Density-oriented diagnostic data compression strategy for characterization of embedded memories in Automotive Systems-on-Chip

Density-oriented diagnostic data compression strategy for characterization of embedded memories in Automotive Systems-on-Chip

G. Insinga, M. Battilana
Dipartimento di Automatica e Informatica
Politecnico di Torino, Italy

M. Coppetta, N. Mautone, G. Carnevale,
M. Giltrelli, P. Scaramuzza, R. Ullmann
INFINEON Technologies, Germany and Italy

***Abstract*—Embedded System-on-Chip (SoC) memory requirements in the Automotive industry are constantly growing. For this reason, memories occupy a significant part of Automotive SoC's die area, increasing the defect probability inside the embedded storage. Automotive SoC manufacturers need to deeply test their embedded memories as they are one of the significant contributors to the yield of their devices. The test effort increases for the characterization of new technologies and new families of devices that need to be characterized by the manufacturers. These tests generate a massive quantity of diagnostic information that is incredibly valuable for designers and technology experts. This diagnostic information can be analyzed to identify and correct possible weaknesses and misbehavior. The easiest way to collect memory diagnostic information consists of failure bitmaps in which each fault is saved as coordinates. This method is the simplest solution to implement. However, logging the coordinates of every fault may generate an unmanageable quantity of data. This problem is exacerbated when there is an on-chip limitation on the amount of data that can be saved or transmitted to the external world.**

This paper presents an optimized on-chip compression algorithm that allows to reduce the required on-chip memory to store diagnostic information during embedded memory testing. This solution allows the reconstruction of a failure bitmap, generating a topological representation of the density of the failings bits in the embedded on-chip memory. The proposed approach effectively reduces the used storage to a fraction with respect to the one used by the original failing bitmap. The algorithm uses a coordinates-based approach, in which the memory is logically divided into equally divided sectors. The small time overhead introduced by the algorithm is compensated by the ability to achieve optimal space utilization.

***Index Terms*—Automotive SoC, reliability, memory diagnosis, compression**

I. Introduction

Testing is crucial to ensure that each commercialized device is within the specifications [1]. Embedded on-chip memories (eMemories) used in the Automotive industry occupy a large portion of the overall die area. Given their considerable dimension, memories have a significant effect on yield. For this reason, manufacturers deeply test their devices, collecting as much information as possible about their design while checking the covering of all the product requirements. Technology experts and designers can then use this data to strengthen their designs and devise countermeasures for the most common failure scenarios. So, all the data collected help manufacturers to enhance their devices' yield.

Two common approaches used to efficiently test the memories are software and hardware-based. Software-based approaches are the simplest and the most used solutions for memory verification. These methods use the external ATE and the integrated CPU of a SoC to perform the memory verification. The software approaches are the most flexible because they are easily modifiable and adaptable at any time. The main drawback of these approaches is the time overhead with respect to the hardware-based ones.

Specialized hardware modules have been designed and included in the devices to speed up the execution of the memory tests [2] [3] [4] [5] [6] [8] [9]. A classic approach consists of implementing an hardware Built-In Self-Test (BIST) module to perform at-speed verification of the eMemories. These hardware modules share a different flexibility than their software counterparts but are able to significantly decrease the test time.

During the early characterization steps of the devices, manufacturers perform a complete test suite to identify the weaknesses of their memories. This characterization is especially useful for new designs and technologies. One of the classic steps that manufacturers perform is aim at validating the chosen reference current for the eMemories of their devices [10]. This reference value is used to evaluate and compare the value stored on a bit cell, discerning between 0s and 1s. Executing memory verification while shifting the reference current of the sense amplifiers from the nominal value can highlight recurring failings patterns. The more the reference current is shifted, the more failing patterns appear. If the dedicated diagnostic data storage is limited, it is crucial to retain valuable information about these failing patterns. Indeed, in case of a large number of faults, not all diagnostic information can be saved using lossless encoding algorithms. Nevertheless, a valuable information for Designers and technology experts is the fault density along the DUT memories, or in other words, the topological distribution of the faults even without knowing the exact fail position.

