE, Infineon Technologies, Padova, Italy
PIERRE SCARAMUZZA, Infineon Technologies, Padova, Italy
RUDOLF ULLMANN, Infineon Technologies AG, Neubiberg, Germany

In modern automotive system-on-chip (SoC) designs, large embedded flash memories have become a standard feature. Since they occupy a significant percentage of the die area, their impact on the SoCs' overall yield is substantial, making them a critical component in the production process. Embedded memories are then deeply tested to unsure their reliability. The data collected through these tests are fundamental to chip designers and test engineers to iron out their designs and understand the most common failure mechanisms. A common approach for data collection is the generation of bitmaps based on the gathering of individual fail coordinates in a list-based fashion. Other more efficient compaction or compression approaches exist and all these approaches can use dedicated internal memories to store the result of a given test. Unfortunately, all the methods currently found in the literature do not allow diagnostic data retention along multiple tests, requiring constant and time-consuming communications with the external tester, increasing the test cost for the manufacturers.

This article presents an on-chip algorithm to compact and retain diagnostic information from multi-step embedded memories testing. The foundation of this work lies in an efficient shape recognition and encoding algorithm. The collected information is stored in a dedicated nonvolatile on-chip memory. Information about the tests that generated a given set of fault shapes is also encoded in this dedicated diagnostic memory, enabling manufacturers to collect all the diagnostic information at the end of their test flow. Experimental results on over 110 Automotive SoCs made by Infineon™ show that using the proposed approach, 100% of the diagnostic information of devices undergoing a standard automotive-grade test flow is permanently encodable

Authors' Contact Information: Paolo Bernardi, Polytechnic of Turin, Torino, Piemonte, Italy; e-mail: paolo.bernardi@polito.it; Giorgio Insinga (corresponding author), Polytechnic of Turin, Torino, Italy; e-mail: giorgio.insinga@polito.it; Matteo Battilana, Infineon Technologies, Padova, Padova, Italy; e-mail: Matteo.Battilana-EE@infineon.com; Peter Beer, Infineon Technologies AG, Neubiberg, BY, Germany; e-mail: Peter.Beer@infineon.com; Giambattista Carnevale, Infineon Technologies, Padova, Padova, Italy; e-mail: Giambattista.Carnevale@infineon.com; Matteo Coppetta, Infineon Technologies, Padova, Padova, Italy; e-mail: Matteo.Coppetta@infineon.com; Nellina Mautone, Infineon Technologies AG, Neubiberg, BY, Germany; e-mail: Nellina.Mautone@infineon.com; Alberto Repele, Infineon Technologies, Padova, Padova, Italy; e-mail: Alberto.Repele@infineon.com; Pierre Scaramuzza, Infineon Technologies, Padova, Padova, Italy; e-mail: Pierre.Scaramuzza@infineon.com; Rudolf Ullmann, Infineon Technologies AG, Neubiberg, BY, Germany; e-mail: rudolf.ullmann@infineon.com.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).
ACM 1539-9087/2026/01-ART6
https://doi.org/10.1145/3766550

in a limited 24 KB diagnostic space while also consistently reducing the total test time of up to 53.8% with respect to traditional list-based approaches.

CCS Concepts: • **Hardware → Memory test and repair**; **Test data compression;**

Additional Key Words and Phrases: Automotive SoC, reliability, memory diagnosis, compaction, data encoding

ACM Reference Format:
Paolo Bernardi, Giorgio Insinga, Matteo Battilana, Peter Beer, Giambattista Carnevale, Matteo Coppetta, Nellina Mautone, Alberto Repele, Pierre Scaramuzza, and Rudolf Ullmann. 2026. A Versatile Strategy for Comprehensive Data Collection and Retention in Embedded SoC Memories. *ACM Trans. Embedd. Comput. Syst.* 25, 1, Article 6 (January 2026), 15 pages. https://doi.org/10.1145/3766550

1 Introduction

The amount of memory embedded in **System-on-Chips** (**SoCs**) is constantly increasing. The reasons for this continuous expansion must be investigated in the increased complexity of SoCs and the advancements in the technology nodes used. The more complex the device, the more developers are able to create more powerful applications, requiring more and more memory for the data and the program part.

While single-cell dimensions are scaling down alongside the adopted technology node, the memory area is not scaling down proportionally. SoCs need more and more program and data memory, and, consequently, the embedded memories keep a significant amount of the chip surface.

Due to their significant dimensions, embedded memories strongly impact the yield of the devices into which they are integrated. Consequently, manufacturers make a strong effort to assess the reliability and the characteristics of the embedded memories of their SoCs [1–3]. The testing efforts are particularly increased in the case of safety-critical systems such as the aerospace and automotive SoCs, where malfunction can endanger human lives. For these reasons, embedded memories in automotive SoCs follow complex test flows designed to stress them and discover as many weak cells and near-breaking point components as possible [4, 5]. These long test flows analyze the memories in a heterogeneous set of different conditions, such as:

— Temperature ranges: operating temperatures for embedded memories usually go from -40℃ to 125℃. Memories need to be tested in order to ensure their reliability within this temperature range.
— Sense amplifier reference current: changing the reference current for the memory sense amplifiers and verifying its correct behavior creates a margin test. This test is useful to understand structural properties of the memory since faults tend to concentrate in specific areas [6, 7].
— Voltage: every SoC has a range of acceptable supply voltages. Suppliers test their SoCs at multiple points to validate their correct behavior in the specified voltage range.

While there are several kind of memories that can be embedded in a SoC, each one of them is organized in regular structures that can be represented as a matrix with X rows and Y columns. As a consequence, algorithms that compact or compress faults in the memory based on geometrical considerations (such as [7] and [8]) are immediately applicable to any kind of embedded memory (similarly to what is described for MBISTs design in [9]). So while this article focuses specifically on flash memory implementations, the proposed algorithms are applicable to other embedded memory types due to their shared matrix organization.

Devices Under Test (DUTs) are therefore, deeply tested in a variety of different conditions [6], generating huge quantities of diagnostic data, useful for production purposes, initial

characterization, field return analysis, and so on. Several methodologies to store and export these data to the external world exist in literature. The simplest and fastest one is the one in [10], which reports each single failing cell one by one in a list fashion. More efficient solutions can range from the lossless and efficient shape encoding representation [8] to the incredibly space-saving but imprecise compression methods in [7], [11], and [12]. In particular, [8] and [7] are very efficient regarding memory requirements but always need slow communication with the test machine at the end of each test to transmit the diagnostic information before the following test.

