A Novel Approach to Extract Embedded Memory Design Parameter Through Irradiation Test

A Novel Approach to Extract Embedded Memory Design Parameter Through Irradiation Test

P. Bernardi, G. Insinga, N. Kolahimahmoudi
Dipartimento di Automatica e Informatica
Politecnico di Torino, Italy

***Abstract*—With the capability improvements in modern System-on-Chips (SoCs), the complexity of SoCs is increasing. Thus, manufacturers are investing heavily in designing and testing their devices. This complexity is causing a continuous expansion in the size of embedded memory structures. As a result of the shrinking dimensions of the transistors, memories are increasingly susceptible to Multiple Bit Upsets due to cosmic radiations.**

Testing memories requires more details about the internal hardware configurations. However, these details are not provided to the final customer, who is left with inexplicable effects.

This paper proposes a new method to reconstruct architectural details from embedded SoC memories. This method extracts memory design parameters from Multiple Bit Upsets (MBUs) generated through a single irradiation test. The algorithm was tested on around 5,500 randomly generated memories. Each memory was injected with 100 Multiple Event Upsets (MEUs). The algorithm was set to test for each memory 20, 40, 60, 80, and 100 MEUs to validate the proposed approach. Alongside the correct memory design configuration (MDC), the algorithm found other possible MDCs. The quantity of these equivalent configurations decreased with the increment of the considered MEUs. This number decreased to an average of 2 equivalent MDCs when considering 100 MEUs.

***Index Terms*—SoC, reliability, memory diagnosis, memory feature extraction, irradiation test**

I. Introduction

In the past few years, the complexity of modern SoCs has exponentially grown with each generation. More advanced nodes and more transistors per area unit are the classic improvements manufacturers have for their devices. These improvements affect the customers that make more advanced and memory-dependant programs and pose additional risks and challenges in memory testing. Smaller transistors are more prone to process variations, decreasing the yield of the devices. However, the problems do not stop during production as, even after that, there are risks to the reliability of the systems. In safety-critical applications such as the automotive field, the final SoC customers are interested in the reliability of their program. These customers want to test the SoCs against the most common issues arising during their systems' operational life [1]. One typical phenomenon that electronic devices encounter during their operation is due to cosmic radiations made of high-energy charged particles [2]. These particles can interact with the transistors and flip their state, generating Single Event Upsets (SEUs) [3] [4]. Countermeasures that span the classic error correction code (ECC) have been devised to mitigate the effects of cosmic radiations. For instance, correction mechanisms that use duplication or triplication of the most critical hardware or direct transistor modifications take the name of radiation hardening [5].

The effect of these high-energy particles is most prominent in the embedded memories of devices. Memory matrix structures are especially prone to bit flip because of their small transistors placed in a grid, one next to the other. Even worse, due to their proximity, multiple-bit flips can be caused by a single particle generating Multiple Even Upsets (MEUs) [6]. Such events may happen, for example, when a charged particle hits the intersection between two or more cells. These phenomena are well studied in the literature [7] as they are particularly problematic in safety-critical environments. For instance, in automotive [8] applications or in space applications [9] where there is a higher level of radiation with respect to the sea level.