A crucial part of the test process is to enhance the storage efficiency of the diagnostic information. Especially during the initial device characterization, testing the memories may generate a massive amount of data that need to be collected and analyzed. A possible solution [7] to collect diagnostic information accumulates failures in a bit failing map. Such a map is generated by a matrix representation of the memory under test. In the resulting matrix, each tested bit is reported one by one. This algorithm is memory intensive as each bit of the memory has to be transmitted to the external Automatic Test Equipment (ATE) for analysis purposes. Communications

between an ATE and the Device Under Test (DUT) are very time-consuming and must be limited to avoid unnecessary test time overhead.

A different solution [11] compacts diagnostic information in shape-encoding segments called slices. This lossless representation can significantly reduce the memory needed to represent the diagnostic information for specific failure shapes. Together with the approach in [7], also this solution cannot cover 100% of fail cases within a reasonable time and dedicated on-chip memory space.

The solution proposed in this paper presents an innovative and optimized compression solution to collect memory diagnostic information. By resorting to on-chip computation, memory is represented in equally divided sections called pixels. Using this approach, the equally divided sections that are encoded using a color-based association to highlight possible recurring patterns in memory failings. Our solution is able to perform data collection with minimal time overhead while gathering high-quality topological diagnostic information with minimal memory requirements.

The paper is organized in the following way: in Section II, the internal memory structure is presented and analyzed. Particular attention has been posed to provide information about memory characterization using different reading reference values. In Section III, we explain the proposed solution, including all steps needed to encode memory diagnostic information using our optimized compression algorithm and how to color-encode them. Section IV highlights our experimental results on an Automotive SoC made by Infineon Technologies. In Section V, the obtained results are evaluated, providing some conclusive considerations.

II. BACKGROUND

A. Memory organization

Bits in the embedded memories are arranged as a matrix. Each row in this matrix is called wordline, while the columns are called bitlines. In a typical non-volatile memory, a wordline contains a certain number of pages which are composed of a predefined number of bits. For the user, a page represents the smallest addressable unit.

Wordlines are then combined to form a physical sector. Finally, physical sectors are combined to form a memory bank.

Big memories are made by creating multiple independent replicas of the memory banks, Fig. 1.

When a read operation is issued, the Sense Amplifier (Fig. 2) assigns the corresponding bit cell value. This operation is repeated for all the bits of the page being read. The Sense Amplifier is an analog component connected to the bitlines that compares the current driven by the bit cell against a reference value. If the value is above the reference, the 0 value is assigned to the bit cell, 1 otherwise.

B. Characterization and Memory Current Margin Test

Memory units may present many faults in the early steps of technology development. Characterization processes are used to perform studies about statistical topological distribution that may be present due to non-optimal architectural design or specific process variation or different working parameters (temperature, frequency and so on) [10]. Many verification steps are

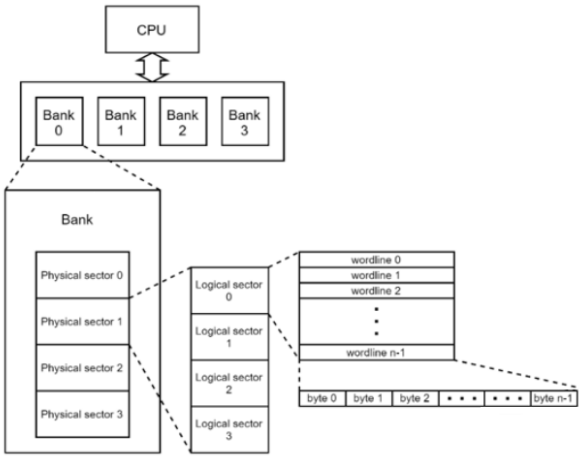


Fig. 1: Embedded memory organization

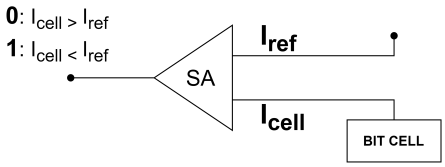


Fig. 2: Sense Amplifier

performed to find fault distribution when changing the Sense Amplifier reference current and observing the corresponding bit misbehavior. This process can highlight recurring defects in the memory behavior, such as non-uniform topological faults distributions, that are process and technology-dependent.

