A Structural Testing Approach for SRAM Address Decoders using Cell-Aware Methodology

Xhesila Xhafa, Eric Faehn, Patrick Girard, Arnaud Virazel

A Structural Testing Approach for SRAM Address Decoders using Cell-Aware 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 to ensure the quality of a System on Chip (SoC) as technology nodes shrink, making circuits more prone to defects and reliability issues at nanometer scales. This paper presents an efficient testing flow based on an adaptation of the Cell-Aware (CA) test concept for the testing of memory address decoders. With the use of an Automatic Test Pattern Generator (ATPG), two different decoder architectures are tested for stuck and transition faults, addressing both intra and inter-cell defects. In this work, we show that this methodology results in a 100% intra and inter-cell defect coverage for both Transition Faults (TFs) and Stuck-At Faults (SAFs). To compare our results with existing solutions, the MATS++ algorithm has been used. A 51% improvement in TFs and 8% improvement in SAFs test coverages have been obtained through our Cell-Aware methodology.**

***Index Terms*—Memory testing, Structural testing, SRAM address decoders, Cell-Aware models, ATPG**

I. Introduction

Recent applications of Integrated Circuits (ICs) require an extensive amount of data to be stored and processed. Therefore, memory blocks now occupy a significant proportion, often up to 90%, of the System-on-Chip (SoC) area [1]. As the technology node shrinks, the memory density has increased significantly. However, with smaller transistor sizes and shorter proximities in interconnects due to higher densities, memories are now increasingly susceptible to defects and reliability challenges [2]. Thus, the testing of advanced and newly emerging memories has become a critical step in ensuring the quality and functional safety of manufactured ICs.

The most prominent memory testing approach is based on functional testing (i.e., March algorithms) [3]. However, with the increasing complexity of circuit designs, including emerging memory technologies such as MRAM and RRAM, functional testing becomes impractical due to high development costs and insufficient defect coverage to meet quality standards [4]. To address this, a shift from functional to structural testing is proposed in this paper. Structural testing relies on precise defect location and its impact on the input-output relationship, reducing fault modeling abstraction from the functional to transistor level, resulting in fewer test escapes and higher defect coverage. One implementation of structural testing is the Cell-Aware (CA) methodology, which addresses intra-cell defects in standard cells to reduce test escapes not caused by interconnect defects [5]. An initial flow for adapting CA to memory testing has been presented in [6] and applied to an SRAM bit cell, tailored for the memory array but insufficient for ensuring maximum coverage of defects in blocks outside the array. The SRAM memory periphery, particularly address decoders that comprise a large number of transistors and interconnects, is highly prone to defects thus contributing to overall test escapes [7], [8].

In this paper, we address the limitations of the aforementioned flow and propose an improved version that is adapted to testing address decoders. We introduce a systematic and comprehensive approach for SRAM address decoder testing that is based on the CA methodology and aims to achieve maximum defect coverage. The defects considered are located on interconnections and at the transistor level in each standard cell that composes the address decoders. A 4x4 array SRAM architecture, including its periphery, has been used as case study. The results are compared and validated with a functional approach, the MATS++ algorithm that has been developed to target address decoder faults with minimal complexity. Our methodology has demonstrated a notable improvement in defect coverage, achieving a 4% increase in SAFs and a 49% improvement in TFs fault coverages.

The paper is organized as follows. In Section II, the modified flow for address decoder testing using the CA methodology is introduced. The structural testing of address decoders is described in Section III. The obtained results and validation are given in Section IV. Finally, conclusions and perspectives are drawn in Section V.

II. Cell-Aware Methodology for Address Decoder Testing

The flow presented in [6] introduces a preliminary adaptation of the Cell-Aware (CA) methodology for memory testing but is limited to detecting intra-cell defects in the SRAM bit-cell, excluding the memory periphery. Here, we expand this flow to test address decoders in an SRAM architecture (see Fig. 1).

The CA methodology, initially designed for digital IC testing, involves generating CA models for each standard cell

within a digital circuit. This process starts with identifying potential intra-cell defects, such as transistor-level shorts and opens, and layout-based defects. Defects are injected into the circuit, and SPICE simulations are run to observe deviations in output signals. Detected defects are associated with input patterns, and this information is compiled into a CA model, which includes both statically and dynamically detected defects. These models are then used by an Automatic Test Pattern Generator (ATPG) to produce patterns for testing inter-cell and intra-cell defects in the overall IC.

A 4x4 memory array designed using a 28nm FDSOI technology that includes the memory array, address decoders, sense amplifier, write driver and pre-charge circuits is used as a case study. Following the initial flow, the memory is described at the gate level (structural Verilog) to generate CA models. These models, along with the gate-level structure, are used by the ATPG to generate patterns for detecting interconnect and cell-aware defects, considering SAF for static defects and TF for dynamic defects (1st ATPG run, see Fig. 1).

