SRAM Periphery Testing using the Cell-Aware Test Methodology

Xhesila Xhafa, Eric Faehn, Patrick Girard, Arnaud Virazel

SRAM Periphery Testing using the Cell-Aware Test Methodology

X. Xhafa[1], E. Faehn[2], P. Girard[1], A. Virazel[1]

[1]LIRMM - University of Montpellier/CNRS
Montpellier, France
xxhafa, girard, virazel@lirmm.fr

[2]STMicroelectronics
Crolles, France
eric.faehn@st.com

***Abstract*—Testing memory circuits is crucial for ensuring the quality and reliability of System-on-Chip (SoC) designs, especially as shrinking technology nodes increase susceptibility to nanometer-scale defects. This paper introduces an enhanced methodology for memory testing, leveraging the Cell-Aware (CA) test concept. Building on prior work for SRAM array testing [1], we extend the CA methodology to include periphery testing by generating, for the first time, CA models for each memory Input-Output (I/O) element, covering key components such as address decoders, write drivers, and sense amplifiers. We present results from testing these periphery components using the CA methodology. Additionally, we compare existing SRAM testing techniques with our CA methodology for the decoder and I/O circuitry. To ensure a fair comparison, we selected minimal March tests designed to detect functional faults in peripheral circuits, aligning with the fault models targeted by our approach. A quantitative analysis of fault coverage demonstrates the effectiveness of our methodology compared to March algorithms, particularly in terms of test complexity.**

***Index Terms*—Memory testing, Structural testing, Memory Periphery, Cell-Aware models, ATPG**

I. Introduction

In modern IC applications like Edge Computing, Machine Learning (ML), and the Internet of Things (IoT), there is a growing need to process and store large amounts of data. As a result, memory blocks now take up a substantial portion of System-on-Chip (SoC) areas [2]. At the same time, CMOS transistor technology has scaled down to nanometers, increasing memory density. However, this miniaturization makes memories more prone to defects and reliability issues [3], making memory testing crucial for ensuring IC quality and safety.

The most prominent memory testing approach is functional testing, which verifies the proper memory operation through a series of read and write operations. March tests are the most well-known algorithms developed to functionally test memory architectures [4], [5]. However, due to the high abstraction level of functional testing and the increasing complexity of circuit designs, including emerging memory technologies like MRAM and RRAM, functional testing is insufficient to ensure the optimal Defect Parts Per Million (DPPM) for these advanced memories [3], [6]. Hence, the development of more comprehensive and cost-effective testing methodologies, to ensure the reliability and performance of advanced memory technologies and modern ICs is imperative.

To address this issue, a paradigm shift from functional to structural testing by using the CA test methodology has been proposed in [1]. CA testing, originally developed for digital IC testing, addresses intra-cell defects in standard cells and has been proposed as a way to reduce test escapes that are not caused by defects on interconnects [7]. The CA methodology relies on the generation of CA models for each standard cell. CA models are dictionaries that associate intra-cell defects (specifically, transistor short and open defects) with the input cell patterns that detect them. The information in the CA models is then exploited by an Automatic Test Pattern Generator (ATPG) to generate test patterns that target this set of fault models. Although CA testing has been widely adopted in digital IC testing, its application on memory circuits remains novel.

The preliminary flow, presented in [1] for the adaptation of the CA methodology to memory testing has been applied to the SRAM bit-cell. In this paper, we address the limitations of the aforementioned flow and propose an improved version that is adapted to testing not only the SRAM memory array but also the memory periphery since defects on the latter have proven to have a significant impact on overall test escapes [5], [8]. Initially, the proposed flow includes the generation of CA models for the write driver, sense amplifier, and the standard cells that compose the address decoders. However, module-level simulations for defect detection prove to be insufficient due to the nature of fault propagation in a memory architecture. Hence, the UnDetected (UD) defects are further analyzed. The proposed solution involves the generation of custom CA models that contain additional information on the detection conditions of UD defects. To generate the custom CA models, the full memory architecture is used for simulations. However, the number of defects included in these CA models is significantly lower, since most of the defects are included in module-level CA models.

The CA testing methodology offers significant advantages by reducing the abstraction level of fault modeling from the functional down to the physical and transistor levels, which ensures the detection of intra-cell defects. One of the key

benefits of CA testing is its efficiency—SPICE simulations are only required once per standard cell (or memory block in memory testing contexts). These CA models can then be reused across multiple designs, significantly reducing time, resources, and testing costs. Furthermore, the CA methodology enhances defect diagnosis by identifying the precise location of defects, thereby accelerating physical failure analysis, which is essential for improving yield and addressing customer returns [9], [10].

It is essential to highlight the difference between fault types in traditional memory testing and those used by the ATPG tool in our methodology. Conventional FFMs represent functional deviations in memory behavior due to defects. These models guide the generation of March elements to detect specific faults. In contrast, our approach shifts from functional to structural fault modeling. Through ATPG, we introduce faults in the interconnections of memory modules, as well as those defined by the CA model tables, which directly represent defects simulated during CA model generation. This approach allows us to quantify fault coverage based on ATPG-generated patterns targeting both interconnection and CA model faults.

Through this methodology, we provide the following contributions:

- Firstly, transistor-level defects are systematically considered, and detailed information on defect location and detection condition is provided in the module-level CA model.
- Secondly, using the proposed flow, fewer SPICE simulations ought to be carried out at the memory level, hence reducing the test generation time.
- Thirdly, by structurally addressing each defect, the number of test patterns is optimized for each memory component, in contrast with well-known March tests that can often have high test complexity due to the linear correlation with memory size.

The paper is organized as follows. In Section II, a background on existing literature regarding SRAM periphery testing is given. In Section III, the flow of the CA methodology adaptation for SRAM periphery testing is detailed. In Section IV, the structural testing of address decoders is described. In Sections V and VI, the obtained results on write driver and sense amplifier testing are given, respectively. In Section VII, a comparison between existing solutions and our methodology is provided. Finally, conclusions and perspectives are drawn in Section VIII.

II. Background on SRAM Periphery Testing

A. Address Decoders

Address decoders are crucial for memory cell selection in read and write operations. Initially, researchers believed testing the memory array would also test the decoders, as defects would appear as faults in the array, with the MATS++ algorithm covering Address Decoder Faults (AFs), Stuck-At Faults (SAFs), and Transition Faults (TFs) in SRAM [11]. However, with the shift from NMOS to CMOS, this approach became inadequate, as CMOS decoder defects couldn't be detected by MATS++ [12]. To address these defects, researchers proposed decoder-specific patterns for March tests and new methods like combining Built-In Self Test (BIST) to detect Address Decoder Delay Faults (ADFs) [13] and modifying March C- for Address Decoder Open Faults (ADOFs) [14]. These solutions primarily rely on March tests, remaining in the realm of functional testing.

B. Write Driver

The Write Driver (WD) is essential for accurate data writing in memory systems, and defects, often modeled using resistors, can occur due to manufacturing issues. These defects are typically represented as static or dynamic functional fault models. Static faults in the memory periphery, covered by tests for memory cell array faults, are classified as memory cell array faults [15]. Dynamic faults, like resistive open defects, are modeled using the Slow Write Driver Fault or Un-Restored Write Fault models [16], and larger defects may cause the Un-Restored Destructive Write Fault (URDWF) model, affecting other cells in the same column. The March C- algorithm is commonly used for WD testing, but no structural approaches are documented.

C. Sense Amplifier

The Sense Amplifier (SA) in a memory architecture is an essential block. The ultimate purpose of a sense amplifier is to reduce the time and power consumption during the reading phase in a memory architecture. The correct functionality of the sense amplifier ensures the intended reading operation to be realized. Like in the case of the WD, defects in the SA can be caused during the manufacturing process. Several works in the literature have analyzed possible defects that can impair the functionality of the SA. However, the focus is kept on resistive opens, since they are known to cause dynamic faults with more complex detection conditions. Some of the FFMs that describe the faulty behavior of the SA are described in [17], [18]. Moreover, March tests (i.e., March C-) are used to test the aforementioned defects in the SA. Similarly to the WD, a structural approach to SA testing is lacking in the literature.

III. Cell-Aware Methodology for SRAM Periphery Testing

A. An Overview of the CA Methodology

The CA test methodology is originally developed for digital IC testing. As transistor size shrank to nanometer level, intra-cell defects (at transistor level) became more prominent. Hence a need for new fault models arose, apart from traditional ones (i.e., stuck-at fault, transition fault) that are injected in interconnections to represent inter-cell defects [7]. The CA methodology is a systematic way of representing defects inside standard cells that compose a digital IC. The basis of CA testing is the generation of Cell-Aware models. These models are files that contain information on the location of probable defects that can occur inside a given standard cell, as well

as the detection conditions (i.e., the input-output patterns that detect a defect).

The generation of CA models for each standard cell involves multiple steps. First, all possible defects within each standard cell, known as intra-cell defects, are identified. These defects include transistor-level faults such as shorts and opens, as well as layout-based defects arising from the physical circuit layout. By modifying Design Rule Check (DRC) parameters in existing DRC tools, adjacent polygons that are likely to cause defects are identified, and the defects are modeled as shorts (a low-value resistor) between the affected nets, whereas locations with thin polygons can be modeled as opens (a high-value resistor) [19]. Each identified defect is then injected into the circuit, and SPICE simulations are run. If a deviation from the expected behavior of the standard cell is observed in the output signals, the injected defect is marked as detected and associated with the corresponding input pattern.