The resulting diagnostic information can be seen as coordinates that describe the physical location of the faults present for the selected margin.

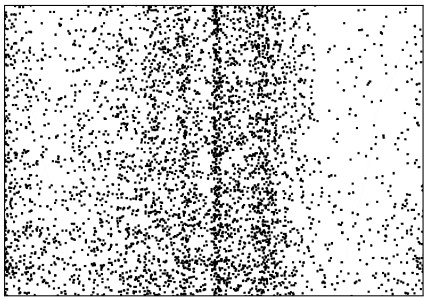


Fig. 3: Standard bit fail map generated shifting the reference current of the Sense Amplifiers

C. Fault topological distribution

Technology experts are interested in topological failings patterns with respect to their exact positions. They benefit from a compressed overview of the failings in the memory that shows the fault density more than their precise locations. The benefit is especially evident for fault-dense DUTs, where the number of faults limits the ability to represent a large portion of the memory. Also, the exact coordinates of every fault are not needed for this type of characterization.

A topological failings pattern corresponds to an increased probability of faults in a specific area of the physical memory.

Method	Density accurate	Storage	On-Chip
Landzberg [7]	Yes	High	Yes
Chen [4]	No	Variable with resolution	Possible
Bernardi [11]	Yes	Low	Yes
Proposed	Yes	Variable with resolution	Yes

TABLE I: Comparison between features of various approaches

Multiple factors can interfere with the correct reading, altering the programming state of the cells and then generating a topological failings pattern. Fig. 3 represents the failing information using a standard Bitmap in which each black dot corresponds to a failure. Failures are focused on the center, highlighting a topological failing pattern.

D. Algorithms for diagnostic data collection

The most straightforward algorithm for collecting the diagnostic data from the embedded memory test is the one described in Landzberg et al. [7]. In his approach, the ATE directly accesses the memory under test, reading bit by bit. The failures found during these procedures are directly exported through their coordinate with no further manipulation. The failure constellation can be reconstructed by simply analyzing the collection of exported coordinates. On the other hand, Bernardi et al. [11] developed an algorithm to collect failure information using the concept of encoding and pattern recognition. Faults are collected in color-encoded segments called slices. In this way, the authors were able to be more memory efficient with respect to [7]. Lastly, Chen et al. [4] proposed a compression method, able to reduce the amount of memory needed to represent a failure constellation at the cost of the accuracy of the representation.

III. PROPOSED APPROACH

The proposed solution is based on the compression of diagnostic failing information during the execution of an on-chip memory verification. Our solution can overcome the explosion in memory requirements required by [7] and, for sparse faults, by [11] to represent fault-dense scenarios.

The base element of the compression algorithm is called pixel, which represents a portion of the memory of the DUT. A pixel is composed of a configurable number of wordlines and bitlines. These two parameters are freely configurable to achieve various levels of compression. The smaller the pixel's dimension, the greater the resolution at the expense of the memory requirements. Each pixel comprises a two-dimensional coordinate and a counter representing the number of faults for the represented memory portion.

Table I shows a comparison between our proposed approach and other diagnostic data collection algorithms. Our proposed approach is topologically accurate, showing the density of memory fails in different areas of the memory. This characteristic is in common with the algorithms shown in Bernardi et al. [11] and Landberg [7]. As for storage requirements, similar to the algorithm of Chen et al. [4], it varies based on the chosen resolution. In common with Bernardi et al. and Chen, the proposed approach's memory requirements are lower than the Landzberg approach. Lastly, all the compared methods are executable on-chip (with [4] being originally tested with additional hardware and tester capabilities).