In this article, we will present an optimized method to store and preserve on-chip the diagnostic information coming from the overall test flow of the DUT. The resulting cumulative data can then be exported at the end of the flow, saving an incredible amount of test time previously used for continuous communication with the external tester. The time saved makes this approach suitable for production environments where reducing time and costs is paramount. Additionally, using efficient techniques such as [8] and [7], all the diagnostic information can be stored inside the device's limited dedicated diagnostic memory. Besides reducing communication time with the external tester, permanently storing the diagnostic history of the DUT's embedded memories is incredibly valuable in case of field returns as it simplifies their diagnosis process.

The article is organized as follows: Section II briefly explains the embedded memory organization and analyzes the eMemory testing flow to understand the primary source of our diagnostic information. In section III, we explain in detail our approach and the process that goes from the fails' coordinates discovery to their encoding and to their permanent on-chip storage. Section IV shows our experimental results on over 110 devices undergoing a real Automotive-grade test flow. In Section V, we make some final considerations.

2 Background

This section presents the fundamental concepts of eMemories, their structure, and their tests to understand where the diagnostic data comes from and why they are so important for automotive SoC manufacturers. Moreover, this section introduces the optimized diagnostic data encoding algorithms that are integrated in the proposed approach.

2.1 Embedded Memory Organization

In eMemories, the bits composing the memory matrix are organized into rows called wordlines and columns called bitlines. Each wordline is further divided into pages of a certain number of bits. Pages represent the minimum granularity for the memory and are made by a certain amount of bits. The entire page shall be accessed and, eventually, modified to read or program a single bit. Single memory units of a given amount of wordlines and bitlines are called physical sectors. Lastly, the higher level consists of banks that group multiple physical sectors. It is also important to mention a common memory organization called scrambling, composed of multiplexed and mirrored bits as detailed in [13]:

— Multiplexing: bits with the same index are physically adjacent in the wordline
— Mirroring: the wordline is mirrored using its middle point as a symmetry axis.

Figure 1 shows a visual representation of a 16-bit memory organized in 4-bit words. Physically, it implements a mux factor of 4 and a mirror every 2 scrambled bits.

Depending on the type of the memory itself (FLASH, RAM, and RRAM) there may be slight difference or peculiarities about the internal organization. What is important to note is that they all follow a regular structure, that is, still composed of a matrix of bits organized in X rows and Y columns. So any reasoning that works on the matrix representation of a Flash-based memory can easily be transposed to other kind of memories.

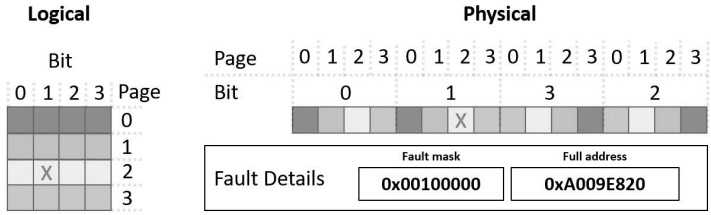


Fig. 1. Memory organization and fault details as received during the tests.

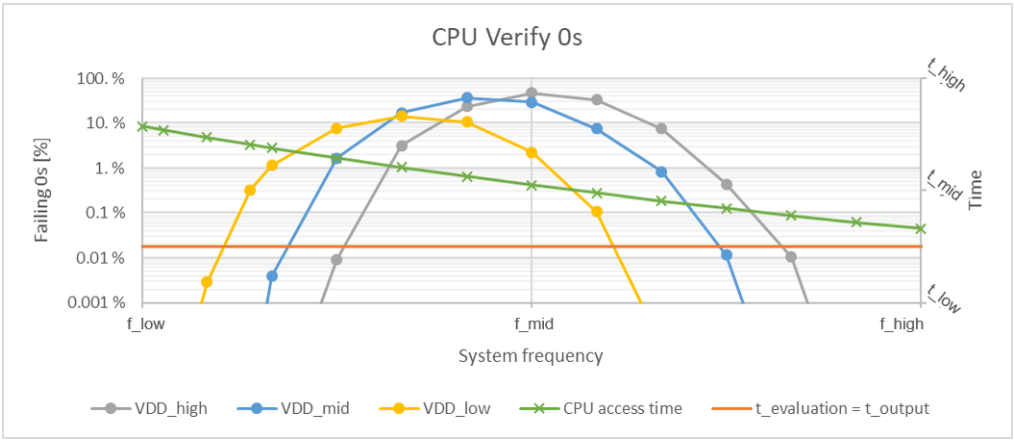


Fig. 2. Distribution of faults based on memory supply voltage and read access time.

2.2 Test Flow

Since requirements for automotive fields are particularly strict, eMemories have to be reliable across various environmental and electrical conditions. The reliability of the embedded memories has to be assessed in conditions such as:

- Temperature: standard operating temperatures for embedded memories range from -40℃ to 125℃. During characterization and mass production, DUTs are actually tested in temperatures slightly above and slightly below the specified ones to maintain a certain safety margin.
- Voltage: also for supply Voltage there is a specified range of acceptable values. To assess the correct behavior of the devices along the entire range, manufacturers test values slightly below and slightly above the specified values.
- Timing: Memory readings or writings follow a set of precisely timed operations involving multiple actors such as the sense amplifiers, the row and column selectors, and, generally, signals propagation. The correct timing behavior has to be deeply tested during characterization since is one of the key aspect of every memory read or write operations. Figure 2 (from [6]) illustrates how supply voltage and read access time affect the fault distribution in an Infineon™ Automotive SoC embedded memory.
- Sense Amplifier reference: a circuit named **Sense Amplifier** (**SA**) is used to decide whether the content of a bit cell is a 0 or a 1. The SA, represented in Figure 3, compares the current of the read memory cell against a reference current. If the current of the cell is higher than the reference, the bit cells is read as 0, 1 otherwise. The manufacturer sets the reference current depending on the technology, and its correct calibration is fundamental to obtaining a valid read from the memory.