At the end of the analog simulations, a dictionary mapping all input patterns to the detected defects is generated. This dictionary, presented in a table, along with the location and size information for each injected defect, is compiled into a file known as the CA model. The CA model includes both statically detected defects (requiring only one pattern for detection) and dynamically detected defects (requiring two patterns). These CA models are then used by an Automatic Test Pattern Generator, along with the gate-level digital IC description (Verilog), to create circuit-level patterns that test the overall IC performance for inter-cell and intra-cell defects.

B. CA Methodology Adaptation to SRAM Periphery Testing

Prior work in adapting the CA methodology to SRAM testing is given in [1]. However, the work presented in [1] only addresses the SRAM bit-cell testing, providing a limited solution for holistic memory testing. In this section, an improved flow that addresses defects not only in the memory array but also in the periphery is described.

The flow, shown in Fig. 1, begins with the hierarchical spice netlist of a 4x4 memory architecture designed in 28nm FDSOI, used as a case study. The architecture includes the memory array, address decoders, write driver, sense amplifier, and pre-charge circuits and it is shown in Fig. 2 (a). The fault-free simulation for a sequence of Write (W) and Read (R) operations (i.e., W1R1W0R0) is given in Fig. 2 (b). The first step in applying the CA methodology to memory testing is to convert the SPICE netlist into a Verilog one, such that gate-level memory models (resembling the standard cells in the case of digital ICs) are obtained. A functional verification tool, namely ESP from Synospsys, which is commonly used to perform equivalence checking between SPICE and Verilog netlists, has been employed for this step. The described 4x4 SRAM memory consists of combinational and analog circuit blocks. Combinational blocks like the address decoders and WD are straightforwardly represented at the gate level. However, additional steps need to be taken for analog blocks like the SA. The gate-level netlist is then simulated using a Verilog simulator to verify correct behavior. The main

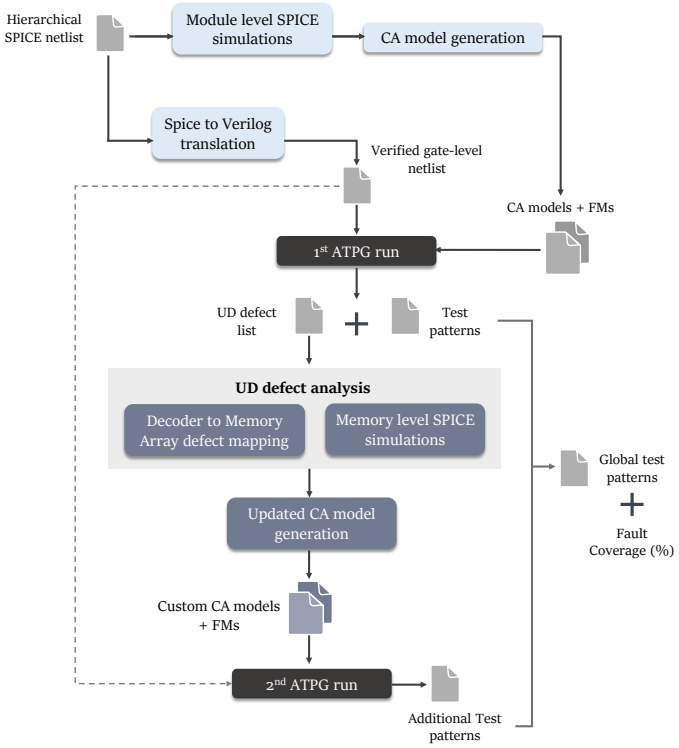


Fig. 1: SRAM periphery testing flow using the CA methodology

difference between SPICE and gate-level simulations is the analog behavior of the Bit-Line (BL) and Bit-Line Bar (BLB) signals, which discharge over time in analog simulations. However, the final signal levels match in both SPICE and Verilog simulations. This gate-level description methodology is not limited to the SRAM architecture; it is technology and size-independent, making it applicable to larger CMOS-based SRAM architectures and other memory types like MRAM that use transistor-based periphery blocks.

Since the objective is to use an ATPG for the generation of test patterns, the netlist needs to be adapted such that it is suited for a combinational ATPG. However, a combinational ATPG does not allow the bidirectional nature of the memory. Hence, the netlist is separated such that the write and read operations are done in different time sequences. In the architecture used for write operations, the content of each cell is considered as an output. In the read architecture, the inner storing nodes are considered as inputs to represent the Stored Bits STB.

In parallel, the process of CA model generation for each memory block is realized. The first step is to identify all the possible defects that can occur in each sub-circuit (intra-cell defects). In CA models, intra-cell defects are represented by transistor-level defects and layout-based defects. Transistor-level defects include shorts and opens. Layout-based intra-cell defects depend on the physical layout of a circuit. By modifying the DRC rules, adjacent polygons that can cause potential defects are identified. Consequently, the intra-cell

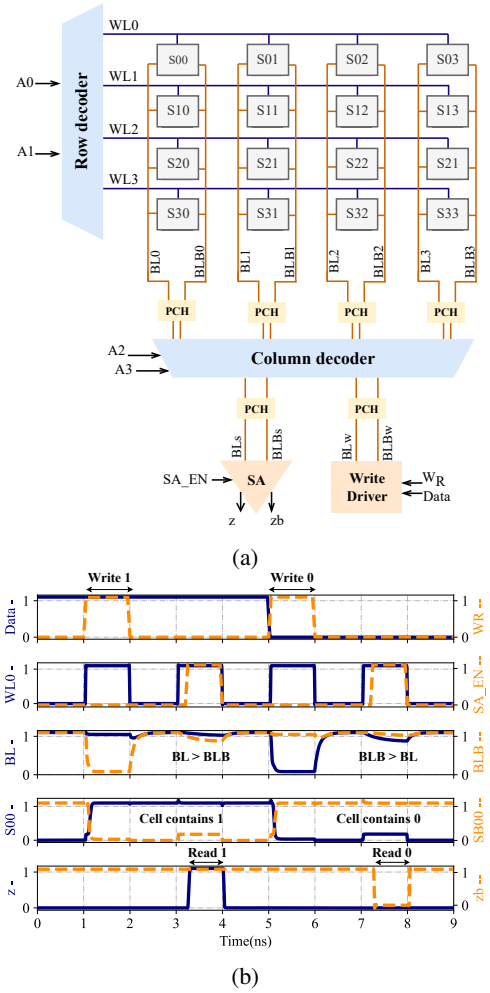


Fig. 2: (a) The 4x4 SRAM memory and (b) its fault-free simulation for a W1R1W0R0 sequence of operations.

defect list is enriched. Each defect is then injected into the circuit one by one, and electrical simulations are run using HSPICE. If a deviation in the output signals is observed due to the presence of the injected defect, the defect is mapped as detected by the corresponding input pattern. At the end of the analog simulations, a dictionary that maps all input patterns to the detected defects is generated. The CA model includes two tables; the static table where a single pattern suffices to detect the defect and the dynamic one where two patterns are necessary to detect the defect.

Next, the generated CA models are used by an ATPG, in conjunction with the circuit description at the gate level (Verilog description), to generate external patterns that test the overall performance of the memory in the presence of intra-cell and inter-cell defects. The ATPG from TMAX from Synopsys has been used for this purpose [20]. Inter-cell defects are located in the interconnections of each memory gate-level block, while intra-cell defects are the cell-aware defects identified during the CA model generation step. The fault models considered are Stuck-At-Fault (SAF) for statically-detected defects and Transition Fault (TF) for dynamically-detected defects.

The ATPG generates a list of test patterns and a list of UnDetected (UD) defects. The next step of the flow is to analyze the UD defects. For blocks such as the WD and SA, the UD defects are injected into the corresponding blocks, and memory-level SPICE simulations are performed to identify the dynamic operations needed to detect them. Several studies indicate that dynamic operations accessing more than one bit-cell (i.e., dynamic 2-cell faults) are necessary for detecting specific defects [16], [17], [21] in the memory periphery. Such operations cannot be covered by module-level simulations alone, resulting in UD defects after the first ATPG run. To address this issue, the effects of UD defects are mapped into the memory array through custom CA models generated for bit-cells of the same column controlled by the same WD and SA.

Like in the case of the WD and SA, detecting the UD defects in the decoder requires dynamic patterns involving more than one cell in the memory array. However, for address decoders, it is possible to deduce the necessary address order and operation using the information in the standard cells CA model that compose the decoders, with no need for memory level simulations. Since the row decoder controls the activation of Word Lines (WLs), only one column of the memory array is necessary for this task. To allow the ATPG to understand the switching of WLs that enables addressing two cells, a new CA model is generated for the first column (CLM0) of the array. This emulates the detection conditions so the ATPG can generate structural patterns. This step is detailed further in Section IV. After generating the custom CA models, another ATPG run is performed to obtain the additional patterns. The process concludes by merging all test patterns and calculating the fault coverage. In the ATPG tool, intra-cell defects are treated as faults, so the tool computes *fault coverage* as the ratio of detected faults to the total number of considered faults, including both inter-cell and intra-cell faults.