To study the effects of cosmic radiation, final SoC customers test their devices in irradiation facilities where devices are "bombarded" under a heavy radiation flux. These procedures accelerate the appearance of SEUs and MBUs and allow the programmers to see the effects on the execution of their programs. However, this study method is expensive, time-consuming, and needed for any application. Due to the high cost of the irradiation tests, alternative methods exist to evaluate the effect of transient faults in memories in SoCs. One of the cheapest approaches is to test the memory through fault injection techniques. These techniques are applied in software (i.e., using the proposed method), and the test cost is much lower than the irradiation test. Moreover, these tests can be applied faster, meaning less test time. In fault injection, the user must know the architectural details of the memories. However, these details are currently unavailable, and memories are thus seen as black boxes. In a black box representation, a memory can be represented by words made of a certain number of bits, placed one after the other, with a straightforward organization. Real embedded memories, however, can be organized with multiple physical parameters such as column and row mirroring, scrambling, and so on, as described in [10]. Considering the aforementioned details of different memory representations, fault injection of memories is only possible when architectural details are known. Mixed approaches are shown in [2] [11] that uses real (and limited) irradiation tests data to assess the effect of Radiation-Induced Soft-Errors in different applications. This article suggests how to use data from static and possibly short irradiation experiments to extract the architectural details of the memories and, once known, perform fully accurate fault injections. The presented algorithm correctly reconstructs the internal memory organization by using the effect of the MEUs from irradiation tests. This algorithm can reconstruct multiple parameters such as scrambling, mirroring, and column/row swapping. It operates by exploiting the property of memory MEUs to be physically close to each other. A thousand memory tests with multiple organizations and with multiple kinds of MEUs were used to validate this approach. These tests always gave the correct memory organization and some functionally equivalent ones. The paper is organized in the following way: in Section II, the internal memory structure is presented and analyzed, followed by the difference between logical and physical addresses. Then the focus shifts to the irradiation tests and how they can generate the phenomenon of MEUs with the dimensions of modern memories. Section III explains the proposed solution, including all steps needed to reconstruct the memory parameters starting from the corresponding MBUs. Section IV highlights the experimental results on a vast series of tested memories with different organizations. In Section V,

979-8-3503-2599-7/23/$31.00 ©2023 IEEE

the obtained results are evaluated, and some conclusions are provided.

II. BACKGROUND

This section analyzes the embedded memories' most important aspects and how they correlate with the irradiation tests that generate the data that can be used for our approach.

A. Embedded Memories general organization

Embedded memories are organized in regular structures that are easy to replicate and manage [10]. A general overview is depicted in Fig. 1. The topmost block comprises several independent memory banks. These are entirely separate structures that can be placed around the die surface. Each bank is then subdivided into several physical sectors (and may contain a Memory Built-in Self Test (MBIST) to speed up testing operations). Physical sectors are then subdivided into Logical sectors that are divided into Wordlines. Wordlines are composed of multiple words that contain a certain number of Bytes. Bytes represent the minimum granularity accessible by the final user.

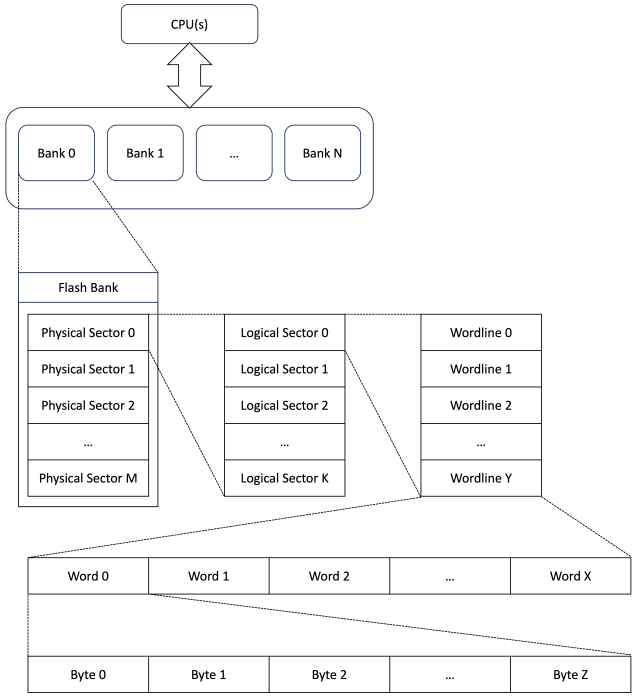

Fig. 1: A general organization of embedded memories.

B. Logical vs. Physical addresses

In the programmers' view, the logic memory is a progression of words going from the lowest index to the highest one, as seen in the left part of Fig. 2. In this view, the memory is a "black box" in which bits in words are again organized one after the other from the lowest indexed one to the highest indexed one. Standard memories, however, are organized in a more complex structure to improve their reliability and chances of being repairable with some integrated spare components.

Manufacturers use a set of Memory Design Configurations (MDCs) to modify the organization of their memories. Referring to Fig. I, some of the MDCs are:

- WORDS: The total number of words in the memory
- MUX: The number of words in a row (also called wordline).
- BITS: The number of bits contained in a single word
- BIT ORDER sequence: Describes how the bits of the words are organized in a wordline.
- Mirroring: This parameter decides if the memory has portions repeated in alternated order. The mirroring comprises either column (bitline) and/or row (wordline) mirroring.
- BIT SCRAMBLING: Bits in wordlines are divided in blocks in which each block i is made of bits with index i^{th}

Fig. 2, 3, 4, 5 and 6 show some possible examples of MDCs. In the figures, a single wordline is shown per example, and consequently, only parameters affecting the rows are shown. However, similar configurations can also be applied to the bitlines (column) of the memories.