The proposed approach does not require additional hardware in the DUT. It is also compatible with BIST-based memory tests, as well as the CPU-based ones. Moreover, the proposed approach can be used in conjunction with the methods [7] and [11] by dynamically switching to the compression approach when the available dedicated on-chip storage is running low.

A. SLAC Pixel structures

Each pixel is encoded into the on-chip memory as a structure that includes three parameters: x and y coordinates and the number of faults. This approach allows the algorithm to achieve the minimum possible used space by exploiting bit-wise operators to manage the structure parameters. Each pixel is represented using 4 Bytes in the following way (Fig. 4):

- 1 Byte, X pixel coordinate
- 1 Byte, Y pixel coordinate
- 2 Bytes, faults counter

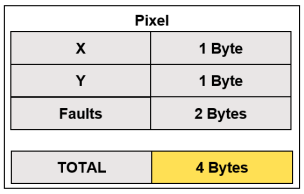


Fig. 4: Pixel encoding structure

B. On-chip memorization of the encoded information

When a new fault is discovered and processed by the proposed algorithm for compression:

- The corresponding pixel X and Y coordinates are identified. These two values are computed in the following way:

$$X_{\text{Pixel}} = \left\lfloor \frac{X_{\text{Fault}}}{bitsPerPixel} \right\rfloor \quad (1)$$

$$Y_{\text{Pixel}} = \left\lfloor \frac{Y_{\text{Fault}}}{bitsPerPixel} \right\rfloor \quad (2)$$

  Where $bitsPerPixel$ corresponds to the pixel's dimension terms of number of bitlines or wordlines. X_{Fault} and Y_{Fault} represent the fault coordinate.
- The pixel with the corresponding X and Y coordinates is indexed, and the counter used to represent the number of faults is increased by one.

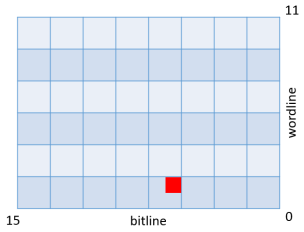
To optimize the algorithm's speed, available memory has been organized using a set-associative approach as in Bernardi et al. [11]. Given a number of N sets, the allocated on-chip memory for the diagnostic data is equally divided into N portions. When a new fault is found, the corresponding pixel is indexed using the X coordinate. The pixel location is used to generate the set and tag in the following way:

$$SET = X_{\text{Pixel}} \% N \quad (3)$$

$$TAG = \frac{X_{\text{Pixel}}}{N} \quad (4)$$

Each SET represents a different list that is indexed once a new fault is found. A linear search is then performed over all pixels already present in the selected SET, checking for correspondence using the TAG.

For example, Fig. 5 depicts a memory composed of 16 bitlines and 12 wordlines. In this example, each pixel represented by a square composed of 2 bitlines/wordlines each ($bitsPerPixel = 2$). The red square represents a fault located at coordinates $X = 6$ and $Y = 1$. Fig. 5 shows an example of how the SET and TAG of the pixel are computed starting from a given fault.



$X = 6$ $\quad X_{\text{Pixel}} = 3$

$Y = 1$ $\quad Y_{\text{Pixel}} = 0$

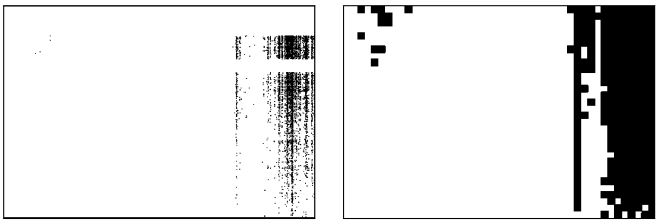
$SET = 1$

$TAG = 0$

Fig. 5: Example of set and tag computation starting from a fault

Another important aspect is that a pixel is stored in the on-chip memory only if there is at least one fault. The first fault will trigger its creation. This approach can reduce the storage used to the minimum, avoiding storing useless information.