It is important to note that the CA methodology targets defects at the module level, which ensures that scalability remains unaffected. The flow presented in Figure 1 is applicable to SRAMs of varying sizes and types, as the fundamental building blocks of the memory remain the same. While the CA model generation process can be repeated for different designs and technology nodes, this is a one-time procedure. Once generated, the CA model for each module can be reused as needed. Although SRAM arrays often constitute up to 80% of the memory, the CA model for the SRAM bit-cell is created once and reused to address all potential defects in each of the bit-cells. The same principle applies to address decoders, which are composed of reusable cells throughout the design.

Moreover, the CA methodology can be extended to other memory technologies beyond SRAM. Since most memory architectures share common blocks such as decoders, sense amplifiers, write drivers, and bit-cells, the presented flow in Figure 1 can be adapted for emerging memory technologies, especially for the periphery circuitry. The primary difference

lies in the CA model generation process for the bit-cell itself. For instance, MRAM and RRAM bit-cells differ from the transistor-based SRAM bitcell, requiring adjustments in defect modeling to account for technology-specific characteristics.

IV. Structural Testing of Address Decoders

To apply our structural testing methodology to address decoders in SRAMs, we have considered a 2-bit NOR-based row decoder, shown in Fig. 3. This decoder is part of the 4x4 SRAM memory architecture that has been used as a case study.

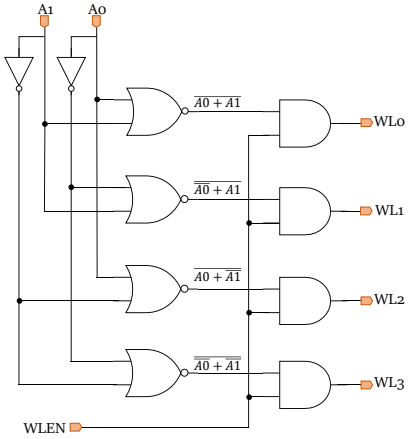


Fig. 3: Row address decoder structure

A. CA Model Generation for Row Decoder Cells

CA models need to be generated for each module type in the decoder. For this design, we utilize standard cells from the 28nm technology library from STMicroelectronics. Specifically, CA models are generated for NOR and AND gates, following the steps described in the CA model generation process (see Section III-A). These CA model files contain detailed information on defect types and their locations within the cell, along with detection tables that include the necessary patterns to detect these defects. Defects with similar detection conditions are classified as equivalent. During the CA model generation, non-resistive open and short defects are considered, although resistive defects can also be addressed by CA tests [22]. Each defect is individually injected and SPICE simulations are run.

Defects are classified and organized into static and dynamic tables based on their detection conditions. The location and type of defects listed in the tables are depicted in Fig. 4 and 5. Tables I and II show the static detection tables for NOR and AND gates. In these tables, a '1' under a defect name indicates that the pattern of the same row detects the defect, while a '0' means that it does not. It is important to note that in the static table, each pattern corresponds to a maximum of one stimulus.

The dynamic tables of the NOR and AND CA models are shown in Tables III and IV, respectively. In these tables, the 'R' notation indicates a rising transition from logic '0' to logic '1'. Conversely, the 'F' notation indicates a falling transition

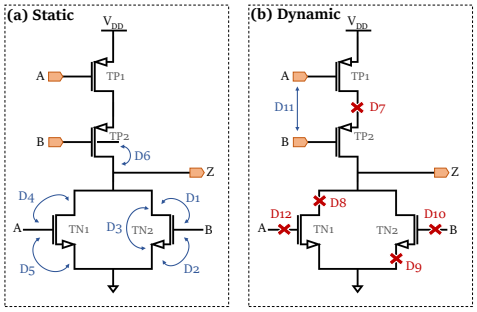


Fig. 4: The NOR schematic for defects in (a) static CA model table (b) dynamic CA model table

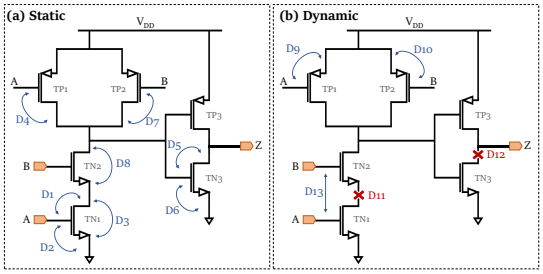


Fig. 5: The AND schematic for defects in (a) static CA model table (b) dynamic CA model table

from logic '1' to '0'. To better understand Table III we can analyze pattern 1. The '0R-F' input-output pattern can detect defects D9 to D11. This means that a logic '0' on input A and a rising transition on input B of the NOR gate can cause the output to deviate from its golden value, in the presence of D9, D10, or D11.

B. Decoder to Memory Array Defect Mapping

Given the simple decoder architecture as a logic circuit, if the ATPG is run only on the row decoder, achieving a 100% fault coverage is straightforward. However, in the context of a 4x4 memory architecture, the propagation of each inter-cell and intra-cell defect from the decoder's defective nets to the

	Inputs		Output	Defects					
	A	**B**	**Z**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
Pt.1	0	0	1	1	0	1	1	0	0
Pt.2	0	1	0	1	1	0	0	0	1
Pt.3	1	0	0	0	0	0	1	1	1
Pt.4	1	1	0	1	0	0	1	0	1

TABLE I: Static CA model table for defects in the NOR gate

	Inputs		Output	Defects							
	A	**B**	**Z**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**
Pt.1	0	0	0	0	0	0	1	1	1	1	0
Pt.2	0	1	0	1	0	1	1	1	1	0	0
Pt.3	1	0	0	0	0	0	0	1	1	1	1
Pt.4	1	1	1	1	1	0	1	1	0	1	0

TABLE II: Static CA model table for defects in the AND gate

	Inputs		Output	Defects					
	A	**B**	**Z**	**D7**	**D8**	**D9**	**D10**	**D11**	**D12**
Pt.1	0	R	F	0	0	1	1	1	0
Pt.2	R	0	F	0	1	0	0	1	1
Pt.3	0	F	R	1	1	0	0	0	0

TABLE III: Dynamic CA model table for defects in the NOR gate

	Inputs		Output	Defects				
	A	**B**	**Z**	**D9**	**D10**	**D11**	**D12**	**D13**
Pt.1	1	R	R	0	0	1	0	0
Pt.2	F	1	F	1	0	0	1	1
Pt.3	1	F	F	0	1	0	1	1

TABLE IV: Dynamic CA model table for defects in the AND gate

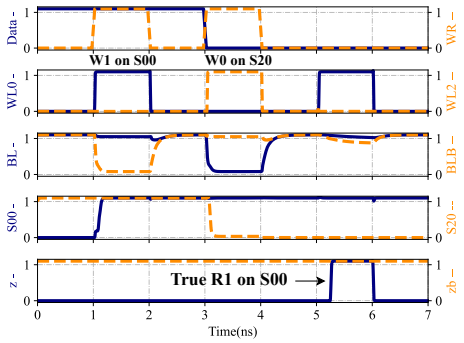
memory's output (i.e., the Sense Amplifier) is required. This propagation can be hindered by other memory blocks, making some defects, particularly open defects in the NOR gates, undetectable using only detection tables from module-level CA modules. For instance, a defect causing a falling transition fault on a Word Line (WL) might be blocked from reaching an observable output (z and zb nodes in Fig 2), preventing defect propagation.

To illustrate this, consider defect D10, an open defect at the gate terminal of transistor TN2 in the NOR gate (Fig. 4). Suppose D10 is in the NOR gate controlling the WL0 signal (Fig. 7). SPICE simulations shown in Figure 6 indicate that the following detection conditions are necessary:

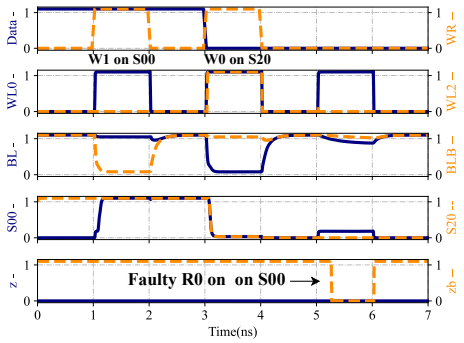
- W0 at address <A0, A1> = <0, 0>: WL0 is active
- W1 at address <0, 1>: WL2 is active. Due to a falling transition delay caused by D3, WL0 remains active during this W1 operation.
- Read at address <0, 0>: '0' is expected, but '1' is read.

Structurally, as shown in Table III, defect D10 is detected only by Pt.1, requiring a level '0' on input A and a rising transition 'R' on input B of the NOR gate, causing a delay in the output Z's falling transition. The inputs (A, B) of the defective NOR1 gate map to the row decoder inputs <A0, A1>. Thus, to detect D10, the decoder address must switch from <0, 0> to <0, 1>. In this memory architecture, the falling transition delay propagates to WL0 when the WL ENable signal (WLEN) is active. However, the falling transition of WL0 (selecting S0) is blocked by the access transistor primitive in the 6T SRAM bit-cell, preventing defect propagation (see Fig. 7).