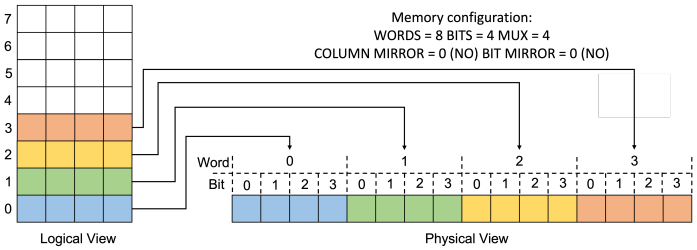

Fig. 2: An example of an MDC with no bitline or bit mirroring.

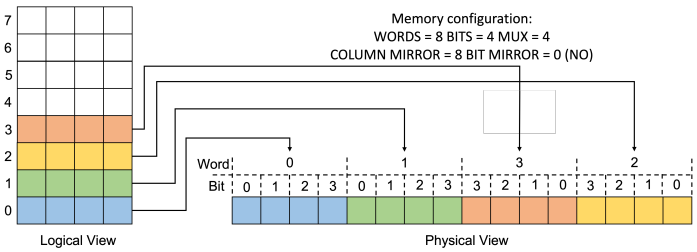

Fig. 3: An example of an MDC with column mirroring

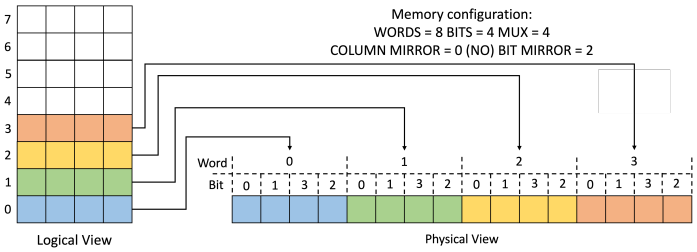

Fig. 4: An example of an MDC with bit mirroring

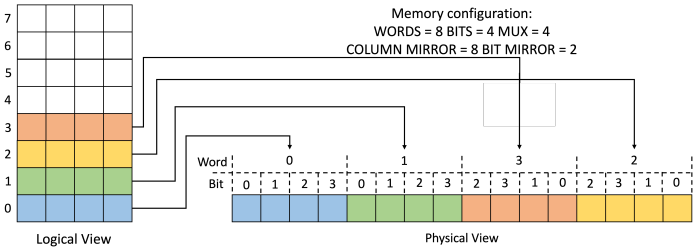

Fig. 5: An example of an MDC with column and bit mirroring

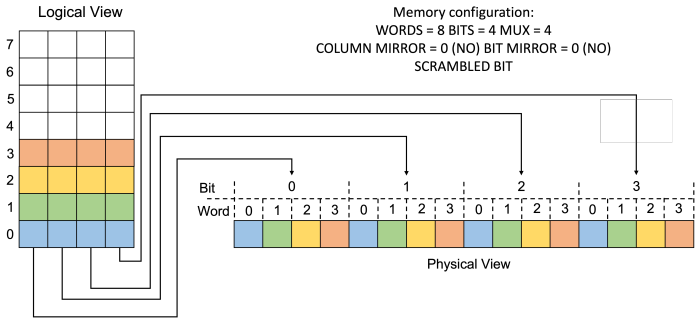

Fig. 6: An example of an MDC with bit scrambling

C. Irradiation test and Multiple Bit Upsets

Due to their small transistor dimension, modern circuits are particularly sensitive to Cosmic radiations [7]. These star-produced radiations are made of high-energy charged particles that can interact with the transistors inside our electronic devices. When interacting with transistors, cosmic radiations can flip the bit of a memory cell of a logic gate, generating a Single Event Upset (SEU) [3] [4]. Since the dimension of the transistors is constantly decreasing, a single charged particle can hit and flip the status of multiple of them, generating what is called Multiple Event Upset [6]. For the case of embedded memories, a MEU directly translates to multiple bit-flips in the data stored in the memory as represented in Fig.7 that shows the Straightforward organization of the memory as an example.

