A Runtime Efficient Graph-Based Cell-Aware Model Generation for Structural SRAM Testing

Gianmarco Mongelli, Xhesila Xhafa, Eric Faehn, Dylan Robins, Patrick Girard, Arnaud Virazel

A Runtime Efficient Graph-Based Cell-Aware Model Generation for Structural SRAM Testing

G. Mongelli[1,2], X. Xhafa[2], E. Faehn[1], D. Robins[1], P. Girard[2], A. Virazel[2]

[1]STMicroelectronics
Crolles, France
gianmarco.mongelli, eric.faehn, dylan.robins@st.com

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

***Abstract*—Testing advanced memories is essential for ensuring modern System-on-Chip quality. As transistor size continues to shrink, the probability of the occurrence of manufacturing defects increases, making conventional functional testing of SRAMs inadequate for achieving required Defect Parts per Million (DPPM). To address this issue, a novel structural testing approach using the Cell-Aware (CA) test methodology has been proposed in [1]. With this methodology, structural test patterns are obtained through an Automatic Test Pattern Generator (ATPG) by exploiting analog CA models (i.e., based on exhaustive analog simulations) for SRAM primary blocks. However, the generation of CA models through analog simulations is time-consuming and technology dependent. A methodology, namely TrUnDeL proposed in [2], is used to accelerate the CA model generation, combining a switch-level graph-based solution and analog simulations. In this work, we propose an adaptation of TrUnDeL to generate the CA models, using only the switch-level graph-based simulations. TrUnDeL CA models are then used by the ATPG to generate structural test patterns. Through a validation flow, we demonstrate that the aforementioned patterns, generated without running any analog simulations, achieve nearly the same fault coverage as the test patterns generated using exhaustive analog simulations on an SRAM case study. The time required to generate CA models is drastically reduced with TrUnDeL, from 1 hour to 15 seconds for the considered case study.**

***Index Terms*—Memory testing, SRAM, CA models, graph theory, ATPG.**

I. Introduction

With the advent of technologies like Edge Computing and the Internet of Things (IoT), the need for processing and storing large amounts of data has grown significantly. As a result, memory blocks, such as SRAMs, now occupy a significant portion of System-on-Chip (SoC) designs [3]. As CMOS transistors continue to scale down to nanometer sizes, the probability of manufacturing defects and reliability issues increases due to higher memory densities [4], [5]. Therefore, novel memory testing techniques, that address the challenges associated with advancements in shorter technology nodes and highly dense memories, are imperative to ensure the quality and reliability of memory systems.

Traditional SRAM testing methods, like March tests, are based on Functional Fault Models (FFMs) [6], [7]. These FFMs are characterized by a high abstraction level and do not systematically address the surge of new transistor level defects found in short node technology-based SRAMs. A structural approach to address transistor level defects is proposed in [1]. In this latter work, a shift from functional to structural testing using the Cell-Aware (CA) test methodology to address transistor-level defects is proposed. The CA methodology is based on the generation of CA models for each SRAM primary block. The considered CA defects in the generation of the CA models are open and short defects in the terminals of each transistor, as well as layout-based defects [8], [9]. Each defect is injected for every input stimuli of the primary SRAM block. As a result, the CA model provides a dictionary that maps which defects are detectable by which input stimuli. The generated CA models are then used by an Automatic Test Pattern Generator (ATPG) to generate structural test patterns that target CA defects, aiming to improve defect coverage and minimize test complexity.

The classical CA model generation flow is based on exhaustive analog simulations and is therefore time-consuming and technology-dependent [10]. To accelerate the CA model generation process, a methodology named Transistor Undetectable Defect Eliminator (TrUnDeL) was introduced in [2]. TrUnDeL identifies undetectable defects for each stimulus using a switch-level graph-based approach. Analog simulations are only performed for the remaining defects classified as possibly detected, thus reducing the number of simulations. TrUnDeL has been initially applied to characterize standard cell libraries, effectively decreasing the generation time by a factor of 3.

In this paper, we propose a methodology to generate structural test patterns for testing a 4x4 SRAM memory case study using ATPG, exploiting the CA models generated by TrUnDeL. The TrUnDeL CA models are generated using only the switch-level graph-based solution, without any analog simulations of defects classified as possibly detected. This marks a significant difference from previous work on TrUnDeL, where analog simulations were still employed for a subset of defects. Through validation, we demonstrate that the fault coverage achieved with ATPG using analog CA models (i.e., CA models based on exhaustive analog simulations) is nearly matched by using TrUnDeL CA models.