To account for the impact of UD defects on the memory array, these defects can be incorporated into a new CA model that allows for the control of multiple WLs. Since the row decoder is responsible for activating the WLs, considering just one column of the memory array for the CA model is sufficient. Thus, the CLM0 CA model is created. It is important to note that, apart from verification purposes (shown for D10), SPICE simulations were not used for all defects to generate this new CA model. Instead, the mapping of



(a)



(b)

Fig. 6: Fault-free behaviour (a) for operations W1(S00)W0(S20)R1(S00) and faulty behaviour (b) due to D10 in NOR1

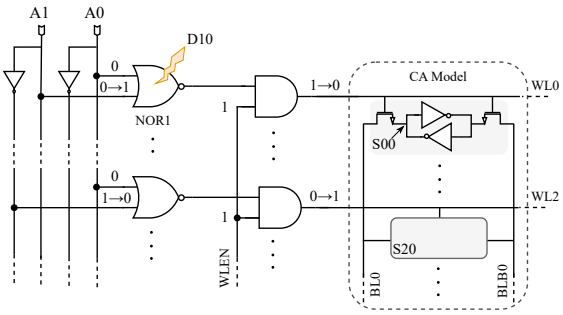


Fig. 7: Propagation of D10

the targeted defects was done only based on the information provided in the CA models of the NOR gates.

The mapping process is illustrated in Table V for the dynamic patterns in the NOR gate controlling WL0. Given that the inputs of the NOR gates are controlled by the row address input signals, we can deduce the necessary address switching for the defect to become apparent. This address switching indicates which WL should be deactivated and which should be activated. Additionally, for the defect to be observable, the data content must switch, so the bit lines are included in the CLM0 CA model as inputs with rising and falling transitions (the choice of transition is arbitrary).

NOR1 Input		→	CLM0 CA model Inputs					
A0	**A1**	**Address Order**	**WL0**	**WL1**	**WL2**	**WL3**	**BL0**	**BLB0**
0	R	<0,0> to <0,1>	F	0	R	0	R	F
R	0	<0,0> to <1,0>	F	R	0	0	R	F
0	F	<0,1> to <0,0>	R	0	F	0	R	F

TABLE V: Address switching order to map defects from decoder to memory array

An example of the CLM0 CA model dynamic table is shown in Table VI. This table displays the pattern that detects the previously analyzed D10 defect in NOR1. Since the defect is manifested by two consecutive write operations, the write architecture is utilized, and the observable outputs are the inner nodes of the bit-cells in CLM0 that contain the written value. This pattern detects all the defects identified by Pt.1 in the NOR dynamic table. The same process is applied to all UD defects in all the gates of the row decoder.

Inputs						Outputs				Defects
WL0	**WL1**	**WL2**	**WL3**	**BL0**	**BLB0**	**S00**	**S10**	**S20**	**S30**	**D9,D10,D11**
F	0	R	0	R	F	0	0	R	0	1

TABLE VI: Dynamic table of the CLM0 CA model

C. ATPG Results for Row Decoder

The obtained CA models are used by an ATPG to generate test patterns that detect defects in interconnections, as well as static and dynamic CA model defects in the row decoder of a 4x4 SRAM architecture.

1) Static Faults

The SAF detection process in the decoder involves three phases. Initially, only the SAFs on the interconnections are targeted, and ATPG generates the necessary circuit-level patterns for their detection. This set of patterns is saved for later use. Afterward, Static Cell-Aware (SCA) defects from the CA models are added, and a fault simulation is run using the patterns from the first phase to check coverage. If the coverage is below 100%, a second ATPG run generates additional patterns for the undetected intra-cell defects. A total of 112 inter and intra-cell defects are considered, resulting in 12 patterns, 8 for W1R1 operations on CLM0 and 4 for detecting D8 in the AND standard cell, which causes a Stuck-At-1 Fault (SAF1) on each WL. The ATPG generates four read patterns while the decoder is inactive, labeled as R_EN0.

After the two ATPG runs, fault coverage reaches 96%. However, four defects (D3 in each AND gate) remain undetected. D3 is detected only with a '01' input, meaning the AND gate's output for the corresponding WL remains inactive. This situation is similar to open defects, where activating another WL is necessary to propagate the defect. To address this, D3 is mapped to the CLM0 CA model, following the same procedure as for open defects. This final step raises the SAF fault coverage to 100%.

2) Dynamic Faults

The pattern generation process for TFs and defects in the Dynamic Cell-Aware (DCA) detection table follows a similar approach to SAFs. For the TF model, 92 inter and intra-cell defects are considered. Initially, 4 static patterns target inter-cell defects by writing values into each CLM0 cell, producing 14 transition patterns (28 stimuli) in the first ATPG run. Figure 8 (a) shows these patterns and corresponding memory operations in terms of row decoder inputs <A0, A1>. After adding intra-cell defects from the Dynamic Cell-Aware (DCA) table, a fault simulation is run using these patterns. No additional patterns are generated in the second ATPG run, achieving 60% defect coverage.

As discussed in Section IV-B, some NOR intra-cell defects causing falling TFs at the WL-s remain undetected. These defects are mapped to the CLM0 CA model, and following the procedure in Section IV-B, a second ATPG run increases defect coverage to 100%. In Fig. 8 (b), the lined arrows represent W0W1 operations between addresses (e.g., '00' to '01'), while the curved lines show the read operation at the initial address (e.g., '00'). This step requires 20 additional stimuli, bringing the total to 52 stimuli for full TF detection. The cumulative defect coverage is shown after each ATPG run in Fig. 8, and the transition patterns cover the entire graph.

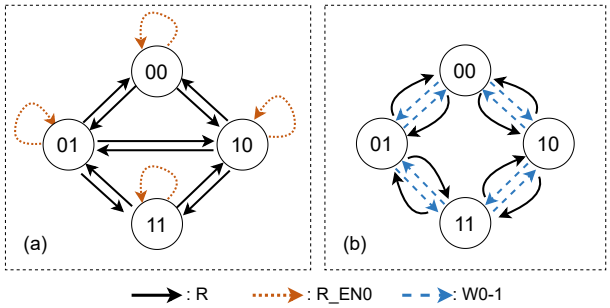


Fig. 8: Address orders of TF detection patterns for each ATPG run, where in (a) DC = 60%, (b) DC = 100%.

V. Structural Testing of the Write Driver

As previously mentioned in Section II, no structural approaches have been explored on WDs, so far. In this section we demonstrate how the CA methodology can be used for WD testing. Firstly, we show the generation of CA models for each component of the WD. Then, the ATPG generated structural patterns necessary for testing the SAF, TF and defects in CA models are given. Moreover, we show the defect mapping process for UD defects to a custom CA model to obtain maximum fault coverage.

A. Write Driver Topology and its CA Model Generation

The WD is composed of two parts, as shown in Fig 9. The first part (a) is called the WD controller and the purpose of this block is to control the Data and WD Enable (WR) signals. The outputs of the controller are the W0 and W1 signals that act as inputs for the WD core. When W1 is high (W0 is low), the desired data to be written is '1', hence the WD core maintains BL to digital level '1' and pulls down BLB at digital level '0'. The inverse happens when the data

input is '0'. The controller part of the WD is composed of NOR and NOT standard cells. The WD core needs to be treated as a separate memory module, the CA model of which has to be generated. The defects in the controller consist of defects included in the NOR2 CA model, which is previously explained. Note that this demonstrates another utility of the CA model methodology since for several blocks that are used several times in the design, the SPICE simulations do not need to be repeated. Moreover, the defects in the NOT gate are only modeled by SAF or TFs.

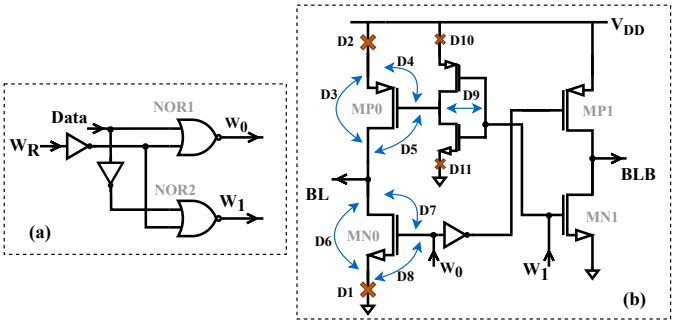


Fig. 9: Write Driver Controller (a) and Core (b) and considered defects

The considered defects in the WD core are shown in Fig. 9 (b) only for one side of the WD core since the design is symmetrical. Hence, the detection conditions are inversed for each symmetrical defect. Following the CA model generation method, each defect is injected and the defective WD core is simulated in SPICE. The results are compared with the golden behavior of the WD core. Despite considering only defects in the WD core, the simulated circuit also includes the Pre-Charge (PCH) for pre-charging the WD output bit-lines to V_{DD}. This inclusion is motivated by Fig. 10, where defect D2, an open defect in the source of transistor MP0, delays the charging operation of BL during a write operation. With this defect, the pull-up of node BL cannot occur, resulting in Fig. 10 (c), where BL remains at digital level '0' and a W1 operation fails. However, the bit-line outputs from the WD are connected to a PCH circuit. Thus, simulating the WD core with the PCH ensures the pull-up is enacted regardless of the defect, preventing any faulty behavior, as shown in Fig. 10 (d). Consequently, D2 is not included in the CA model table, presented in Table VII.