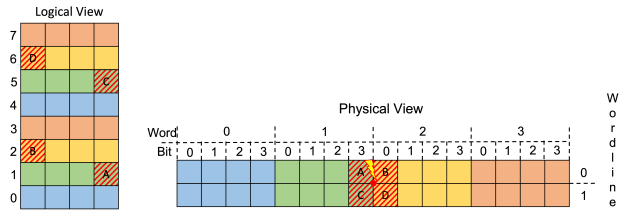

Fig. 7: An illustration of MEUs in the bits of multiple cells.

To study the effect of cosmic radiation, manufacturers can test their devices in irradiation facilities. These testing facilities irradiate the Devices Under Test (DUT) with a radiation flux multiple orders of magnitudes higher than the one present in nature. This testing environment allows manufacturers to collect years of data in hours/days. Focusing on the memories means collecting all the SEU and MEU, possibly from a charged particle hitting a memory cell or an intersection between cells.

D. Fault injections

Software developers are interested in the reliability of their programs in the case of SEUs and MEUs, as the Irradiation test is expensive and slow to perform and is usually performed in a separate facility, requiring additional logistical efforts. Fault injections, on the other hand, are a classic approach for studying the effect of faults in a system without actually waiting for them to happen "naturally" during the device's lifetime or to perform irradiation tests. [12] Faults are "simulated" by the software inside a physically healthy device. An example of fault injection is the bit flip of a location in the memory to observe the effect on the chosen test program. A fault injection experiment involving multiple faults takes the name of the campaign. A fault injection campaign in which the target is to inject single SEUs at a time is performed in multiple steps such as:

1) **Choose a test program**: In this step, a test program is chosen to see the effect of faults.
2) **Perform a golden execution**: During this phase, no faults are injected, and the outputs and timing of the test program are observed. These parameters will be used as a reference to understand the effects of the injected faults.
3) **Fault generation**: The designers choose which circuit part to perform fault injection in this phase. Moreover, they choose the criteria for choosing the location and timing of the faults. Random fault location and timing represent naturally occurring SEUs and MEUs, but need many experiments to obtain meaningful data.
4) **Fault Injection**: A single fault is selected and injected at the desired location and timing.
5) **Result check**: The output of the test program is observed. Depending on the outcome, we can have a go (correct results), no go (wrong results), and timeout (device in hard fault, endless loop, etc.).

Steps 4 and 5 are repeated until all the desired faults are injected. The results are analyzed to understand the most critical part of the execution of the test program.

III. PROPOSED APPROACH

The proposed approach uses an efficient algorithm to extract the MDCs by observing the memory MEUs from irradiation tests. MEUs are generated by the interaction of a charged particle with multiple memory cells. The affected cells can, for example, share a corner or be one in front of the other. For this reason, the algorithm assumes that each set of MEUs is composed of faults near each other, following the principle that a charged particle generates failing patterns (MEUs) such as the ones illustrated in Fig. 8. As can be seen, each bit in the failing pattern is next to at least another failing bit in either the same row or column. The proximity of the MEUs is the central assumption of our algorithm that can test multiple possible memory organizations to extrapolate the correct ones and exclude the wrong ones as soon as possible.

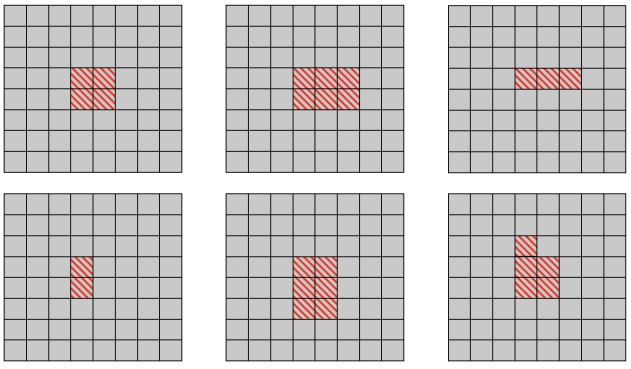

Fig. 8: A general view of MEUs in memory cells.