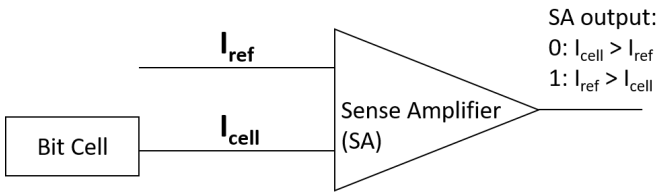


Fig. 3. The SA decides if the content of a bit cell is a 0 or a 1.

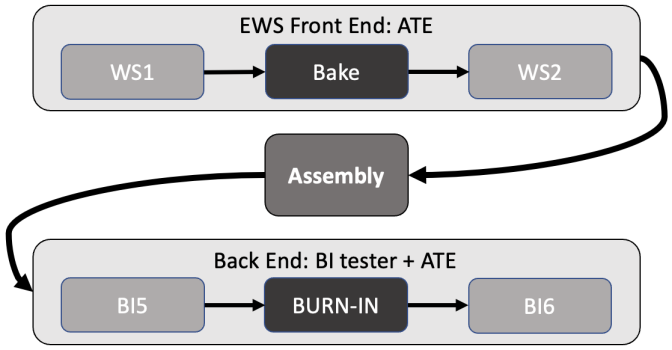


Fig. 4. eFlash test flow from wafer level to package level.

To test all these aspects, a comprehensive test flow is developed. The test flow is a collection of multiple test steps performed in different configurations. Apart from testing the DUT's behavior at different environmental conditions, these test flows also search for faults such as the classic stuck-at and slow-to-raise faults.

For these reasons, automotive SoC manufacturers have to test the embedded memories of their devices deeply. Multiple sensitization and detection steps are performed in different conditions to test the memories fully. The sensitization consists of:

— Active: program, erase, disturb, and stress operations.
— Passive: Retention bake and accelerated read disturb.

As described in Figure 4, these sensitization-detection steps are performed both in the Front End, at the wafer level, and in the **Back End** (**BE**) after packaging.

The main goal of these tests is to discard defective devices as soon as possible, to optimize the process costs.

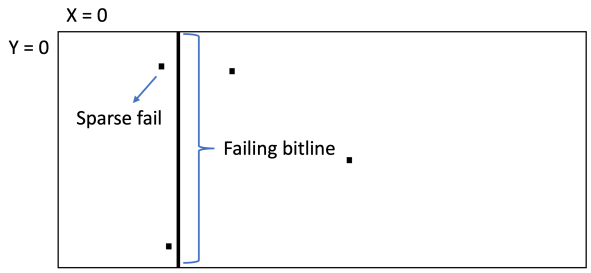
The first step is the **Wafer Sort** (**WS**), in which the manufacturers skim the first defective units. This is followed by the data retention Bake and by another WS to help assess the data storage capacity of the memories under test.

Devices that survived the previous steps are then assembled (packaged) and tested again in the BE after a Burn-in that stresses the memories with a series of memory operations. Figure 4 graphically explains this test flow.

2.3 Diagnostic Data Encoding Methods

The test flows in the preceding paragraph produce a large volume of diagnostic data. This data must be managed effectively by the manufacturers who seek to both cut test time and boost their yield (and, consequently, reduce test cost).

To handle this data there are multiple methods found in the literature that span from list-based methods [10] to more advanced shape recognition methods [8] to lossy compression methods [7].

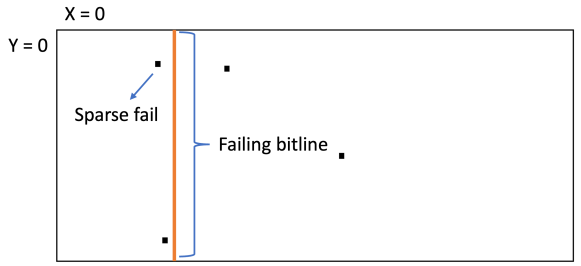


Fault Location	
X	Y
4	5
6	0
6	1
6	2
...	...

Sparse fault

Faults of the failing bitline

Fig. 5. List-based method example.



Encoded Slice			
Start (X,Y)	Stop (X,Y)	Color	
4,5	4,5	Black	Sparse fault
6,0	6,1024	Orange	Faults of the failing bitline
9, 6	9,6	Black	Sparse fault
18, 50	18,50	Black	Sparse fault
5,900	5,900	Black	Sparse fault

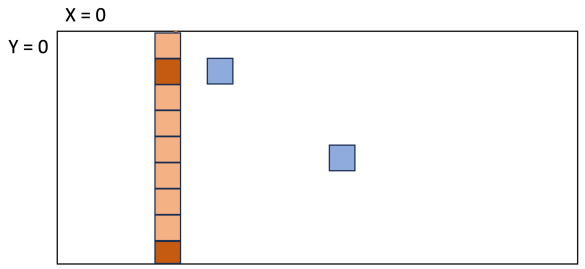
Fig. 6. Compacted representation of failing bits.

2.3.1 List-based Methods. List-based method such as the one presented in [10] are the easiest ones both conceptually and from an algorithmic point of view. Each fault found is represented in a list together with its coordinates. The advantage of this method is in the immediate processing of the data, which just requires storing the fault location without any processing. The disadvantage lies in the memory requirements to store all the fault location one by one. Faults that involve an entire bitline or wordline or even a physical sector can take a huge amount of space to be represented. Figure 5 shows an example of a failing bitline and some sparse faults represented with Landzberg's method.

2.3.2 Shape Recognition Methods. Shape-recognition methods such as the one presented in [8] are more efficient than List-Based methods in terms of required diagnostic memory. At the cost of a short time overhead, these lossless methods achieve significant compaction of the diagnostic data by recognizing and encoding common fault shapes with efficient codes. Figure 6 graphically shows how this approach differs from the list-based ones, with faults encoded in colored segments called "slices". Each slice has a start address, a stop address, and a color that encodes the shapes of the failures inside it (for example, a red slice contains errors just in the odd or even positions, in a checkerboard style).

2.3.3 Compression Methods. Compression methods are lossy approaches that represent general features of memory faults, without storing the exact location of each single element. The method described in [7] is an example of compression algorithm that divides the memory in squares called "Pixels". Each pixel stores the amount of faults, giving information about the failure density of a certain memory area. Figure 7 shows how the same example of Figures 6 and 5 is encoded using the pixel compression.