An example of the effect of one of the short defects (D3) in the WD core is given in Fig. 11. Due to this defect, the BL output of the WD is connected to V_{DD} and cannot be discharged. This means that a W0 operation cannot be acted on the selected cell. The defect acts as a static fault in the WD core. However, due to the nature of the memory, we have to take into consideration the initial value of the selected bit-cell. To ensure that we detect the lack of transition between two written data values the sensitizing operation W0 has to be preceded by a W1 operation. Hence, despite the static behavior, all the defects in the WD core are mapped in the

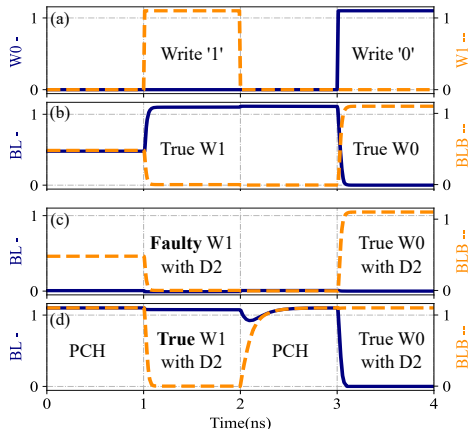


Fig. 10: (b) Fault-free behavior of the WD core without PCH and behavior of the WD due to D2 injection (c) without PCH and (d) with PCH.

dynamic table since two operations are necessary to ensure their detection.

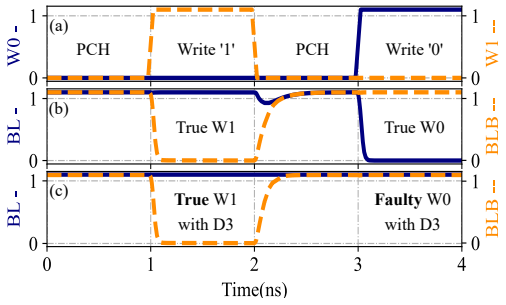


Fig. 11: (b) Fault-free behaviour of the WD core without D3 and (c) faulty behaviour due to D3

In the dynamic CA model table shown in Table VII, defects under which a '1' is denoted, manifest a faulty behavior on the first output (BL). A '2' is denoted when the effect of the defect can be seen on the second output (BLB).

	Inputs		Outputs		Defects						
	W0	**W1**	**BL**	**BLB**	**D1**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**
Pt.1	R	F	F	R	1	1	1	1	0	0	1
Pt.2	F	R	R	F	0	0	0	0	2	2	0

TABLE VII: Dynamic table of WD core CA model

B. ATPG Results for Write Driver Testing

The next step is to use the ATPG to generate memory-level patterns that detect the defects located in the WD structure. SAFs, TFs, and CA model defects are considered. Static faults in the WD include the SAF on interconnections between each sub-module in the WD and defects in the static tables of the NOR CA models. A total of 38 defects is injected and the first ATPG is run. The generated patterns correspond to static W0 and W1 operations. However, as previously discussed, we

have to make sure that the selected bit-cell contains the inverse value of the desired write operation. Hence, since the static write operations are included in the dynamic ones, we consider the resulting patterns for dynamic faults to be sufficient for static ones as well.

From the first ATPG run for dynamic faults (TF and defects in dynamic CA model table), the resulting patterns are shown in Table VIII. The inputs of the memory level circuit are the 4 bits of address decoders (the first two bits select the WLs; the last two bits select the columns), the WR: WD enable signal, the Data, the CEq: signal that activates the pre-charge circuit (PMOS based), and WLEN: the decoder Enable signal. The outputs of the memory are the (storing nodes; See Fig. 7) S nodes of each bit-cell that are observable points for the write architecture.

	Inputs					Outputs			Sensitizing
	Address	**WR**	**Data**	**CEq**	**WLEN**	**S00**	**S01**	**S..**	**Operations**
P1/0	0000	1	0	1	1				W0W1(R1)
P1/1	0000	1	1	1	1	1	X	..	
P2/0	0000	1	1	1	1				W1W0(R0)
P2/1	0000	1	0	1	1	0	X	..	

TABLE VIII: Generated patterns from the 1st ATPG run

From the results, we notice that the ATPG selects one cell, the output of which is S00, and the patterns correspond to W0W1 and W1W0, both followed by a read operation. Note that between each write operation, a PCH operation is also carried. The obtained fault coverage is 84% after the first ATPG run. To improve the fault coverage, the UD defects are simulated in SPICE at memory-level. An example of one of the simulated UD defects corresponds to D9 in Fig. 4, located in NOR1 of the WD controller. This defect hinders the output of NOR1 from discharging hence the WD is kept ON and continuously writes '0'. After running the SPICE simulation we conclude that the detection conditions for this defect are as follows:

- W0 in address (A0, A1) = (0, 0)
- Read in address (1, 0) where 1 is previously stored: '1' is expected yet '0' is read.

The effect of D9 can be seen in Fig. 12. On the left side of the figure, the correct behavior of the memory is shown for operations W0 in cell S00 and R1 in cell S10. On the right side, D9 in NOR1 is injected and the same simulation is repeated. We observe that due to D9 the content of the cell S10 changes right before the read operation is realized since BL is already discharged. Hence, the read operation is destructive to the cell content and the wrong value is read. For this type of defect a dynamic pattern in which a read is immediately acted after a write is necessary. Since the defect is manifested and detected during a read operation, the necessary patterns need to be generated using the read architecture which has been previously explained in Section III-B. All UD defects are of the same nature, hence they are mapped in the custom CA model used to detect dynamic 2-cell faults.

In Table IX, the inputs of the custom CA model are the stored bit values of cells S00 and S10, as well as word lines WL0 and WL1. The outputs are the bit lines of the first column. The ensure the detection, the stored bit values in the two cells are complementary values. To ensure the 2-cell access the word lines WL0 and WL1 need to be falling and rising. The outputs BL and BLB rise and fall according to the cell contents. D9 has a similar effect if located on NOR2, hence a second pattern has been included in the table, to account for the inverse scenario.

In Table IX, the custom CA model receives inputs from the stored bit values of cells S00 and S10, along with word lines WL0 and WL1. The outputs are the bit lines of the first column. To ensure detection, the stored bit values in the two cells must be complementary. For 2-cell access, word lines WL0 and WL1 need to transition, with WL0 falling and WL1 rising. The outputs BL and BLB signals rise and fall according to the cell contents being read. A similar effect occurs with D9 if placed on NOR2, where due to the defect the WD is kept ON and continuously writes '1'. Hence a second pattern is included in the table to represent the inverse scenario. The number three in Table IX represents the fact that the effect of the defect is observed on both outputs, since the outputs BL0 and BLB0 are complementary.

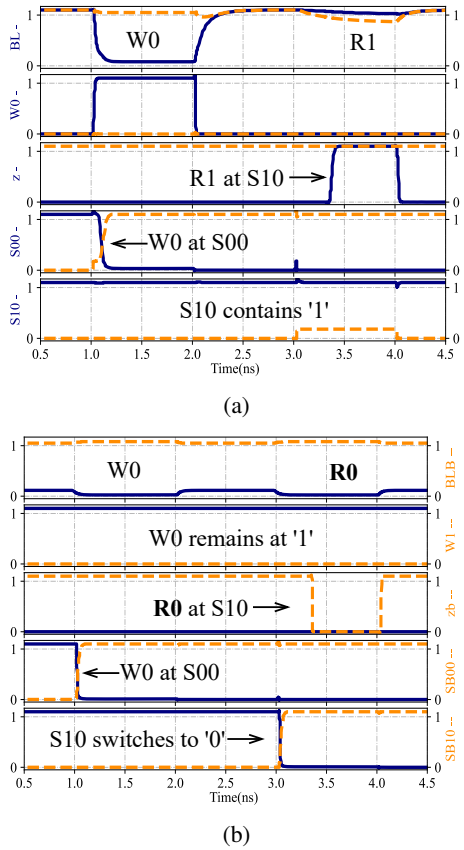


Fig. 12: Fault-free behaviour (a) and faulty behaviour due to D9 in WD (b) for operations W0(S00)R1(S10)

The generated patterns are shown in Table X. These patterns ensure the address switching to access two different cells

	Inputs				Output		Defects	
	S00	**S10**	**WL0**	**WL1**	**BL0**	**BLB0**	**D9 (NOR1)**	**D9 (NOR2)**
Pt.1	0	1	F	R	R	F	3	0
Pt.2	1	0	F	R	F	R	0	3

TABLE IX: Custom CA model for 2-cell dynamic faults in WD

containing complementary values. The selection of the two cells is arbitrary, provided they are in the same column. The fault coverage after the second ATPG run reaches 100%. Hence, a total of 4 dynamic patterns is necessary for the complete WD testing.

	Inputs							Outputs	
	Address	**WLEN**	**CEq**	**SA_EN**	**S00**	**S10**	**S..**	**z**	**zb**
P1/0	0000	1	1	X	0	1	X		
P1/1	0100	1	1	1	0	1	X	1	1
P2/0	0000	1	1	X	1	0	X		
P2/1	0100	1	1	1	1	0	X	0	0

TABLE X: Additional patterns generated from the 2nd ATPG run for WD testing

VI. Structural Testing of the Sense Amplifier

In this section, we demonstrate the application of the CA methodology for SA testing. First, we present the generation of CA models for the SA block. Next, we provide the ATPG-generated structural patterns necessary for testing SAF and TF in interconnections, as well as, defects in the CA model. Additionally, we describe the defect mapping process for UD defects to a custom CA model to achieve maximum fault coverage.