A. Memory reconstruction

The proposed algorithm's aim is based on the assumption that each MEU from the irradiation tests generates faults located near each other, as shown in Fig.8 an. These MEUs stem from a charged particle hitting the intersection between 2 or 4 memory cells, causing them to be interpreted as failing bits. So MEU is composed of a variable number of faults. Due to this assumption, the proposed algorithm performs a series of steps for each set of MEUs received:

1) **Guess an MDC**: The aim of the algorithm is set to test all the possible MDCs systematically. It achieves this goal by changing the various parameters one at a time until an

exhaustive search is performed. This approach represents a guess by the algorithm that will then test this MDC to discover if it can be the source of the MEUs.

2) **Fault reconstruction**: This reconstruction is performed considering logical addresses of faults and the selected MDCs. As each MEU is made of a certain number of faults, at the output of this step, the algorithm obtains a set of physical coordinates (one for each fault in the MEU set).
3) **Check the contiguity between the reconstructed faults**: Given the fault coordinates, the distance between each fault is checked. The main objective is to find all faults' locations in close proximity. In other words, a group of fault locations is called MEU when:
   - There are at least two fault locations with a Euclidean distance of 1 from each other.
   - The third fault location can be added to this faults' locations group if it has a Euclidean distance equal to 1 from at least one of two faults' locations.
   - Other faults can be added to the mentioned group if they have a Euclidean distance equal to 1 from at least one of the other faults' locations group. These groups of x and y coordinates should create some patterns, such as the one shown in Fig. 8.

   For instance, in Fig.9, the location of the MEU is compliant with the chosen MDC because the faults are in the same neighborhood. However, as can be seen in Fig.10, the chosen MDC is non-compliant because the faults are in different neighborhoods and separated.

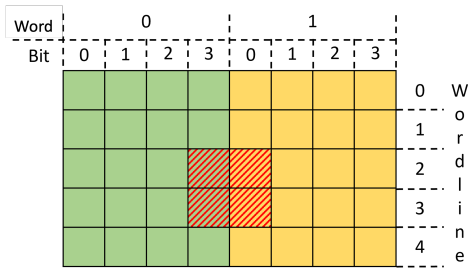

Fig. 9: A scheme of reconstructed memory with the correct MDC.

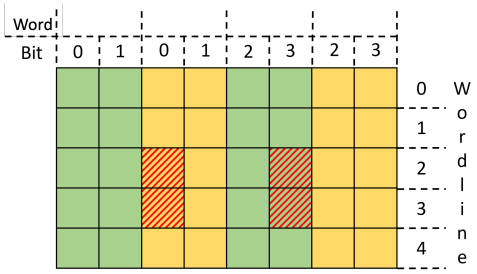

Fig. 10: A scheme of reconstructed memory with the wrong MDC.

4) **Update correct combinations or remove MDC from possible solutions**: If the combination is marked as correct, the algorithm increases a counter that counts how many times the current MDC was identified as correct. Otherwise, no further steps are performed, and the MDC is removed from possible solutions.

The steps from 2 to 4 are repeated for all the MEU sets related to a given memory. The algorithm continues to test until all the MDCs have been tested. Once all the MDCs have been fully explored, the possible combinations are ordered from the most probable to the least probable. The correct memory organization is in the most probable ones. Often the algorithm reports equivalent. The complete algorithm is provided in algorithm1. This algorithm has a programmer's point of view on the proposed approach.