2.3.4 SRAM Specific Compression and Compaction Approaches. Other diagnostic data compression and compaction techniques are already present in the literature for specific memory types



Encoded Pixel		
X	Y	Faults inside
2	0	6
2	1	7
2	2	6
2	3	6
...	...	

Faults of the failing bitline + sparse

Fig. 7. Compressed representation of failing bits.

such as SRAM. In particular [14] shows two compaction methods that works for embedded SRAM memories, reducing diagnostic data requirements by 28%. The methods anyway require dedicated hardware and is specialized for SRAM-based memories.

3 Proposed Approach

The proposed approach is a methodology to efficiently encode diagnostic data along the entire test flow of embedded memories in automotive SoCs. Using the proposed approach, it is possible to memorize on-chip the entire diagnostic history of the device. This diagnostic information can then be downloaded altogether at the end of the device test flow, contrary to the previous methods in which the download is performed at the end of each test. As communication with the tester machine is a slow process, having a single dump at the end of the test flow is a considerable added value for manufacturers, especially in a productive environment where reducing test time (and, consequently, costs) is paramount.

The strategy presented in this article is a further encoding layer for data analyzed through the methods proposed in [8] and [7]. The entire test flow data encoded in this efficient way can be stored in a small integrated and dedicated diagnostic memory and collected at the end of the SoCs test flow as graphically explained in Figure 8. All information about faults and most importantly in which test step each individual fault was found. Similarly to [8] and [7], just the first appearance of a fault is stored for each given test, as the purpose of the proposed algorithm is to extract a information about the bit cell (or memory area) status at a given combination of temperature, speed, content and so on. In case of a march-like test, each read is considered a new test, so there is no information loss.

The base elements of this new encoding algorithm are called basic and control slices. Basic slices contains the same information as the slices reported in [8] or the pixel slices [7] depending on the chosen fault collection algorithm. Control slices on the other hands are needed to “separate” basic slices belonging to different physical sectors and, more importantly, to different test steps.

Both these kinds of slices are saved in the dedicated diagnostic memory with basic slices always being preceded by at least a control slice.

To efficiently reach its goals the proposed algorithm reorganizes the structure in [8] and [7] and perform a post processing that creates basic and control slices.

The ability to permanently store diagnostic information inside the device represents a fundamental architectural advancement that extends beyond simple data reorganization. Similar to how network protocols like Ethernet employ distinct layers (physical, data link, network, transport, etc.) that each serve critical but independent functions, our proposed solution introduces an entirely new architectural layer to the diagnostic data management stack. This layer operates above the encoding mechanisms presented in [8] and [7], providing persistent storage capabilities while

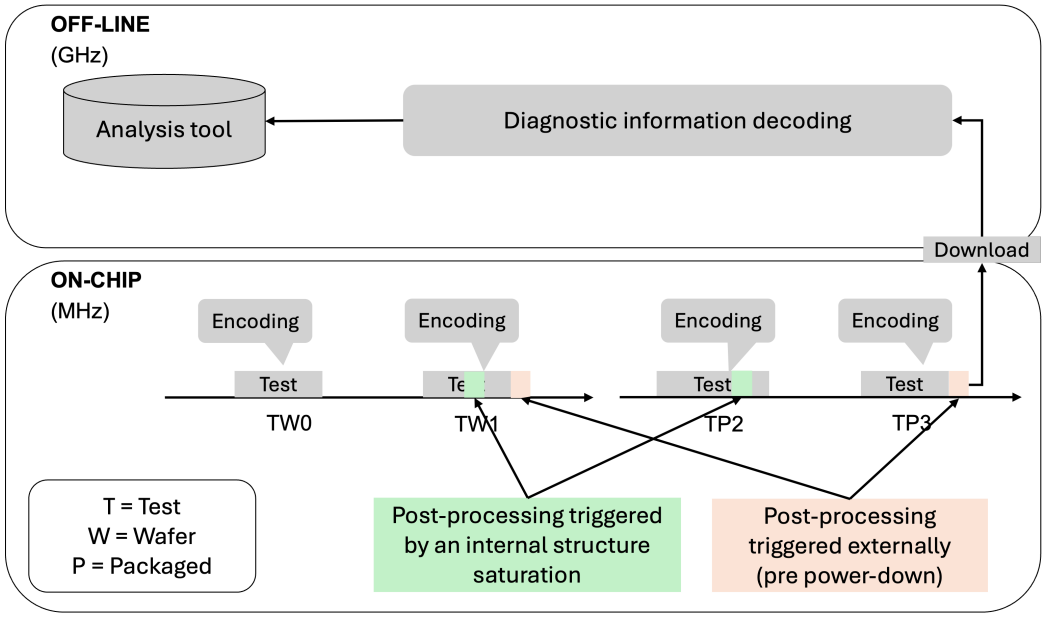


Fig. 8. The complete test flow of a device using our proposed approach.

maintaining compatibility with the underlying compression algorithms. The creation of "First of Chain" Links and Elements forms the foundation of this new layer, establishing a structured framework that enables the coexistence and management of diagnostic data from multiple test steps. Our persistence layer adds the critical capability of multi-test data retention without modifying the core encoding algorithms, thereby creating a complete, production-ready diagnostic solution.

3.1 Linked Slices Organization

Following the method proposed in [8] and [7] faults are encoded as soon as they are received in structures called "Slices".

Efficiently organizing this encoded data is crucial in the proposed algorithm and the following post-processing step.

Following the organization in [8] and [7], the diagnostic memory is split into several equal size structures called sets. Like a cache organization, these sets are made by a given number of slices. To fill these sets, the following steps are needed:

(1) Analyze the incoming fault to understand its address.
(2) Decide the correct set to store the information of the fault.
(3) Search inside the set to find a slice to store the incoming fault. Otherwise, a new slice is created.

Step number 3 is particularly slow as it requires exploring a given set to discover if a new slice is needed or if an existing slice is able to store the incoming fault. To improve the speed and prepare the slices for the proposed encoding step, the proposed approach improves the set organization by linking the slices belonging to the same bitline or wordline of a specific physical sector. Together, the linked structures form what is called a "slice chain". An example of slices composing a chain is graphically shown in Figure 9.