A. Sense Amplifier Topology and its CA Model Generation

To understand how manufacturing defects can alter the operation of the sense amplifier, it is first important to understand its operation and functionality. The ultimate purpose of a sense amplifier is to reduce the time and power consumption during the reading phase in a memory architecture. In between write and read operations, the bit-lines (BL and BLB) are charged to a specific voltage (in our work, at V_{DD}). The operation of the sense amplifier is enabled slightly after the pre-charge phase has ended. This delay allows one of the bit-lines to discharge according to the content of the accessed cell. If the cell contains a '1', the BL will remain at V_{DD}, whilst the BLB starts to discharge. If the cell contains a '0', BL discharges, while BLB remains at V_{DD}. When the SA is activated, the small difference between the bit-lines is detected, and given that the SA operates correctly, the outputs of the SA will be amplified to the expected bit values stored in the bit-cell.

The chosen structural topology for the SA is shown in Fig. 13. Depending on design specifications, different SA topologies can be used for the I/O circuitry of an SRAM [23], [24]. The design presented here is a current-based SA. Apart from the benefits regarding its low leakage power and high speed for high-performance applications, the choice for the design is intentional since it is important in the context of CA model generation to have distinguished inputs and outputs for each block. In this design, the bit-lines that serve as differential inputs to the amplifier, are separated from the amplified SA outputs. The expected behavior of the SA at memory-level has been previously shown in Fig. 2. In a fault-free simulation, when SA is enabled its outputs converge at either '1' or '0' depending on the stored data value on the selected cell.

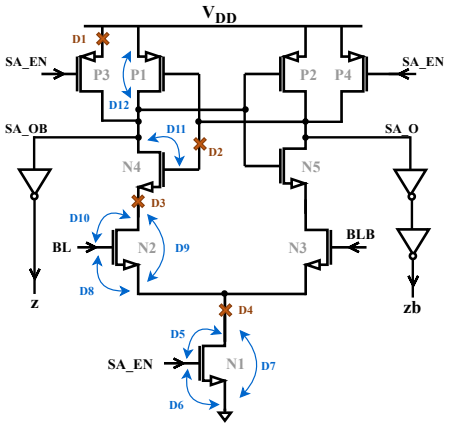


Fig. 13: Sense Amplifier and a subset of the injected defects

The generation of the CA model for the SA is done similarly to the WD core. The module-level simulations are run for each injected defect and the simulation results are compared with the expected outputs of the SA. Defects shown in Fig. 13 are a subset of all the injected defects. Equivalent and symmetrical defects are not included. During SPICE simulations, several types of defect behavior are observed, the examples of which are given in Fig. 14. The inputs of the module are the SA Enable signal (SA_EN) and the bit-lines. The expected output of the SA for operations R1R0 is given in Fig. 14 (a). In PCH operations are carried out between each read. Figure 14 (b) shows the output of the sense amplifier when defect D3 is injected. Due to this defect, the SA_OB output of the SA is blocked from being pulled down. Hence, the output z is stuck at '0'. This prevents the SA from correctly reading a cell containing '1'. For this reason, the defect is mapped as detected by Pt.1 of the static table of the CA model of SA, given in Table XI. A similar static fault behavior is observed for defects D1, D2, D8, D9, D10, and D12. Each of them is detected by a single R0 or R1 operation.

Notice that in Table XI, BL and BLB values are digital, whereas the signals BL and BLB shown in Fig. 14 are analog. As previously discussed in Section III-B, one of the main differences between the SPICE and Verilog representations of the SRAM architecture is the lack of analog behavior of the bit-lines that cannot discharge in time in the digital domain, but achieve their final value without the need of a SA. This means that the SA in the digital representation acts purely as a buffer between its inputs and outputs, given that the SA is enabled. Despite losing its functionality in the digital representation, the defect injection and SPICE simulations are done in the analog

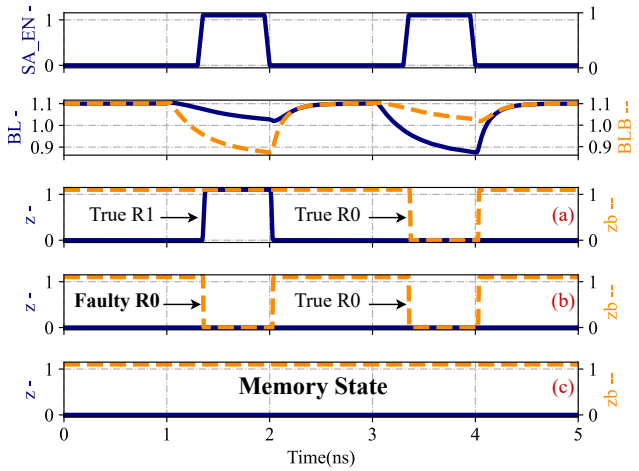


Fig. 14: SA module simulations for R1R0 operations (a) Fault-free, (b) Faulty due to D3, and (c) Faulty due to D6

domain, hence we can correctly map the effect of defects in the SA to the CA model.

	Inputs			Outputs		Defects						
	BLs	**BLBs**	**SA_EN**	**z**	**zb**	**D1**	**D2**	**D3**	**D8**	**D9**	**D10**	**D12**
Pt.1	0	1	1	0	0	3	3	0	3	3	0	0
Pt.2	1	0	1	1	1	0	0	3	0	0	3	3

TABLE XI: Static table of CA model of SA

Another type of defect in the SA is one that causes the outputs of the sense amplifier to remain in a memory state. An example of such a defect is D6, a short between the SA_EN and ground, as illustrated in Fig. 14. Due to this defect, the tail transistor N1 remains OFF, preventing the SA from being enabled. Consequently, the outputs of the SA do not converge to a single value. This scenario is interpreted by the data output circuitry as a memory state, in which the previous data read is stored.

To detect such a defect, a dynamic pattern is necessary. To ensure that a different value from the previously stored one is read, two read operations in two distinct cells containing complementary values need to be performed. Hence, this type of defect can be mapped to the previously explained custom CA model used to detect dynamic 2-cell faults in the WD. The cells S00 and S10 activated by WL0 and WL1, respectively, located in the first column of the architecture, are used as inputs. The dynamic custom CA model table, including defects D4 to D7 and D11, is shown in Table XII. Note that both patterns can be used to detect this type of defect, since the lack of transition in the read value can be detected in either direction.

B. ATPG Results for Sense Amplifier Testing

The next step is to use the ATPG to generate testing patterns for the SA at memory level. The CA models generated previously, together with the Verilog description of the read SRAM architecture are used by the ATPG for this step.

	Inputs						Defects
	S00	**S10**	**WL0**	**WL1**	**BL0**	**BLB0**	**D4-D7, D11**
Pt.1	0	1	F	R	R	F	3
Pt.2	1	0	F	R	F	R	3

TABLE XII: Custom CA model for 2-cell dynamic faults in SA

Initially, the SAF in interconnections and the defects from the static table of the CA model are injected. A total of 24 (10 in interconnections and 14 from SCA table) are considered for the stuck fault model. Two patterns shown in Table XIII are generated. The patterns correspond to single R0 and R1 operations, that eventually need to be preceded by W0 and W1 operations. The initial fault coverage is 92%. The UD faults are the SAFs at the SA_EN input. For a SA0 (Stuck-At 0) fault, the defect that electrically represents the fault can potentially be defect D6, shown in Fig. 13. This defect is a short between the input SA_EN and ground and its detection conditions are explained in the previous section.

	Inputs							Outputs	
	Address	**WLEN**	**CEq**	**SA_EN**	**S00**	**S10**	**S..**	**z**	**zb**
P1	0000	1	1	1	0	X	X	0	0
P2	0000	1	1	1	1	X	X	1	1

TABLE XIII: Patterns generated for SAF and SCA in the SA

The SA1 (Stuck-At 1 fault) for SA_EN is not identified among the analyzed defects. To address this, we performed memory-level SPICE simulations to determine the detection conditions and properly map the defect. The simulation results are shown in Fig.15. From these simulations, we observe that the sense amplifier, which is constantly enabled, amplifies a small difference between the bit-lines, causing the outputs to get stuck at '0'. Therefore, the defect can be detected by a simple R1 operation, and we have mapped this defect in the static table. Even though this fault is not dynamic, it was classified as undetectable (UD) by the ATPG because, for a SA1 fault at the enable signal, a '0' must be applied to SA_EN. Under this condition, the sense amplifier outputs remain at high impedance values (ZZ), as BL and BLB cannot propagate when SA_EN = '0'. In this scenario, the ATPG cannot make a decision on the detectability of the fault, hence a SPICE simulation is necessary.

From this analysis, we can map the SA0 and SA1 faults in both the dynamic table of the custom CA model and the CA static table of SA. Re-running the ATPG after this step improves the fault coverage to 100%.