Fig. 6a represents a Bitmap on a memory bank using a compaction algorithm [11]. This algorithm is based on encoding and compacting faults while performing on-chip memory verification. To reduce the used storage, faults are compacted and encoded as contiguous slices. In this case, only a portion of the memory could be represented due to the on-chip storage limitation. Fig. 6b represents the same failing information using our compression algorithm. The compressed bitmap allows to overcome the storage limitation. Each square corresponds to 128 wordlines and 128 bitlines and has been colored if the portion has at least one fault.



(a) Bernardi et al. (b) Proposed

Fig. 6: Bitmap on one single bank with sparse faults, presenting a topological failings pattern towards the right

Fig. 7 shows a zoomed version of the example in Fig. 6 reconstructed starting from the data produced by the algorithm of Chen et al. [4]. The resolution was set to the maximum allowed to represent the entire memory in the dedicated on-chip memory reserved for diagnostic information. The red dots in the picture represent faults, while the blue dots represent the area in which the presence of faults is uncertain. For this uncertainty, Chen et al. method is unsuitable for characterization studies, where the topological fault distribution is the most crucial parameter for designers and technology experts.

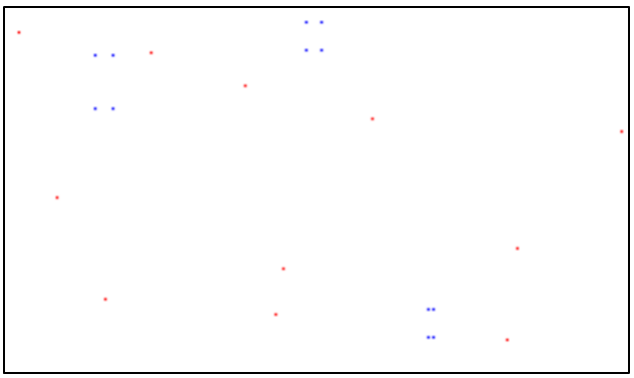


Fig. 7: Example of reconstructed memory using Chen et al. algorithm

C. Color gradient for visualization

To better visualize the generated pixels, we developed a post-verification tool that creates a colored representation of the memory under test. This tool runs offline on the tester or on an external computer after collecting the pixels from the DUT. The corresponding color depends on the number of faults represented by the pixel. A naive approach would be to divide multiple ranges of faults and assign each to a different color. The problem with this solution is that the entire spectrum must be manually assigned. To overcome this limitation, the value is mapped into the three RGB components. Each pixel that has at least one fault is color mapped using three different sine functions, centered respectively at 0.0, 0.5, and 1.0 in the following way:

$$\begin{aligned} \text{RED} \quad & sin(\pi \cdot faults_{norm} - \frac{\pi}{2.0}) \\ \text{GREEN} \quad & sin(\pi \cdot faults_{norm}) \\ \text{BLUE} \quad & sin(\pi \cdot faults_{norm} + \frac{\pi}{2.0}) \end{aligned} \tag{5}$$

Where $faults_{norm}$ refers to the normalized number of faults with respect to the maximum number of representable ones by a pixel (on turn dependent by its dimension):

$$faults_{norm} = \frac{faults_{count} - 1}{faults_{max}} \tag{6}$$

Value of $faults_{max}$ can be offline trimmed to highlight better topological patterns that appear with higher or lower faults.

The plot of the three sine functions is available in Fig. 8. Coloring examples are depicted at Fig. 9.