Figure 10 visualizes how the search for a slice is handled. The first step is to determine the correct slice set from the fault address. From the fault address a tag is extracted. With Tag, Set and the

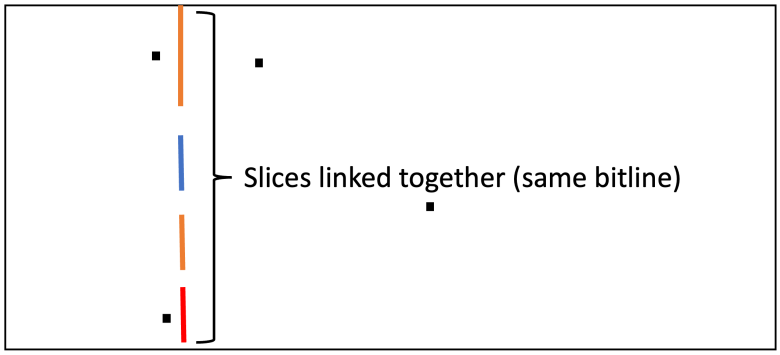


Fig. 9. Example of slices linked together.

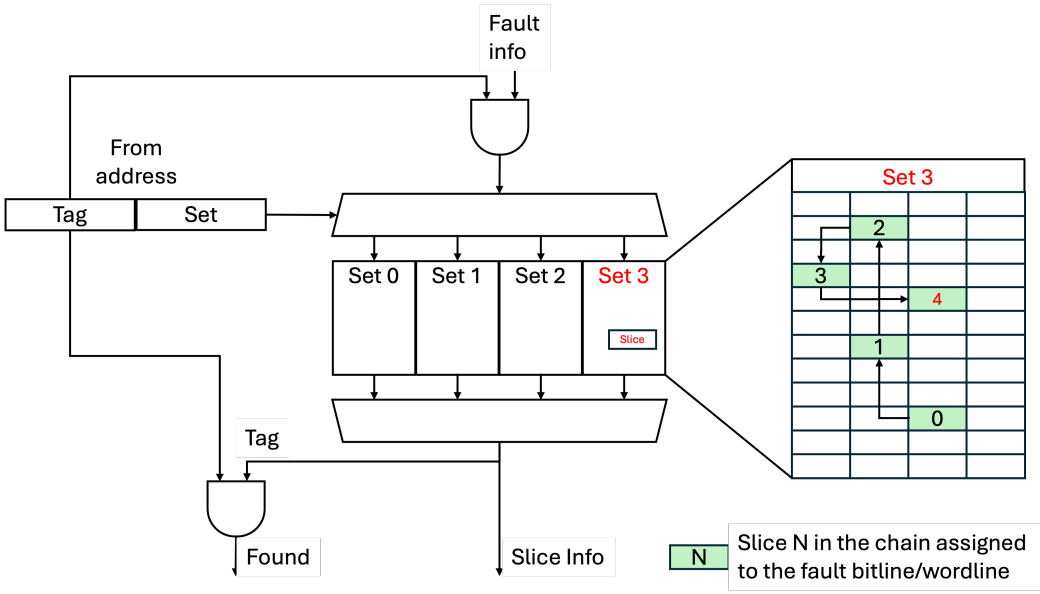


Fig. 10. Search of the correct slice inside the slice sets using tag, set and the newly introduced slice chains.

Fault info, it is possible to determine the slice chain that contain the slice of interest. In the figure, for Set 3 each rectangle represents a slice while the green ones represents the slices of a slice chain. The chain organization is incredibly flexible and allows slice insertion, removal, and move with a simple link update on the connected slices.

3.2 Multi Test Step Representation and First Chain Slice Retrieval

Additional information are needed to allow faults from different test steps to coexist in the same structures. The new slice chain organization plays a crucial role in dividing faults based on the test step that found them, as each slice chain is assigned to a specific test step. Consequently, every time a new fault is discovered, a check is performed to determine if a slice in a slice chain can encode it (i.e., the slice belongs to a slice chain with the correct bank, physical sector, test step etc.). If the current test step differs from the slice chain's, a new slice chain is created, and the first slice stores the newfound fault.

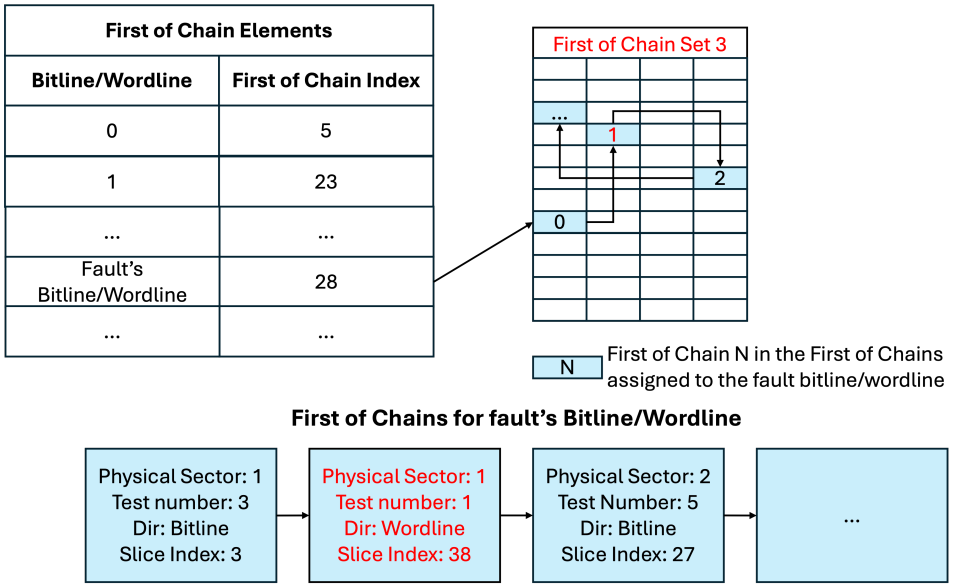


Fig. 11. The organization of the first of chain structure.

The information about the test step is not actually stored in the slice chain itself. Instead a new structure called "First Of Chain" is created for each slice set. This structure speeds up the search of a given slice chain, referring to a specific test step. To achieve this goal, the First Of Chain structure is further divided into multiple instances of "Links" and "Elements".