Through this methodology we provide the following contributions:

- Firstly, by avoiding analog simulations, our approach further accelerates the generation of TrUnDeL CA models (from 1 hour to 15 seconds for the considered case study) while maintaining a high test coverage for the SRAM case study, when compared to analog CA models.
- Secondly, while prior work applied TrUnDeL to standard cells, this paper extends its application to the diverse circuit structures of SRAM primary blocks, requiring specific adaptations to the TrUnDeL flow.
- Thirdly, TrUnDeL CA models are technology-independent, as the graph-based methodology is solely based on the transistor configuration of the circuit, without considering different technological aspects. As a result, TrUnDeL CA models are reusable across different SRAM.

This paper is organized as follows. Section II describes related prior work. Section III presents the TrUnDeL CA ATPG methodology. Validation and comparison results are presented in Section IV. Conclusions and perspectives are given in Section V.

II. Related Prior Work

Testing advanced memories, particularly SRAMs, requires innovative approaches to address the increasing probability of manufacturing defects as transistor sizes shrink. One recent approach found in the literature is the adaptation of the CA test methodology, originally designed for testing digital IC circuits, to structurally test memories with the use of an ATPG [1], [11], [12].

The process for adapting the CA methodology is shown in Fig. 1 and starts with converting a SPICE netlist, which describes the memory at the transistor level, into a gate-level netlist suitable for ATPG. This conversion ensures that each primary memory block is represented as a gate-level module, analogous to standard cells in digital ICs. A 4x4 SRAM architecture is used as a case study. The memory is organized in the following modules: memory array, sense amplifier, Write Driver (WD), address decoders and precharge circuits. Each of these modules are composed of primary blocks that are the fundamental indivisible circuits, such as the SRAM bitcell, the sense amplifier, the WD core, precharge circuit and all digital gates (i.e., NOR, AND). For each primary block, CA models are generated. Then, the gate-level memory description and CA models are provided to the ATPG to generate memory-level patterns for testing.

The CA methodology is used to detect and localize manufacturing defects for testing and diagnosis purposes for a given circuit. The CA model generation process starts by identifying potential CA defects, such as transistor-level shorts and opens, as well as layout-based defects. The CA methodology characterizes the circuit by building a dictionary, called Defect-Detection Matrix (DDM), that contains all the information about the detectability status of each defect df for each stimulus s. To build the DDM, one analog simulation

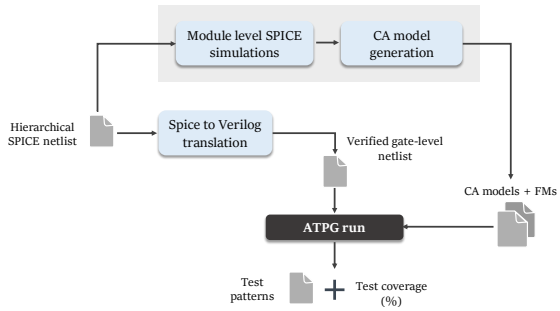

Fig. 1. Structural SRAM testing flow.

is performed for each df injected and for each s applied. Then, a detectability status is assigned to each (s, df) pair: UnDetectable (UD) if the defect does not affect the cell's outputs, Detectable (D) if the defect affects the cell's outputs. In the DDM, the defects that are classified as D for one-cycle stimuli are called static defects, while the defects classified as D for two-cycle stimuli are called dynamic defects. An example cell DDM is shown in Table I. The rows corresponds to the applied stimuli, while the columns correspond to the injected defects. Any entry of the matrix refers to a given (s, df) pair, and a detectability status is associated with each entry.

TABLE I
Example Defect-Detection Matrix.

	df_0	df_1	df_2	df_3	...	df_{j-1}
s_0	UD	D	UD	D	...	D
...	...	...	...	...	...	
s_{i-1}	D	D	UD	UD	...	UD

A drawback of the CA methodology is that it is a time-consuming process, as it needs to perform one analog simulation for each (s, df) pair, resulting in $s * df$ analog simulations for a complete circuit characterization. The methodology presented in [2], [13] proposes a graph-based approach that speeds-up the CA model generation process, called Transistor Undetectable Defect eLiminator (TrUnDeL). The TrUnDeL flow is shown in Fig. 2.

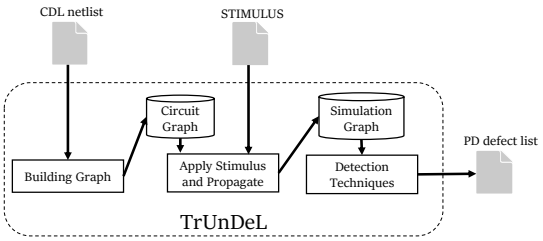

Fig. 2. TrUnDeL flow.