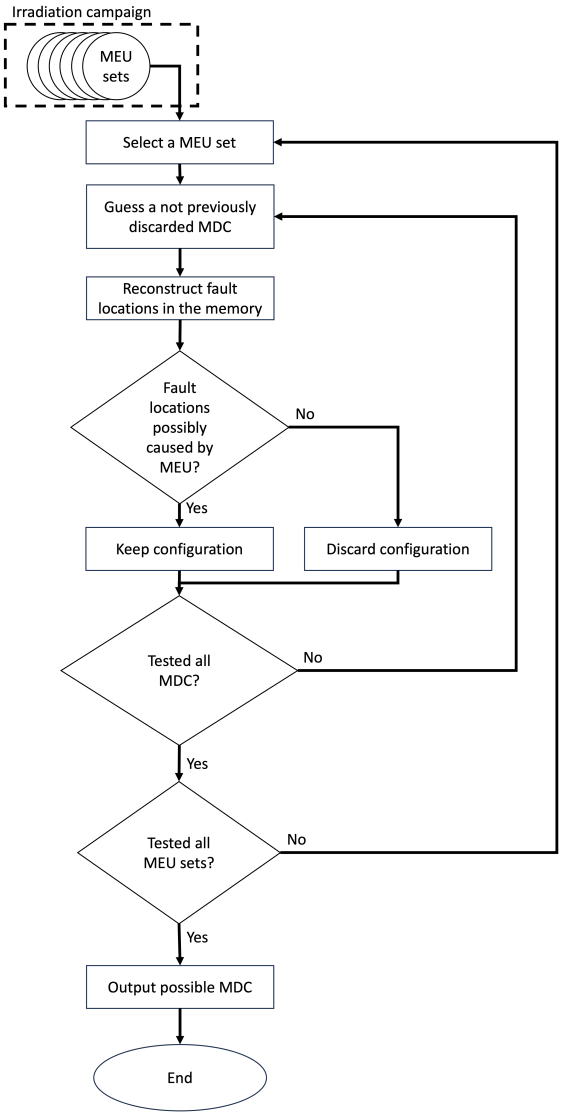

Fig. 11: The flowchart of the proposed algorithm

IV. EXPERIMENTAL RESULTS

The research aims at extracting the memory design parameters, and the proposed approach uses the Algorithm 1. The final results can then be used by test engineers to accurately assess the behavior of their programs in case of radiation-induced MEUs. In order to corroborate the functionality of the proposed approach, a complete simulation and validation environment was developed. This environment comprises multiple scripts that perform a series of steps provided in Fig. 11 to inject the MEUs, and then reconstruct the memory parameters. The Python script responsible for the simulation and injection of the MEUs is called "supervisor script." This supervisor script:

1) Randomly select an MDC made of all the parameters described in the previous section

Memory Design Configurations			
	Min	**Max**	**Steps**
Total Number of Words	2	2^{15}	Exponential (Powers of 2)
Words Per Wordline	2	$2^{\log_2(\text{Total number of words})-3}$	
Bits Per Word	2	2^8	
Wordlines	$\frac{\text{Total Number Of Words}}{\text{Words per Wordline}}$		
Wordlines Mirrored Every	2^0(no mirroring)	$2^{\log_2(\text{Wordlines})}$	Exponential (Powers of 2)
Bitlines	Bits per word * Words per Wordline		
Bitlines Mirrored every	2^0(no mirroring)	$2^{\log_2(\text{Bitlines})}$	Exponential (Powers of 2)

TABLE I: Memory parameters taken into consideration for our experiments

Algorithm 1: Memory parameters extraction algorithm

```
input : A list of MEUs. Each MEU is composed of N
        faults, each with:
          The logical word address of the fault: Addr
          The failing bit inside the word: Q
output: A set of MDCs that can generate the MEUs in
        the list.
foreach MEU in MEUs list do
  foreach Memory Dimension in Possible Memory
   Dimensions list do
    foreach Row Mirroring in list do
      foreach Column Mirroring in list do
        Take the Addr and Q and calculate
         faults' coordinates (n is the number of
         fault locations): (x_i,y_i), (x_{i+1},y_{i+1}), ...,
         (x_{i+n},y_{i+n}) And make a group of
         coordinates by randomly adding one
         coordinate to the group.
        while At least a fault coordinate is
         added to the group in this iteration. do
          foreach Faults' coordinates do
            if The coordinates have a
             Euclidean distance of 1 at least
             with one of the coordinates in
             the group. then
              Add the coordinate to the
               group.
            end
          end
        end
        if The group contains all of the faults'
         coordinates then
          Mark the MDC as one of the
           possible solutions.
        end
        else
          Remove the MDC from the checklist.
        end
      end
    end
  end
end
```

2) Randomly select multiple MEU shapes and their physical location inside the chosen memory. The shapes are selected with respect to the ones described in Fig. 8
3) Calls a C++ memory creator script that simulates the memory and outputs the logical faults addresses
4) Organize the outputs of the memory creator to simplify the checks of the memory reconstructor script

The previous steps were repeated to create a collection of memories and their related MEUs.