The post-processing function is applied offline once the failing information have been downloaded from the test machine. A different color schema can be applied depending on the needs, making this approach even more flexible. In the shown case, the higher the fault density, the more red the pixel will be. A lower number of faults will make the color bluer.

IV. EXPERIMENTAL RESULTS

This section shows our experimental results collected with the proposed compression algorithm using an Automotive SoC device made by Infineon.

The following results are collected while characterizing the eMemories sense amplifier's reference current. In our

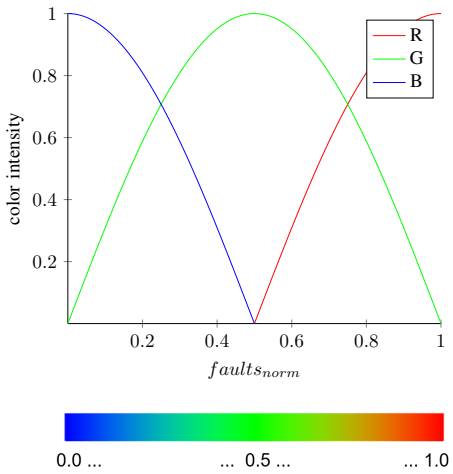


Fig. 8: Mapping function from pixel difference to RGB components

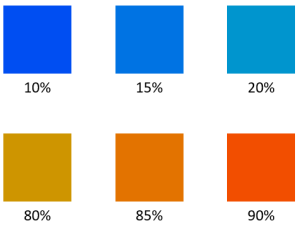


Fig. 9: Example of pixel coloring based on the percentage of faults

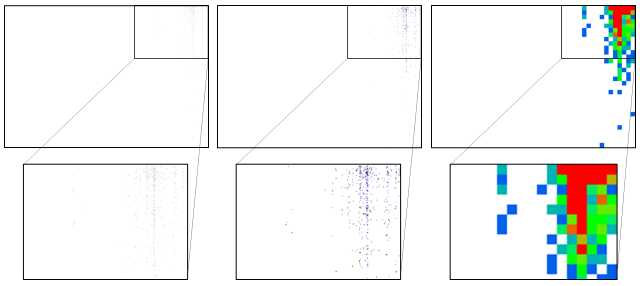
experiments, we had a limitation of an embedded 24KB of memory to collect diagnostic information from these tests. All the results and pictures that will be shown are contained in this integrated memory. Once this memory is saturated by diagnostic information, the data logging stops leading to a loss of information. The following results show the performance of the various bit mapping algorithm with this memory limitation.

A. *Data visualization*

The following pictures illustrate how the same fault configurations are exported and visualized using different approaches. The main goal of our approach is to help designers and technology experts visualize the failure density of their DUT, so the main focus of this paragraph is to show how this density can be discerned using different methods.

The following pictures have been obtained by gradually shifting the reference current of the sense amplifiers, a standard test performed during the characterization of new devices.

Fig. 10 shows the first of these current reference shifts performed on a single bank of the memory under test. In this picture, there is a comparison of three different diagnostic data collection algorithms. In Fig. 10a the simple Landzber bitmapping approach [7], in Fig. 10b the compaction algorithm of Bernardi et al. [11] and in Fig. 10c the proposed compression approach. As can be seen, by using A, identifying the most fault-dense area is not immediate. Even a zoomed-out version, centered on the most fault-dense zone, is challenging to analyze and observe. This situation is slightly better for the compaction algorithm in 10b, where the colors help identify the faults. Using 10c, the topological distribution of faults is immediately evident, indicating the most fault-dense zones on the top right corner of the memory bank. In the following, we will keep the Bernardi approach as a reference for a lossless algorithm.