The process begins by building a graph using information extracted from the corresponding circuit netlist. The graph has two type of nodes (i.e., devices and nets, with edges representing the connections between devices and nets). An

example of inverter graph is shown in Fig. 3, where the device nodes are 'N' and 'P', the net nodes are 'VDD', 'GND', 'NET' and 'OUT' and each transistor has 4 terminal edges representing Bulk (B), Gate (G), Drain (D) and Source (S). The second step involves applying a stimulus to the input nodes of the graph, followed by propagating the stimulus through the various nodes using an iterative event-driven switch level algorithm. The final stage uses different detection techniques to identify UD transistor level defects, remove them from the initial Defect List (DL), and generate the Possibly Detectable (PD) DL as the output.

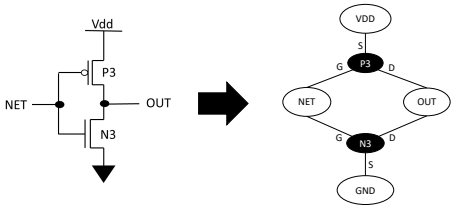

Fig. 3. resulting graph for an inverter

After the TrUnDeL run on the circuit, a set of (s, df) pairs is classified as UD while the remaining pairs are classified as PD. PD (s, df) pair means that one of the cell's output can be affected by the injection of the defect under the given stimulus. Only these PD (s, df) pairs are distinguished into UD and D pairs by using analog simulation, reducing the time-cost of CA model generation. TrUnDeL analyzes both static (s, df) pairs (i.e., one-cycle stimuli) and dynamic pairs (i.e., two-cycle stimuli).

III. TrUnDeL Cell-Aware ATPG

The TrUnDeL CA ATPG and validation flow is shown in Fig. 4. The methodology starts by generating the CA models using TrUnDeL. Then, ATPG uses the TrUnDeL CA models to generate test patterns for a 4x4 SRAM memory case study. Finally, the Fault Coverage (FC) achieved using TrUnDeL CA models in the ATPG process is compared to the FC achieved with analog simulation-based CA models. Each step of the flow is analyzed in detail in the next subsections.

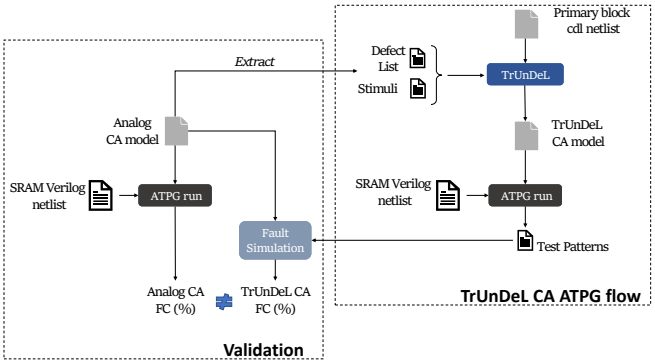

Fig. 4. TrUnDeL CA ATPG and validation flow.

A. TrUnDeL Cell-Aware model generation

The first step of the TrUnDeL CA ATPG flow is to generate CA models using TrUnDeL. Applying TrUnDeL to the SRAM primary blocks requires several adaptations. The first adaptation must be performed in the propagation step of TrUnDeL flow for the bitcell analysis. Fig. 5 shows the bitcell representation in which we want to write the opposite value of the stored one, with BL = '0' (i.e., BL has a path to the GND) and S = '1' (i.e., S has path to VDD). Here, P1 and N2 transistors fight to assign the value to S. The bitcell is designed to have pass transistors with a bigger driving capability than the inverter PMOS transistors. So, the N2 pass transistor wins, and after the write operation ends, the new value of S is '0'.

This is a case not supported by TrUnDeL, as in propagation only signal strength was modeled while the transistor strength was not considered yet. In fact, when a net had a path with both GND and VDD, it was set to 'X' (i.e., unknown value). To fix the problem, we assigned a 'driving capability' to all the pass transistor nodes of the graph. During the propagation, in the case of a net with a path with both GND and VDD, the path that contains the transistor with the 'driving capability' wins. In the bitcell in Fig. 5, S has a path to GND through the pass transistor N2, that has the 'driving capability'. So S becomes '0', consequently SB becomes '1'.

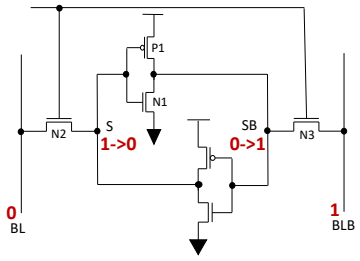

Fig. 5. Writing opposite value in bitcell.