Now that the memories have been generated, a Python memory reconstructor script is called. This script performs the following operations:

1) Select a set of MEUs.
2) For each possible MDC and for each MEU in the set:
   a) Reconstruct all the physical locations of the faults inside the MEU, using their logical address and assumed parameters.
   b) Check the physical location to understand if all faults are adjacent to each other.
   c) If the faults are all adjacent, add the current MDC to the possible correct ones.
3) Collect all the MDCs that were marked as "possibly correct" for all the MEUs in the set.

The output of the aforementioned steps is a list of memory parameters and the number of MEUs they can contain in the correct neighborhood as previously explained in Fig.9 and Fig.10.

A. Memories' and MEUs' parameters

The experiments were performed on around 5,500 different bit-scrambled (as shown in Fig.6) memories with randomly chosen MDCs presented in table.I. A minimum is decided in tableI because the number of the equivalents increases in small-size memories. Because there are few MDCs available, and most of them are equivalent geometrically. These small-size memories were not having a dramatic reduction in possible MDC.

For each memory, a set of 100 MEUs were injected. These MEUs were randomly generated with the following MDCs, also reported in Fig.8:

- Base Fault Count: Between 2 and 6
- Shapes: square, rectangle, single line (wordline or bitline oriented)
- Extra fault: a small probability exists that an extra fault is added adjacent to a previously inserted fault, such as the case in the bottom right of Fig.8

Given the aforementioned parameters, the algorithm tested a total of 550,000 MEUs distributed in 5,500 memories.

Experimental results		
MEU sets per memory (#)	**Average equivalent MDCs (#)**	**Time**
20	6	26 minutes
40	3.64	45 minutes
60	2.79	63 minutes
80	2.31	77 minutes
100	2	96 minutes

TABLE II: Experimental results and performances of the proposed algorithm.

B. Stats and performances

The tests were performed on a Macbook Pro 13" equipped with an Apple M1 processor and 16GB of RAM. Table II summarizes our experimental results. It can be seen that each set of MEU had a different number of equivalent MDCs based on the number of considered MEUs (down to 2 for 100 MEUs). Notably, in the beginning, no data was available about the MDC. But among the candidate memories to be the correct memory with the correct parameters, the correct one is always present in this list. This cross-validation is based on the fact that the parameters were already known when random memories were generated.

C. Equivalent memories

For each MEU set, the algorithm correctly reconstructed the compliant memory parameters to what they belonged to, together with some "equivalent" memories. Fig. 12 and Fig.13 graphically explain why we have such equivalencies. For example, in Fig.12, it is possible to see how the same MEU made of four faults can come from memory with a straightforward organization (on top) or from memory with a Column mirror equal to 2 (on bottom). In Fig.13, another MEU of four faults can come from two different memories. One of the possible memories is mirrored in the center (top), and the other has a column mirror equal to 2 (bottom).

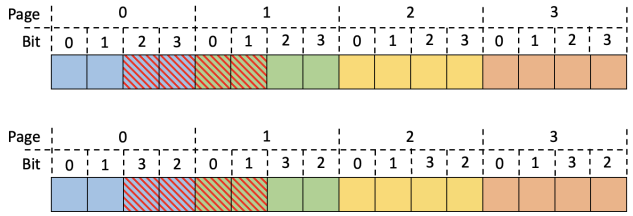

Fig. 12: The MEU in the picture may come from a straightforward memory organization (top) or a mirroring every two columns (bottom)

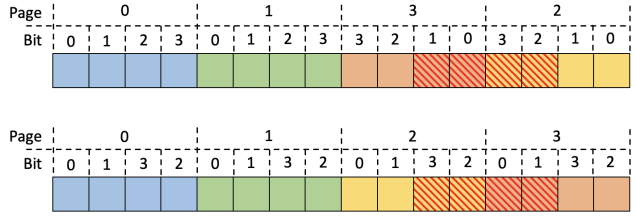

Fig. 13: The MEU in the picture may come from an x-axe mirrored memory organization (top) or a mirroring every two columns one (bottom)

The main reason behind these equivalencies is that the location of two faults can be changed without violating the fault location shapes provided in Fig.8.