For the TFs in the interconnections and the defects in the dynamic table of the custom CA, the testing process follows the same steps as for the static ones. A total of 10 TFs and 5 from the DCA table are injected. The initial fault coverage is 86% and the transition faults on SA_EN TF0 and TF1 are the two faults to be classified as UD. The same procedure as for SA0 and SA1 is followed. Through defect analysis, we observe that the defects require dynamic 2-cell patterns and

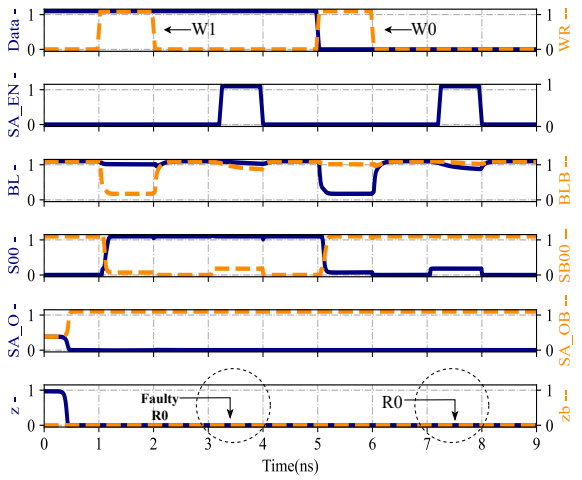


Fig. 15: Faulty behavior of SA due to SA1 at SA_EN

can be mapped to the dynamic table of the custom CA model. Another ATPG run is necessary to increase the fault coverage to 100%. The resulting dynamic patterns are given in Table XIV. For the first two patterns, a write operation is necessary between the two reads to change the content on the selected cell S00. However, accessing the same cell is not a necessary condition to detect the TFs. Hence, the latter two patterns are sufficient for the detection of all dynamic faults in the SA. Moreover the static patterns are included in the dynamic ones. Therefore, a total of 2 dynamic patterns (each containing R0W1R1 or R1W0R1) are necessary for the complete testing of the SA.

	Inputs							Outputs	
	Address	**WLEN**	**CEq**	**SA_EN**	**S00**	**S10**	**S..**	**z**	**zb**
P1/0	0000	1	1	X	0	1	X		
P1/1	0100	1	1	1	0	1	X	1	1
P2/0	0000	1	1	X	1	0	X		
P2/1	0100	1	1	1	1	0	X	0	0

TABLE XIV: Additional patterns generated from the 2nd ATPG run for SA testing

VII. Comparison and Validation Results

This section presents a comprehensive comparison between existing SRAM testing techniques and the results achieved using our CA methodology for the decoder and I/O circuitry. To ensure a fair assessment in terms of test complexity, we selected minimal March tests specifically designed to detect functional faults in the peripheral circuits, aligning with the fault models targeted by our approach. We provide a quantitative analysis of the fault coverage obtained by our methodology compared to that achieved with the selected March algorithms.

A. Decoder Comparison Results

To evaluate and compare the functional and structural testing approaches for address decoders, the MATS++ March algorithm was used as a reference. This algorithm was selected because it represents the simplest March algorithm capable of detecting all address faults (AFs) [25] and it has a $6n$ complexity, where n is the total number of cells in the considered memory array. While other March algorithms, such as the one proposed in [14], meet the conditions for detecting open defects like defect D3 discussed in this paper, they utilize algorithmic and generalized address switching rather than focusing on the specific locations of the defects considered. Moreover, these algorithms are significantly more complex and not directly comparable to our methodology for decoder testing.

The March elements in MATS++ have been translated to ATPG compliant patterns that can be used to run fault simulations in our Verilog memory description. The patterns correspond to write and read operations in the address order indicated for each March element. Specifically, we conducted two fault simulations targeting SAFs and SCAs, corresponding to W0R0 and W1R1 operations in both ascending and descending address sequences. This approach achieved a 92% defect coverage for all considered SAFs, encompassing both intra-cell and inter-cell defects. For TFs and DCAs, we performed two fault simulations using transition patterns derived from two March elements—namely, $\Uparrow$ $(r0w1)$ and $\Downarrow$ $(r1w0r0)$—executed in the specified address orders. This resulted in a 51% defect coverage for all considered TFs. Analysis of the undetected defects revealed that they are associated with falling transition faults and SAFs in the word line enable (WLEN), as well as falling TFs in each word line.

Table XV, shows the results obtained in ATPG for both functional (MATS++) and structural (CA methodology) approach. The complexity of the MATS++ algorithm is calculated as 6×16, where 16 represents the total number of memory cells in our 4x4 SRAM architecture. The number of patterns necessary to detect all defects using the CA approach is less than 96. This is because the ATPG considers only the first column when targeting the generation of patterns for row decoder defects. We have observed that the 52 stimuli necessary to detect TFs include the 12 necessary static stimuli for the detection of SAFs. Consequently, the total number of stimuli for the detection of TF and SAF for both inter and intra-cell defects in the considered row decoder is 52. To further validate our methodology, we applied both testing approaches to a 3-bit decoder within an 8x4 SRAM design. The results, also shown in Table XV, are consistent with those obtained for the 4x4 SRAM, demonstrating the feasibility and efficiency of our CA test methodology.

	ATPG		**MATS++**	
	SAF+SCA	TF+DCA	SAF+SCA	TF+DCA
2-bit Dec.	100%	100%	92%	51%
# of patt.	12	52	6x16 = 96	
3-bit Dec.	100%	100%	93.44%	49.2%
# of patt.	26	100	6x64 = 384	

TABLE XV: CA approach vs. MATS++

B. I/O Comparison Results

As discussed in Section II, research specifically targeting defect analysis in the I/O circuitry of SRAMs remains limited. For a long there has been a prevailing assumption in memory testing that faults in the peripheral circuits would be covered by tests designed for memory cell array faults [25]. However, a study dedicated to detecting faults in SRAM peripherals challenges this assumption and demonstrates that it is not always accurate [26]. In this study, the authors introduce minimal March tests specifically designed to detect peripheral faults and provide a comparative analysis of different March test performances. They explore several FFMs in the SRAM peripherals, particularly the Slow Write Driver Fault (SWDF), Slow Sense Amplifier Fault (SSAF), Slow Precharge Circuit Fault (SPCF), and Bit Line Imbalance Fault (BLIF).

Given the scope of this paper, we focus on the first two functional faults associated with the Write Driver (WD) and Sense Amplifier (SA). The authors in [26] propose four distinct minimal March tests to cover SWDF and SSAF. To address these faults, various March tests were executed under different algorithmic stresses. From the results, the March SR test proved effective in detecting both SWDF and SSAF, but it has a complexity of $14n$ and is designed to test the entire memory. Since our methodology focuses on individual memory blocks, we limit our comparison to minimal March tests specifically designed to detect functional faults in the peripheral circuits with minimal complexity, ensuring a fair comparison.

The minimal March tests are provided in Table XVI. We have selected the March test with the least complexity: March #1 for testing the WD and March #3 for testing the SA. These algorithms were translated into structural patterns that comply with the 4x4 SRAM and the specific memory operation. In Table XVI, the Data Background (DB) plays a crucial role in the test requirements, as it is one of the Degrees of Freedom (DOF) components of March tests, designed to enhance coverage capabilities [27]. The four considered DBs are the solid DB (sDB), checkerboard DB (bDB), column strip DB (cDB), and row stripe DB (rDB). The DB is denoted as D in the test sequence. Moreover, in Table XVI, the x value denotes the fast X (fX) addressing order. Fast X refers to an addressing order where each increment or decrement in the address accesses the adjacent physical row.

#	Name	Fault	DB	Test
1	March WDm	SWDF	bSB, rDB	x⇕(wD); x⇕ (rD); x⇕ (w$\overline{D}$); x⇕ (r$\overline{D}$)
2	March WDw	SWDF	sDB, cDB	⇕ (wD); x⇕($w\overline{D}$, $r\overline{D}$, wD); ⇕ ($w\overline{D}$); x⇕($wD, rD, w\overline{D}$)
3	March SAm	SSAF	sDB, cDB	⇕(wD); x⇕($rD, w\overline{D}$); x ⇕ ($r\overline{D}$, wD)
4	March SAw	SSAF	bSB, rDB	⇕(wD); x⇕(rD, wD); x ⇕ ($w\overline{D}$); ⇕ ($r\overline{D}$, $w\overline{D}$)

TABLE XVI: Minimal tests for SWDFs and SSAFs [26]

The first March test used for testing the SWDF has a complexity of $4n$, where 4 represents the number of operations in the test, and n denotes the number of cells in the memory array under test. The chosen data background for the translation is bDB, and the address ordering is fX. From this translation, a total of 64 patterns are generated. The CA model faults are added to the fault list in the TMAX tool, followed by a fault simulation. All defects listed in Table VII are detected; however, the defects represented in the custom CA model in Table IX are not detected by these 64 patterns. This is due to the fact that the detection condition specified in the CA model is not fulfilled. Specifically, in March Test #1, the transition $WriteD$ on one cell and the $Read\overline{D}$ on another cell within the same column are not fulfilled. As a result, the fault coverage achieved is 78%. The 9 patterns obtained through the ATPG can cover all of the considered faults in the WD core and WD controller.

The third March test used for testing the SSAF has a complexity of $5n$. Five operations are necessary for each bitcell and a total of 80 patterns are obtained from the translation. The chosen data background is sDB and the address order is fX. The CA model faults for the SA are added to the fault list and a fault simulation is run. From this fault simulation, 100% of the considered faults are covered. The minimal March test for SA fault detection proposed in [26] can cover the SA CA model faults efficiently. However, one of the differences between our approach and the March approach is the amount of patterns applied to the memory array to target specific faults in the periphery. The results are summarized in Table XVII. From our obtained results, a total of 6 patterns are necessary to cover the entire fault list in the CA model of the SA, whereas the minimal March test #3 is composed of 80 patterns. Moreover, in [26], faults are not attributed to specific defects in the design. Through the use of CA models, the information on defect location and the exact pattern that detects each of them is known. This information can potentially aid the diagnosis process, thereby highlighting another benefit of the CA methodology.