— First Of Chain Links (left vector in Figure 11): As the name suggests, each entry of this group is a simple link pointing to a First Of Chain Element. The number of links equals the maximum value between maxIndexBitline and maxIndexWorldline in a physical sector. Each time a new fault is discovered, its bitline (or wordline for wordline-oriented encoding) index is used as an index of the corresponding link. A chain slice exists for that wordline or bitline if the link is set.

— First Of Chain Element (Figures 11 and 12): It contains several pieces of information about a specific slice chain. In particular, it stores the chain's Physical sector, test step, direction (bitline or wordline oriented) and the location of the first slice of the chain in the set. Additionally, each element can be linked with another in case of collisions (i.e., multiple chains with the same wordline/bitline index).

3.3 Permanent Encoding

The proposed approach focuses on permanently encoding the diagnostic data in a small on-chip dedicated diagnostic memory. Our encoding compacts diagnostic information from eFlash memories along the entire DUT test flow. The proposed algorithm is a further encoding layer to the approaches presented in [8] and [7] and can store the test step that generated each fault found in the eFlash under test. Slices created following these two approaches are named "Temporary Slices" as they are just used during the initial fault encoding and will be further processed before reaching their final destination in the dedicated diagnostic memory. Being able to also encode the test step, together with meticulous data organization, allows the algorithm to store the entire diagnostic life of a device in a limited space. Using a nomenclature, that is, coherent with the one found in [8] and [7], the main structures of the proposed approach are called "Basic" and "Control" slices. These

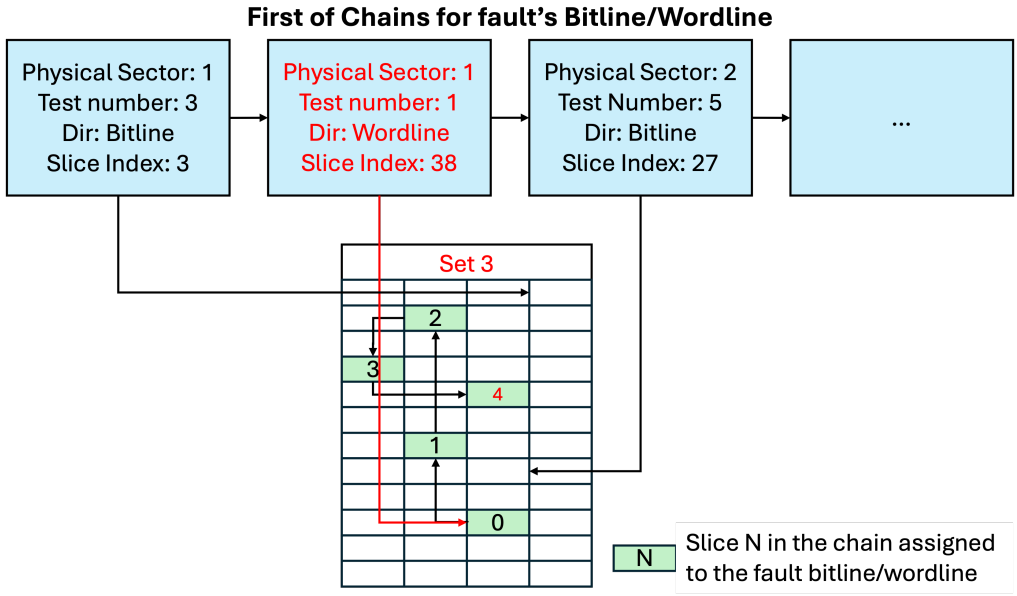


Fig. 12. The appearance of first of chains element for set 0 bitline 0.

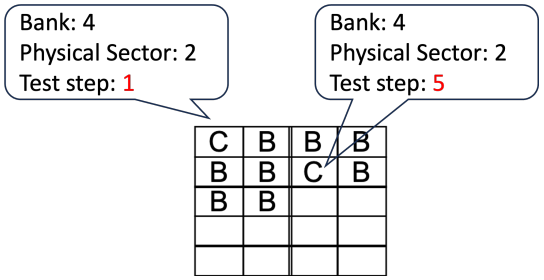


Fig. 13. Control and basic slices in the dedicated diagnostic memory.

slices are created during a process called Post Processing that processes the Temporary Slices and First of Chains contents, reorganizes them in a space-efficient manner, and places the results in the dedicated diagnostic memory. As Temporary slices and First of Chains structures are only saved in RAM, the post processing must be executed before the powering off of the device (for example, before the assembly of the device and at the end of the test flow). Additionally, the post process is called when one of the temporary structures, like a Slice Set, is saturated, causing a temporary pause of the fault encoding process. After the post-processing, all the temporary structures are initialized, and the encoding can continue.

Using the first-of-chain structure and the related temporary slices, the algorithm can store information about the test step that generated a certain failure bitmap and save a device's "testing life" in a reduced amount of space. While stored in the dedicated on-chip buffer or downloaded to the **Automated Test Equipment** (**ATE**) "control slice" are used to differentiate between failure bitmaps coming from different memory banks and physical sectors and different test steps. This concept is visually explained in Figure 13 that shows the condition of the dedicated diagnostic memory after a post processing. In the example, the first control slice refers to Bank 4, Physical

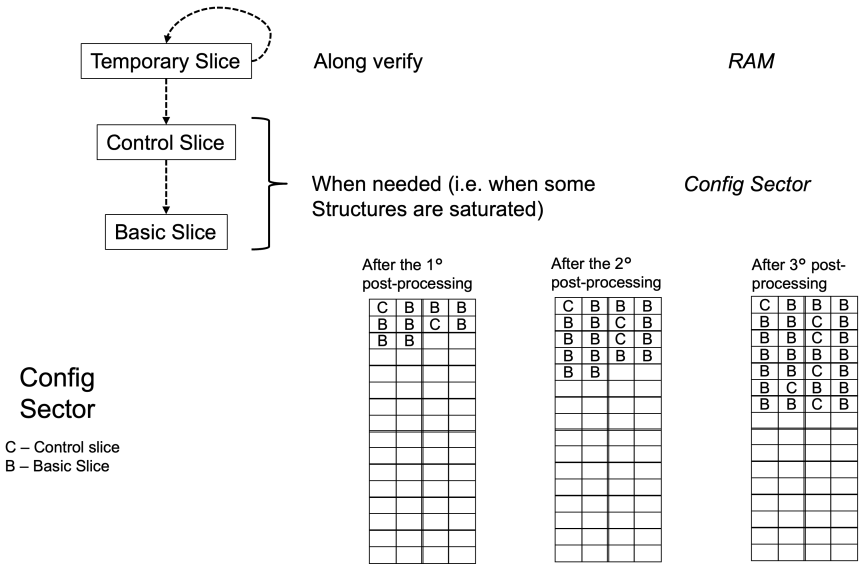


Fig. 14. Permantent storage organization after a number of post processing.