The address decoders are purely composed of digital gates. Hence, when the decoder is tested on its own, a test coverage of 100% is achieved. However, when used in the full memory architecture a considerable percentage of the defects located in the decoder are classified as UnDetectable by the ATPG. The UD defects are analyzed and we have observed that for these defects dynamic patterns that address more than one cell in the memory array are necessary. To address this issue, the proposed solution is to map the effects of the UD defects into the memory array (Decoder to Memory Array defect mapping, see Fig. 1). Since the row decoder controls the activation of Word Lines (WLs), only one column of the memory array is sufficient for this task. In order for the ATPG to comprehend the information regarding the switching of WLs that allows two cells to be addressed, a new CA model is generated for the first column (CLM0) of the array. The idea is to emulate the detection conditions so that the ATPG can generate structural patterns. This step will be further detailed in Section III. To obtain the additional patterns after the CLM0 CA model generation, another ATPG run is necessary (2nd ATPG run, see Fig. 1). Finally, the flow ends by merging all the test patterns and calculating the test coverage. The test coverage is a ratio of detected defects over the total number of defects.

By using the CA methodology as a structural test approach for address decoders we therefore inherit the advantages that come with CA testing. Through CA models, the exact location information of intra-cell defects is available, hence also contributing to the diagnosis process [5].

III. Structural Testing of Address Decoders

A. CA model generation for row decoder cells

CA models need to be generated for each module type in the decoder. In this case, the standard cells from the 28nm technology library have been used for the design. A CA model is generated for NOR and AND gates, using the steps mentioned in the CA model generation flow (c.f. Section II). The CA model files contain all the information on the

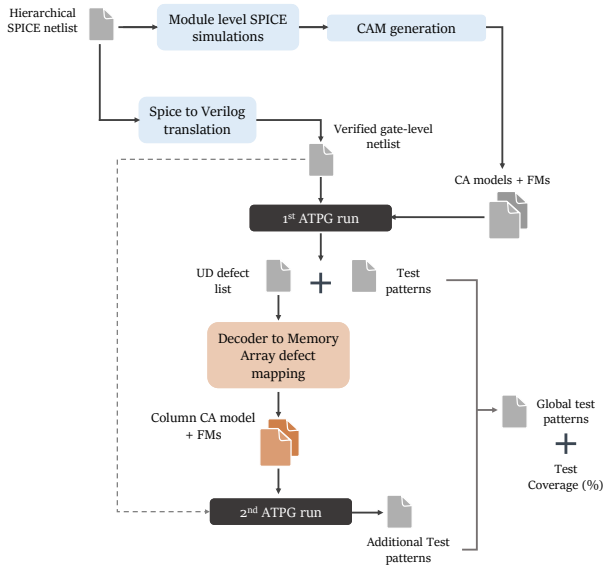

Fig. 1: Address Decoder Test Flow using a CA methodology

defect type and its location within the cell, as well as the detection tables with the necessary patterns to detect them. Defects that have equivalent detection conditions are classified as equivalent. When generating CA models, non-resistive open and short defects are considered, although the CA test can also address resistive defects [9]. Note that each defect is considered separately when running the SPICE simulations.

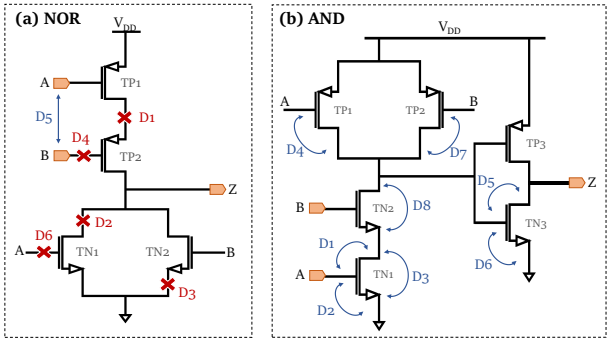

Fig. 2: The (a) NOR and (b) AND schematics including a subset of the injected defects

Depending on their detection conditions, the defects are classified and organized in static and dynamic tables. For demonstration purposes, we will show only one example for each of the tables. The location and type of the given defects in Tables I and II in the NOR gate and the AND gate are shown in Fig. 2. The static table of the AND CA model is shown in Table I. As seen under each defect name, the notation "1" is used to indicate that the pattern detects the defect, and a "0" indicates that the defect is not detected. Note that, in the static table, each pattern corresponds to at most one stimulus.