These examples show some of the possible equivalent memories organizations. The number of equivalent memories can decrease if the MEU has a greater dimension and is not symmetrical. In symmetrical shapes, all the faults can be swapped without violating the memory contiguity, and there are always memories with parameters that fit these MEU shapes.

V. CONCLUSIONS

This paper presented an optimized algorithm for reconstructing the memory organization starting from MEUs from irradiation tests. MEUs were only reported with their logical addresses, as a standard user performing an irradiation test would receive them. The experiments illustrated in simulated memories with a set of injected MEUs show that the algorithm can always reconstruct the original memory organization and a certain number of equivalent memory organizations for that given MEU set. This reconstructed data can then be used to perform accurate and realistic MEUs injections to assess the reliability of safety-critical applications.

REFERENCES

[1] 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. S. Reorda, R. Ullmann, R. Vanhooren, N. Xama, and L. Wu, "Recent trends and perspectives on defect-oriented testing," in *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2022, pp. 1–10.

[2] C. D. Sio, S. Azimi, L. Sterpone, and D. M. Codinachs, "Analysis of proton-induced single event effect in the on-chip memory of embedded process," in *2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2022, pp. 1–6.

[3] C. De Sio, S. Azimi, A. Portaluri, and L. Sterpone, "Seu evaluation of hardened-by-replication software in risc- v soft processor," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2021, pp. 1–6.

[4] D. Chen, E. Wilcox, R. L. Ladbury, H. Kim, A. Phan, C. Seidleck, and K. A. LaBel, "Heavy ion irradiation fluence dependence for single-event upsets in a nand flash memory," *IEEE Transactions on Nuclear Science*, vol. 64, no. 1, pp. 332–337, 2017.

[5] N. Haddad, E. Chan, S. Doyle, A. Kelly, R. Lawrence, D. Lawson, D. Patel, and J. Ross, "The path and challenges to 90nm radiation hardened technology," in *2008 European Conference on Radiation and Its Effects on Components and Systems*, 2008, pp. 269–273.

[6] D. G. Mavis, P. H. Eaton, M. D. Sibley, R. C. Lacoe, E. J. Smith, and K. A. Avery, "Multiple bit upsets and error mitigation in ultra-deep submicron srams," *IEEE Transactions on Nuclear Science*, vol. 55, no. 6, pp. 3288–3294, 2008.

[7] A. D. Tipton, J. A. Pellish, R. A. Reed, R. D. Schrimpf, R. A. Weller, M. H. Mendenhall, B. Sierawski, A. K. Sutton, R. M. Diestelhorst, G. Espinel, J. D. Cressler, P. W. Marshall, and G. Vizkelethy, "Multiple-bit upset in 130 nm cmos technology," *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3259–3264, 2006.

[8] Y. He, Z. Lei, Z. Zhang, C. Peng, J. Li, E. Zhang, and Y. Yang, "Analysis of atmospheric neutron radiation effects in automotive electronics systems," in *2020 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, 2020, pp. 1–4.

[9] A. Williams, M. McEwen, and A. DuSautoy, "Radiation testing for space applications at the national physical laboratory," in *1998 IEEE Radiation Effects Data Workshop. NSREC 98. Workshop Record. Held in conjunction with IEEE Nuclear and Space Radiation Effects Conference (Cat. No.98TH8385)*, 1998, pp. 148–151.

[10] N. Campanelli, T. Kerekes, P. Bernardi, M. D. Carvalho, A. Panariti, M. S. Reorda, D. Appello, and M. Barone, "Cumulative embedded memory failure bitmap display & analysis," in *13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2010, pp. 1–6.

[11] L. M. Luza, A. Ruospo, D. Söderström, C. Cazzaniga, M. Kastriotou, E. Sanchez, A. Bosio, and L. Dilillo, "Emulating the effects of radiation-induced soft-errors for the reliability assessment of neural networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 4, pp. 1867–1882, 2022.

[12] L. D. Abbati, R. Ullmann, G. Paganini, M. Coppetta, L. Zaia, V. Huard, O. Montfort, R. Cantoro, G. Insinga, F. Venini, P. Calao, and P. Bernardi, "Industrial best practice: cases of study by automotive chip- makers," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2021, pp. 1–6.