Sector 2, Test Step 1; until the following control slice is found, all the basic slices share these parameters and report faults from the specified Bank, Physical Sector, and test step.

Figure 14 shows a more complete example with the organization of the dedicated diagnostic memory after a series of three post processing.

4 Experimental Results

This section will show the experimental results collected on automotive-grade Infineon TriCore™-based SoC. Our experiments used the lossless and space-efficient encoding method presented in [8] as a foundation. By integrating our proposed methodology, we retained these benefits while expanding the encoding capabilities to encompass multiple test steps diagnostic information. Furthermore, our approach enables the permanent storage of the complete diagnostic history of the device within a compact integrated memory. Should a compressed representation be desired, our approach can alternatively encode data using the [7] method, applying analogous considerations.

It is important to note that while our experiments focus on Flash-based embedded memories in Infineon TriCore™ automotive SoCs, the proposed algorithms are broadly applicable to other embedded memory technologies. Since all embedded memories—whether Flash, RAM, RRAM, or emerging technologies—are fundamentally organized as regular matrix structures with rows and columns, the geometric fault recognition and encoding principles underlying our approach remain valid. This architectural universality ensures that our multi-test diagnostic retention methodology can be readily adapted to any embedded memory type, making it a versatile solution for comprehensive diagnostic data management across diverse memory technologies. For example, our proposed approach is also used during the tests of RRAMs at Infineon, but the company is not willing to share specific experimental details about this new technology. Nevertheless, the internal usage for RRAM supports the flexibility and reusability of the approach for multiple memory technologies.

4.1 Test Time Cost

To understand the timing benefits of our approach, we performed an experiment on 112 devices that were tested on a typical Automotive-grade eMemory test. The diagnostic data had a dedicated

Table 1. Timing Comparison and Total Proposed Approach Time Saving

Fault Threshold (#)	Devices with faults equal or more than Threshold (#)	Average Buffer dump count for [10] (#)	Average time save with Proposed Approach (%)
0	112 (all)	1.79	10.0
1	86	2.02	13.2
100	59	2.49	19.4
4,500	7	4.57	41.6
6,000	4	6	53.8
15,000	2	5.5	49.1

storage buffer of 24 KB. Every time a device saturates this buffer, the test stops, and the external ATE must download all the diagnostic information, clear the buffer, and signal the device to restart the test.

The devices actually undergo the test flow two times:

— In the first run, every diagnostic data collected was encoded following the method in [10], where each fault coming from the memory is logged individually in a list.
— The second run uses the compaction method in [8] that recognizes fault shapes.

Our results are summarized in Table 1. The table computes averages based on the number of faulty bits. In the first case, the table shows the results for devices that have 0 or more faulty bits (so all of them), showing an average time save of 10% using the proposed approach. As the number of faulty bits increase, the time saved increase up to 53.8% for devices that have more than 6,000 faults. In every case, our proposed approach needed just a single buffer dump and, except for devices with a few faults, saved a considerable amount of time with respect to the traditional Landzberg approach. As the number of faults increases, the method in Landzberg requires a high number of buffer dumps (up to seven) to be able to export all the stored diagnostic data. It is also important to note that at each test step, the tester is able to concurrently analyse multiple devices. The slower devices (the ones that have more faults and require more buffer dumps) are the bottlenecks of the test, as the ATE needs to wait for them to complete their verify before continuing to the following test step.

While the foundational approaches in [8] and [7] focus primarily on compaction and compression efficiency, they incur a test time overhead compared to traditional list-based methods like [10]. Our experimental results show that the combined approach reaches a maximum overhead of 40% in test time for the most fault-dense devices. This overhead consists of two components: approximately 35% from the shape recognition and encoding algorithms of the base methods, and an additional 5% from our proposed multi-test organization layer. However, this timing penalty is greatly compensated by the dramatic reduction in diagnostic memory requirements (up to 96% compression ratio) and, more importantly, by the cumulative time savings from eliminating intermediate data dumps. As shown in Table 1, devices with over 6,000 faults achieve a net time saving of 53.8% despite the per-test overhead, as they require only a single data dump instead of up to seven dumps with traditional approaches.

4.2 Permanent On-chip Storage

The main advantage of the proposed approach is the permanent on-chip storage of diagnostic information. This diagnostic information can be incredibly valuable for manufacturers with easier diagnoses for memory-related field returns. However, permanent storage is only possible if all the diagnostic information can fit on the small 24 KB dedicated memory inside the device itself. To

Table 2. Experimental Results on 112 Devices on two Automotive-grade Memory Test Flows

	Test flow	
	With Margins	Standard
Device Faults Representable in 24 KB (#)	56 (50%)	112 (100%)
Average Diagnostic Memory occupation (KB)	12.3	0.76
Standard Deviation (KB)	12.2	1.01

test the effectiveness of our approach in conjunction with the method in [8], we tested 112 devices using an Automotive-grade eMemory test flow composed of multiple memory verifies.

Similarly to the previous case, these devices underwent two different test flows:

— The first test flow included margin tests that change the reference current in the memories' sense amplifiers. These tests generate huge quantities of data. Complete diagnostic information is not usually needed [6, 7], and these tests are not usually run in production environments, but this approach has been used to push the limits of our approach.

— The second test flow tested the memories in search of structural faults such as stuck-at faults. Patterns such as checkerboard, inverse-checkerboard, solid 0, and solid 1 were applied. Subsequent reads were comparing the content of the memory with the expected content to find faulty memory cells. This scenario is the most realistic scenario for our approach.

The experimental results are summarized in Table 2. Looking at the Margin Test flow, 50% of the devices were successfully represented in the 24 KB available. The other 50% saturated the available memory and were thus partially represented. It is important to notice that with margin tests, the manufacturers are not interested in having the full diagnostic information, but just partial information is often enough to perform the necessary analysis [7]. In particular, the compression method in [7] is more suitable for using the proposed approach in the case of margin tests. With the standard flow, all the devices are representable in the 24 KB embedded diagnostic memory. The complete diagnostic information will then be retained throughout the device's entire useful life. In this flow, on average, just 0.76 KB of memory was actually needed to store all the diagnostic history of the devices (with 1.01 KB of standard deviation).