The dynamic table of the NOR CA model is shown in Table II. In the latter, the 'R' notation indicates a rising transition

	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 I: Static detection table of AND

from logic '0' to logic '1'. On the contrary, the 'F' notation indicates a falling transition from logic '1' to '0'. To better understand Table II we can analyze pattern 1. The '0R-F' input-output pattern can detect defects D3 to D5. 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 D3, D4, or D5.

	Inputs		Output	Defects					
	A	**B**	**Z**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
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	0	0	0	0	0

TABLE II: Dynamic detection table of NOR

B. Decoder to Memory Array Defect Mapping

In a memory circuit, the effect of each inter and intra-cell defect needs to be propagated from the defective net/s located in the decoder to the output of the memory, which in this case is the output of the Sense Amplifier (SA) circuit. It is, therefore, possible that the propagation of certain defects can be hindered by other memory blocks. We have observed that a part of the open defects, predominantly in the NOR gates, are not detected using only the information provided by the detection tables. This is due to the fact that while one defect can cause one world line to have a falling TF, the access from the targeted memory cell to the output is blocked. Therefore, the defect is not propagated.

To better understand this issue, we can analyze defect D3, which is an open defect located at the source net of transistor TN2 of a NOR gate (see Fig. 2). After a SPICE simulation with the injected defect, we observe the following detection conditions:

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

The (A, B) inputs of the defective NOR gate correspond to the row decoder inputs <A0, A1>. For D3 detection, as in the SPICE simulation, the decoder address switches from <0, 0> to <0, 1>. In this memory architecture, the delay in WL0's falling transition is blocked by the SRAM bit cell's access transistor primitive, preventing defect propagation (see Fig. 3). To address undetectable (UD) defects, these defects are mapped into a new CA model allowing to control multiple WLs. Since the row decoder activates the WLs, only one memory column (CLM0) is needed for this CA model. The new CA model is generated without SPICE simulations, using existing CA model information for NOR gates.

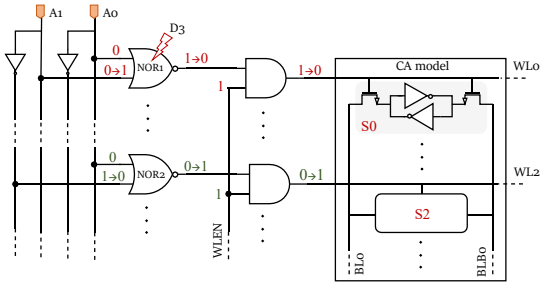

Fig. 3: Propagation of D3

An example of the CLM0 CA model dynamic table is given in Table III. In this table the pattern that detects the previously analyzed D3 in NOR1 is shown. Since the defect is synthesized by two consecutive write operations, the observable outputs are the inner nodes of the bit cells of CLM0 that contain the written value in the cell. This pattern detects all the defects detected by Pt.1 (see Table II) in the NOR dynamic table. The same procedure is followed for all UD defects in all of the gates of the row decoder.

Inputs						Outputs				Defects
WL0	**WL1**	**WL2**	**WL3**	**BL0**	**BLB0**	**S0**	**S1**	**S2**	**S4**	**D3,D4,D5**
F	0	R	0	R	F	0	0	R	0	1

TABLE III: Dynamic table of the CLM0 CA model

IV. Results and Validation

The CA models obtained are utilized by ATPG to generate test patterns for inter and intra-cell defects (SAF and TF models) in the row decoder of a 4x4 SRAM architecture.

A. Static Faults

The SAF detection process in the decoder involves three steps. First, the ATPG generates circuit-level patterns to detect SAFs in interconnections. These patterns are saved, and then, after adding Static Cell-Aware (SCA) defects, a fault simulation is run using the initial patterns to check defect coverage. If coverage is below 100%, another ATPG run generates patterns targeting undetected intra-cell defects. For the SAF model, 112 inter and intra-cell defects are considered. After both ATPG runs, 12 patterns are generated: 8 for W1R1 operation on CLM0 and 4 for detecting D8 of the AND cell, which causes an SAF1 on each WL detected by keeping the WLEN signal at level '0'. The ATPG generates four read patterns while the decoder is inactive, in the presence of D8, the SA reads an unexpected value. This structural pattern is noted as R_EN0.

After the two ATPG runs, the defect coverage is 96%, with four undetected D3 defects on each AND gate. This defect, detected with a '01' input (see Table I), results in an inactive output for each corresponding WL. Similar to an open defect, D3 is mapped to the CLM0 CA model. Following this third step, the SAF defect coverage reaches 100%.