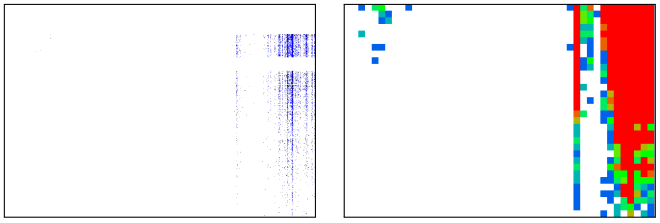


(a) Landzberg (b) Bernardi et al. (c) Proposed

Fig. 10: Bit fail map visual comparison with slight reference current shifting (increment #1)

Fig. 11 further increases the shift in reference current of the sense amplifiers. The fault density is again more visible in Fig. 11b, showing the reconstruction of the bank with the proposed approach. The algorithm of Bernardi et al. in Fig. 11a tries to show the same information in a lossless compaction of the data but is not able to represent the faults in the top left corner of the bank that are starting to fail under the tighter reference current constraints. The Bernardi et al. approach saturates the 24KB of available memory to represent the fault information, reducing the amount of helpful information that reaches the ATE at the end of the test.

Fig. 12 and figure 13 clearly show that the lossless approach cannot give helpful information about the fault happening in the memory under test. The 24KB limit is reached when just a portion of the total faults are discovered, and a vast percentage of memory is not represented at all. On the other hand, the proposed algorithm can correctly represent all the faults in the memory, albeit in a compressed manner.

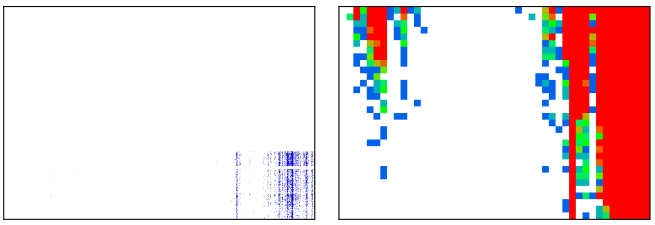


(a) Bernardi et al. (b) Proposed

Fig. 11: Bit fail map visual comparison reference current shift (increment #2)

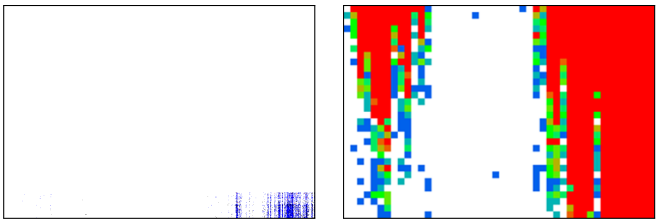
B. *TIMING*

In this section, the timing overhead of the various algorithm is analyzed. The Landzberg approach is always the fastest and will be used as the overhead computation reference. The scenarios considered to make this consideration have a variable amount of faults randomly distributed along the memory under test. As shown in Table II, the overhead of Bernardi et al. and the proposed approach grows with the number of faults found. The overhead of the proposed approach is always lower than



(a) Bernardi et al. (b) Proposed

Fig. 12: Bit fail map visual comparison reference current shift (increment #3)



(a) Bernardi et al. (b) Proposed

Fig. 13: Bit fail map visual comparison reference current shift (increment #4)

the Bernardi et al. one, especially with the growing number of faults. For example, at 5500 faults, the Bernardi approach has a 14.63% of overhead, while the proposed stops at 9.60%.

	Time overhead (% wrt to Landzberg)	
# Faults	Bernardi et al.	Proposed
100	2.33	2.28
250	2.67	2.48
500	3.02	2.67
1000	4.18	3.25
2000	6.81	4.48
4000	11.03	7.35
5500	14.63	9.60

TABLE II: Comparison between timings of various approaches in a memory with randomly distributed faults (with % computed over the overhead with respect to the Landzberg approach)