5 Conclusions

This article has introduced an efficient algorithm for encoding and storing comprehensive diagnostic data of embedded memories in automotive SoCs. By incorporating fault information along with the specific test step at which each fault is detected, our approach offers a complete view of the device's "testing life" within a limited memory footprint. Leveraging linked slice chains and a novel First Of Chain structure, the algorithm ensures fast fault retrieval and analysis. Using basic and control slices provides a compact and easily interpretable data representation. This approach aims at enhancing memory yield improvement activities, refining test methodologies, and ultimately improving the reliability of automotive SoCs. Experimental results from real Automotive test flow performed on over 110 Automotive-grade Infineon^{TM} Tricore SoCs show the validity of our approach. In the experiments, 100% of the diagnostic information was retained in the small 24 KB dedicated and embedded diagnostic memory. Even adding margin tests to the test flow, 50% of the devices were still completely encodable, while the other 50% retained partial but useful information.

Additionally, the proposed approach proved up to 53.8% faster, compared to previous solutions, for the fault-dense cases that typically represent the bottleneck in testing processes.

References

[1] Ad van de Goor, Georgi Gaydadjiev, and Said Hamdioui. 2010. Memory testing with a RISC microcontroller. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'10)*. European Design and Automation Association, Leuven, BEL, 214–219.

[2] Erik Jan Marinissen, Betty Prince, Doris Keitel-Schulz, and Yervant Zorian. 2005. Challenges in embedded memory design and test. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 2 (DATE'05)*. IEEE Computer Society, USA, 722–727. DOI: http://doi.org/10.1109/DATE.2005.92

[3] Tzuo-Fan Chien, Wen-Chi Chao, Chien-Mo Li, Yao-Wen Chang, Kuan-Yu Liao, Ming-Tung Chang, Min-Hsiu Tsai, and Chih-Mou Tseng. 2009. BIST design optimization for large-scale embedded memory cores. In *Proceedings of the 2009 International Conference on Computer-Aided Design (ICCAD'09)*. Association for Computing Machinery, New York, NY, USA, 197–200. DOI: http://doi.org/10.1145/1687399.1687435

[4] Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai. 2012. Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'12)*. EDA Consortium, San Jose, CA, USA, 521–526.

[5] Tei-Wei Kuo, Po-Chun Huang, Yuan-Hao Chang, Chia-Ling Ko, and Chih-Wen Hsueh. 2011. An efficient fault detection algorithm for NAND flash memory. *SIGAPP Appl. Comput. Rev.* 11, 2 (mar 2011), 8–16. DOI: http://doi.org/10.1145/1964144.1964146

[6] P. Bernardi, R. Cantoro, A. Coyette, W. Dobbeleare, M. Fieback, A. Floridia, G. Gielen, J. Gomez, M. Grosso, A. M. Guerriero, I. Guglielminetti, S. Hamdioui, G. Insinga, N. Mautone, N. Mirabella, S. Sartoni, M. Sonza Reorda, R. Ullmann, R. Vanhooren, N. Xama, and L. Wu. 2022. Recent trends and perspectives on defect-oriented testing. In *Proceedings of the 2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. 1–10. DOI: http://doi.org/10.1109/IOLTS56730.2022.9897647

[7] G. Insinga, M. Battilana, M. Coppetta, N. Mautone, G. Carnevale, M. Giltrelli, P. Scaramuzza, and R. Ullmann. 2023. Density-oriented diagnostic data compression strategy for characterization of embedded memories in automotive systems-on-chip. In *Proceedings of the 2023 IEEE European Test Symposium (ETS)*. 1–6. DOI: http://doi.org/10.1109/ETS56758.2023.10174126

[8] P. Bernardi, G. Insinga, G. Paganini, R. Cantoro, P. Beer, M. Coppetta, N. Mautone, G. Carnevale, P. Scaramuzza, and R. Ullmann. 2022. Optimized diagnostic strategy for embedded memories of automotive systems-on-chip. In *Proceedings of the 2022 IEEE European Test Symposium (ETS)*. 1–6. DOI: http://doi.org/10.1109/ETS54262.2022.9810445

[9] Omanakuttan Sheela Nisha and Dr.K. Siva Sankar. 2015. A survey on the architecture for an efficient memory built in self-test for configurable embedded SRAM memory. *Int. J. Mod. Trends Eng. Res.* 2, 3 (2015). Retrieved from https://api.semanticscholar.org/CorpusID:26875254

[10] Abraham H. Landzberg. 1993. *Microelectronics Manufacturing Diagnostics Handbook*. Springer. DOI: http://doi.org/10.1007/978-1-4615-2029-0

[11] P. Bernardi, M. Rebaudengo, and M.S. Reorda. 2003. An efficient algorithm for the extraction of compressed diagnostic information from embedded memory cores. In *Proceedings of the EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.03TH8696)*, Vol. 1. 417–421 vol.1. DOI: http://doi.org/10.1109/ETFA.2003.1247736

[12] J.T. Chen, J. Khare, K. Walker, S. Shaikh, J. Rajski, and W. Maly. 2001. Test response compression and bitmap encoding for embedded memories in manufacturing process monitoring. In *Proceedings of the International Test Conference 2001 (Cat. No.01CH37260)*. 258–267. DOI: http://doi.org/10.1109/TEST.2001.966641

[13] N. Campanelli, T. Kerekes, P. Bernardi, M. De Carvalho, A. Panariti, M. Sonza Reorda, D. Appello, and M. Barone. 2010. Cumulative embedded memory failure bitmap display & analysis. In *Proceedings of the 13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*. 1–6. DOI: http://doi.org/10.1109/DDECS.2010.5654683

[14] Jin-Fu Li, Ruey-Shing Tzeng, and Cheng-Wen Wu. 2002. Diagnostic data compression techniques for embedded memories with built-in self-test. *J. Electronic Testing* 18, 4-5 (08 2002), 515–527. DOI: http://doi.org/10.1023/A:1016557927479

Received 13 December 2024; revised 14 July 2025; accepted 22 August 2025