	Test	Complexity	Fault Coverage
WD	March #1	64	78%
	ATPG	9	100%
SA	March #3	80	100%
	ATPG	6	100%

TABLE XVII: I/O Comparison Results

VIII. Conclusion and Perspectives

This study introduces an enhanced methodology for SRAM memory periphery testing using the CA test concept, generating CA models for critical blocks such as the address decoder, sense amplifier, and write driver. By systematically addressing transistor-level defects and incorporating detailed defect detection conditions into custom CA models, the approach significantly reduces test generation time and complexity (42% less for the row decoder and 77.5% less for the I/O) while ensuring maximum fault coverage compared to selected March algorithms. Future work will consist in applying the methodology to the entire SRAM case study, allowing comparisons with more complex March algorithms targeting static and dynamic defects across the memory architecture. Additionally, this methodology will be explored for use in emerging memory technologies such as SOT and STT MRAMs.

IX. Awknowledgements

This work is done under the framework of the National CNRS 80PRIME project CARMEM, in collaboration with STMicroelectronics - Crolles and SPINTEC - CEA Grenoble.

References

[1] X. Xhafa *et al.*, "On using cell-aware methodology for sram bit cell testing," in *2023 IEEE European Test Symposium (ETS)*, pp. 1–4, 2023.

[2] IRDS, "International roadmap for devices and systems," in *https://irds.ieee.org/editions/2020*, 2020.

[3] P. Girard, Y. Cheng, A. Virazel, W. Zhao, R. Bishnoi, and M. B. Tahoori, "A survey of test and reliability solutions for magnetic random access memories," *Proceedings of the IEEE*, vol. 109, no. 2, pp. 149–169, 2020.

[4] A. J. Van de Goor and Z. Al-Ars, "Functional memory faults: a formal notation and a taxonomy," in *Proceedings 18th IEEE VLSI test symposium*, pp. 281–289, IEEE, 2000.

[5] S. Hamdioui, "Testing static random access memories: Defects, fault models and test patterns," 2004.

[6] M. Fieback, L. Wu, G. C. Medeiros, H. Aziza, S. Rao, E. J. Marinissen, M. Taouil, and S. Hamdioui, "Device-aware test: A new test approach towards dppb level," in *2019 IEEE International Test Conference (ITC)*, pp. 1–10, IEEE, 2019.

[7] F. Hapke *et al.*, "Cell-aware test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1396–1409, 2014.

[8] A. Bosio *et al.*, "Advanced test methods for srams: effective solutions for dynamic fault detection in nanoscaled technologies," 2009.

[9] P. Maxwell *et al.*, "Cell-aware diagnosis: Defective inmates exposed in their cells," in *European Test Symposium*, pp. 1–6, IEEE, 2016.

[10] P. Girard *et al.*, "Defect diagnosis techniques for silicon customer returns," in *Frontiers of Quality Electronic Design (QED) AI, IoT and Hardware Security*, pp. 641–676, SPRINGER, 2023.

[11] M. Klaus and A. J. Van de Goor, "Tests for resistive and capacitive defects in address decoders," in *10th Asian Test Symp.*, pp. 31–36, 2001.

[12] M. Sachdev, "Open defects in cmos ram address decoders," *IEEE design & test of computers*, vol. 14, no. 2, pp. 26–33, 1997.

[13] S. Hamdioui *et al.*, "Opens and delay faults in cmos ram address decoders," *IEEE Trans. on Comp.*, vol. 55, no. 12, pp. 1630–1639, 2006.

[14] L. Dilillo *et al.*, "New march elements for address decoder open and resistive open fault detection in srams," *Journal of Integrated Circuits and Systems*, vol. 3, no. 1, pp. 7–12, 2008.

[15] S. Hamdioui and Z. Al-Ars, "Scan more with memory scan test," in *2009 4th International Conference on Design & Technology of Integrated Systems in Nanoscale Era*, pp. 204–209, 2009.

[16] A. Ney *et al.*, "Slow write driver faults in 65nm sram technology: Analysis and march test solution," in *2007 Design, Automation & Test in Europe Conference & Exhibition*, pp. 1–6, 2007.

[17] A. Ney, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel, and M. Bastian, "Dynamic two-cell incorrect read fault due to resistive-open defects in the sense amplifiers of srams," in *12th IEEE European Test Symposium (ETS'07)*, pp. 97–104, 2007.

[18] A. Ney, P. Girard, S. Pravossoudovitch, A. Virazel, and M. Bastian, "Analysis of resistive-open defects in sram sense amplifiers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 10, pp. 1556–1559, 2009.

[19] A. Ladhar *et al.*, "Efficient and accurate method for intra-gate defect diagnoses in nanometer technology and volume data," in *2009 DATE Conference & Exhibition*, IEEE, 2009.

[20] TMAX, "Tmax user guide," in *https://www.synopsys.com/implementation-and-signoff/test-automation/testmax-atpg.html*.

[21] S. Hamdioui, Z. Al-Ars, A. J. Van De Goor, and M. Rodgers, "Dynamic faults in random-access-memories: Concept, fault models and tests," *Journal of Electronic Testing*, vol. 19, no. 2, pp. 195–205, 2003.

[22] F. Hapke *et al.*, "Cell-aware analysis for small-delay effects and production test results from different fault models," in *International Test Conference*, pp. 1–8, 2011.

[23] B. Mohammad *et al.*, "Comparative study of current mode and voltage mode sense amplifier used for 28nm sram," in *2012 24th International Conference on Microelectronics (ICM)*, pp. 1–6, 2012.

[24] A.-T. Do *et al.*, "Design and sensitivity analysis of a new current-mode sense amplifier for low-power sram," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 2, pp. 196–204, 2011.

[25] A. J. Van de Goor, "Testing semiconductor memories: theory and practice," 1991.

[26] A. J. Van de Goor, S. Hamdioui, and R. Wadsworth, "Detecting faults in the peripheral circuits and an evaluation of sram tests," in *2004 International Conferce on Test*, pp. 114–123, IEEE, 2004.

[27] D. Niggemeyer *et al.*, "Integration of non-classical faults in standard march tests," in *Proceedings. International Workshop on Memory Technology, Design and Testing*, pp. 91–96, 1998.

Xhesila Xhafa received her B.Sc. and M.Sc. degrees in Electronics Engineering from Istanbul Technical University, Turkey, in 2019 and 2021, respectively. She is currently pursuing her Ph.D. at LIRMM/CNRS in Montpellier, France. Her research interests include memory testing, integrated circuit design and reliability.

Eric Faehn obtained an engineer degree in microelectronic and radio electricity from the "Ecole Nationale Supérieure d'Electronique et de Radioélectricité de Grenoble" and a Master of Research degree in integrated system design from the "Université Joseph Fourier de Grenoble" in 2004. Employed after his studies by STMicroelectronics as design engineer and customer support, he first developed eDRAM controllers in the United States for disk drive products. In 2006, he carried out this activity from France for other business groups. In 2008, he started to work on eSRAM and oversaw the definition of optimal test algorithms and the design of the corresponding BIST. In 2010, he focused his activity on memory diagnosis and worked internally and with external tool suppliers and laboratories to enhance test and diagnosis capabilities. In 2019, his expertise has been recognized and he became member of technical staff in this domain.

Patrick Girard received a Ph.D. degree in Microelectronics from the University of Montpellier, France, in 1992. He is currently Research Director at CNRS (French National Center for Scientific Research) and works in the Microelectronics Department of the Laboratory of Computer Science, Robotics and Microelectronics of Montpellier (LIRMM) - France. He is Coordinator of the International Research Project INSIMIA (French-Italian Research Program on the Integrity of Integrated Systems in the Era of Artificial Intelligence) created by the CNRS and the University of Montpellier, France, with the Politecnico di Torino, Italy. He is deputy director of the French scientific network dedicated to research in the fields of Systemon- Chip, Embedded Systems and Connected Objects (SOC2). His research interests include all aspects of digital and memory testing, with emphasis on critical constraints such as timing and power. Robust design of neuromorphic circuits as well as machine learning for test and diagnosis are also part of his new research activities. He has supervised 45 PhD dissertations and has published 12 books or book chapters, 100 journal papers, and more than 250 conference and symposium papers on these fields. Patrick Girard is a Fellow of the IEEE.

Arnaud Virazel received the Ph.D. degree in Microelectronics from the University of Montpellier, France, in 2001. He is currently Professor at the University of Montpellier – LIRMM (Laboratory of Informatics, Robotics and Microelectronics of Montpellier) where he is responsible of the TEST ("Test and dEpendability of microelectronic integrated SysTems") team. He has published 9 books or book chapters, 50 journal papers, and more than 170 conference and symposium papers spanning diverse disciplines, including DfT, reliability, power-aware and memory testing and he has supervised 28 PhD thesis in these fields. He is the head of the electrical engineering department (about 450 students in BSc and MSc programs) at the University of Montpellier. His teaching topics are mainly focusing digital circuit design, test and reliability.