C. Memory requirements

This section is focused on the memory needed to represent the various scenarios depicted in figures from 10 to 13. These cases are taken from a real characterization step performed by Automotive SoC manufacturers. The diagnostic information collection limit is 24KB of on-chip dedicated memory. When this memory is saturated, no additional data is saved, and the related information is lost. It would be possible to notify the ATE of the saturation, wait for it to download the 24KB, and restart the logging. However, this communication with the ATE would be unmanageable for fault-dense scenarios and cases of multiple devices tested in parallel, such as in a mass production environment. As seen from Table III, the case in Fig. 10 is the only one in which all three analyzed methods can fit the diagnostic data in 24KB of dedicated on-chip storage. The Landzberg approach is already near this value, with a memory requirement of 20.62KB. The Bernardi et al. approach requires 90% less with requirements of 2.19KB, a reduction of 90% with respect to Landzberg. Finally, our proposed approach required 0.60KB, a reduction of 97% with respect to Landzberg and of 72,6% with Bernardi et al. In the other cases, up to 13, the lossless algorithms saturates the 24KB available, with the proposed approach requiring only 4.68KB in the worst case. For a visual comparison, Fig. 13 clearly shows the limitation of the lossless approaches. With 24KB, only a tiny portion of the memory is represented. In this case, our compression approach can represent all the memory under test, with information about the fault density in its various zone, invaluable information for Automotive SoC manufacturers.

	Size (KB)		
	Landzberg	Bernardi et al.	Proposed
Fig. 10	20.62	2.19	0.60
Fig. 11	**24.0**	**24.0**	1.82
Fig. 12	**24.0**	**24.0**	2.04
Fig. 13	**24.0**	**24.0**	4.68

TABLE III: Comparison between memory requirements of various approaches

V. CONCLUSIONS

In this paper, we described a novel algorithm that compresses the diagnostic information generated during the test of the embedded memories of Automotive SoC devices. The shown results were taken from a real device made by Infineon, and they show the validity of our approach both in terms of memory requirements and the usefulness of the collected data. The proposed algorithm is particularly effective during device characterization, where the topological fault distribution is the most crucial parameter for technology experts and designers.

REFERENCES

[1] A. van de Goor, G. Gaydadjiev e S. Hamdioui," Memory testing with a RISC microcontroller" in Proc. on Design, Automation and Test in Europe, Dresden, 2010.

[2] P. Bernardi et al. "Cumulative embedded memory failure bitmap display & analysis" in IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, 2010.

[3] S. Abhas, M. K. Gurram e A. Abhijit, "Controller Architecture for Memory BIST Algorithms" in IEEE International Students' Conference on Electrical, ELectronics and Computer Science (SCEECS), 2020.

[4] J. Chen, J. Khare, K. Walker, S. Shaik, J. Rajsky e W. Maly, "Test response compression and bitmap encoding for embedded memories in manufacturing process monitoring" in Proceedings International Test Conference 2001.

[5] P. Bernardi et al. "An efficient algorithm for the extraction of compressed diagnostic information from embedded memory cores" in 2003 IEEE Conference on Emerging Technologies and Factory Automation.

[6] I. Schanstra et al. "Semiconductor Manufacturing Process Monitoring using Built-In Self-Test for Embedded Memories" in Proceedings International Test Conference 1998.

[7] A. L. Landzberg e R. Van Nostrand, Microelectronics Manufacturing Diagnostics Handbook, New York, USA, 1993.

[8] H. WonGi, C. JungDai e C. Hoon, "A programmable memory BIST for embedded memory" in International SoC Design Conference, 2008.

[9] C.-H. Tsai e C.-W. Wu, "Processor-programmable memory BIST for bus-connected embedded memories" in Proc. of the Design Automation Conference, 2001.

[10] P. Bernardi et al., "Recent Trends and Perspectives on Defect-Oriented Testing," 2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS), 2022, pp. 1-10, doi: 10.1109/IOLTS56730.2022.9897647.

[11] P. Bernardi et al., "Optimized diagnostic strategy for embedded memories of Automotive Systems-on-Chip," 2022 IEEE European Test Symposium (ETS), 2022, pp. 1-6, doi: 10.1109/ETS54262.2022.9810445.