B. Transition Faults

For the TF model, 92 inter and intra-cell defects are considered. The pattern generation process for TF detection is similar to that for SAFs. First, inter-cell defects are targeted using 4 static patterns to write values in each cell of CLM0. The initial ATPG run produces 14 transition patterns (28 stimuli). Figure 4 (a) shows these patterns and corresponding memory operations in terms of row decoder inputs <A0, A1>. Next, intra-cell defects from the Dynamic Cell-Aware (DCA) table are added, and a fault simulation is run using these patterns. In the second run, no additional patterns are generated and a defect coverage of 60% is achieved. As discussed in Section III, a part of NOR intra-cell defects that correspond to open defects causing a falling TF at the WL-s, are not detected. For this reason, the defects are mapped in the CA model of CLM0. The procedure explained in Section III-B is followed and using a third ATPG run the defect coverage is increased to 100%. In Fig. 4 (c), the lined arrow indicates the W0W1 operations from one address (e.g. '00') to the other (e.g. '01'). The curved lines indicate the read operation in the initial address (e.g. '00'). This step requires an additional 20 stimuli, resulting in a total of 54 stimuli needed for all TF detection. In Fig. 4, the obtained defect coverage cumulative percentage is indicated after each ATPG run. By combining the three graphs (a), (b) and (c) in Fig. 4, we observe that the obtained transition patterns cover the entire graph.

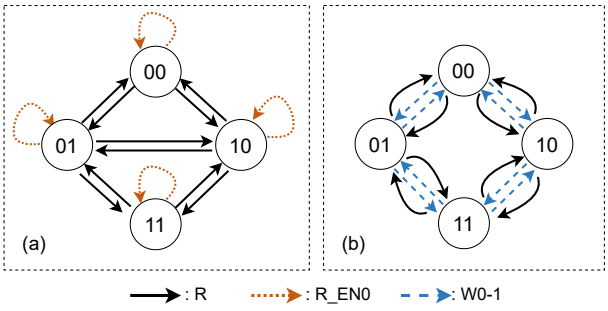

Fig. 4: Address orders of TF detection patterns for each ATPG run, where in (a) DC = 60%, (b) DC = 100%.

C. Comparison Results

To compare functional and structural approaches for testing the address decoder, the MATS++ algorithm is chosen as a reference due to its ability to detect all AFs while maintaining a low complexity [10]. The March elements in MATS++ were translated to ATPG-compliant patterns for fault simulations in our Verilog memory description, corresponding to write and read operations in the specified address order. Four fault simulations for SAFs (W0R0 and W1R1 operations in ascending and descending orders) yielded a 92% defect coverage for both intra- and inter-cell defects. For TFs, two simulations using transition patterns ($\Uparrow (r0w1); \Downarrow (r1w0r0)$) resulted in a 51% defect coverage. The undetected defects are analyzed and we have observed that the defects are related to the falling TF and SAFs in the WLEN, as well as falling TFs in each WL.

Table IV shows ATPG results for both functional (MATS++) and structural (CA methodology) approaches. The MATS++ algorithm's complexity is 6x16 for the 4x4 SRAM case study 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 CLM0 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. For validation purposes, a 3-bit decoder in an 8x4 SRAM design has also been tested using both methodologies. The results are shown in the two last lines of Table IV. Similar results have been obtained for the case of the 3-bit decoder as well, proving 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 IV: Proposed CA approach vs. MATS++

V. Conclusion

This study presents a modified CA methodology for structurally testing of address decoders in SRAM architecture. By using a custom CA model to map UD defects causing 2-cell dynamic faults onto the memory array, we achieved a complete defect coverage for both inter- and intra-cell defects in address decoders. Comparing this with the MATS++ algorithm shows significant improvement using CA methodology with minimal test patterns. Future work will implement this methodology on the entire SRAM memory architecture, generating CA models for all memory blocks to compare with more complex March-like algorithms targeting static and dynamic defects.

References

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

[2] S. Borkar *et al.*, "Microarchitecture and design challenges for gigascale integration," in *MICRO*, vol. 37, pp. 3–3, 2004.

[3] 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.

[4] P. Girard *et al.*, "A survey of test and reliability solutions for magnetic random access memories," *Proceedings of the IEEE*, vol. 109, no. 2, pp. 149–169, 2021.

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

[6] 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.

[7] S. Hamdioui, *Testing static random access memories: defects, fault models and test patterns*, vol. 26. Springer Science, 2004.

[8] A. Bosio *et al.*, *Advanced test methods for SRAMs: effective solutions for dynamic fault detection in nanoscaled technologies*. Springer Science & Business Media, 2009.

[9] 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.

[10] A. J. Van de Goor, *Testing semiconductor memories: theory and practice*. John Wiley & Sons, Inc., 1991.