The second adaptation involves the generation of the DDM for a TrUnDeL CA model. During the classification, TrUnDeL identifies UD pairs, while the remaining pairs are classified as PD. The TrUnDeL CA model is generated by storing UD and PD pairs in the corresponding (s, df) entries of the TrUnDeL DDM.

The key aspect for TrUnDeL CA models is how to interpret a PD (s, df) pair, and the consequences of considering this new detectability status in a CA model. A PD pair means that a defect under a given stimulus might be detected at one of the circuit's outputs. Therefore, in reality, the defect can be either D or UD. TrUnDeL can classify a defect as PD for both static and dynamic stimuli. In analog CA models, if a defect is classified as D for a static stimulus, it is not analyzed for any dynamic stimuli. In TrUnDeL CA models we never conclude that a defect is D for a static stimulus, so we could never remove a static (s, df) pair in the DDM.

To address this problem, one idea could be to analyze all the defects for both static and dynamic stimuli. However, by construction, CA model does not allow mapping a defect for

both static and dynamic stimuli. Moreover, dynamic stimuli contain all the necessary static stimuli to cover all the static defects [12]. As a result, TrUnDeL CA model builds a purely dynamic DDM (i.e., DDM only composed of dynamic stimuli) that guarantees building a compatible CA construction while preventing a loss of fault coverage.

The last aspect of the TrUnDeL DDM construction is that it only addresses transistor-level defects and does not provide any classification for layout-based defects. This is not an issue, as all layout-based defects are simply classified as always PD in the DDM to assure the accuracy of the TrUnDeL CA models. With this final modification, TrUnDeL CA models are ready to be used by an ATPG.

B. ATPG run using TrUnDeL CA models

Once TrUnDeL CA models are ready, an ATPG is run to generate the test patterns. For this purpose, TMAX from Synopsys is used [14]. It uses the TrUnDeL CA models to generate the minimum number of test patterns capable of achieving the maximum possible fault coverage (FC) at the IOs of the 4x4 SRAM memory.

However, one aspect must be considered when using TrUnDeL CA models in the ATPG process. To generate test patterns at SRAM memory level IOs that cover all the defects for a primary block, ATPG exploits the corresponding CA model and its DDM. ATPG starts from the first stimulus of the DDM, it applies it on the IOs of the primary block and it propagates until the IOs of the SRAM memory. Then, it continues until all the defects are covered.

Table II shows the comparison between the TrUnDeL DDM and the analog DDM of a given hypothetical primary block. As we can see, both defects are already PD for first stimulus of the TrUnDeL DDM, while we need both stimuli to detect the two defects in the analog DDM. This is due to the $(s0, df1)$ pair being PD for TrUnDeL DDM but UD for analog DDM. In TrUnDeL methodology all the (s, df) pairs that are PD for TrUnDeL and UD for analog simulation are defined as gap pairs [13]. This means that when using TrUnDeL CA models in ATPG process, we could lose FC due because some stimuli are ignored due to the gap pairs. For example, in this case, the second stimulus is ignored.

TABLE II
TRUNDEL DDM (ON THE LEFT) VS ANALOG DDM (ON THE RIGHT).

	df_0	df_1
s_0	PD	PD
s_1	UD	PD

	df_0	df_1
s_0	D	UD
s_1	UD	D

C. Validation Flow

The last step of the flow shown in Fig. 4 is the validation flow. The generated test patterns from the TrUnDeL CA ATPG flow and analog CA models of the primary blocks are provided to run the fault simulations. As a result, the fault simulation provides the FC achieved using TrUnDeL test patterns. The TrUnDeL FC is then compared with the FC generated by using analog CA models in the ATPG process.

IV. VALIDATION AND COMPARISON RESULTS

A. TrUnDeL Cell-Aware vs analog Cell-Aware models

The generated TrUnDeL CA models of the primary blocks are compared with the already existing analog CA models generated by using analog simulations. The first primary blocks we analyzed are the ones related to the column and row decoders: NOR, AND and Transmission Gate (TG). The comparison between TrUnDeL and analog CA models for these blocks are shown in Fig. 6.a, Fig. 6.b and Fig. 7.a.

The AND that we used is a cell from the P28 FDSOI library. It has 2 inputs, 1 output and 6 transistors. We analyzed 63 defects in total resulting in 360 (s, df) pairs (i.e., 144 static and 216 dynamic). For this cell, TrUnDeL does not produce any misclassifications, meaning that no D (s, df) pairs for analog CA model are incorrectly classified as UD for TrUnDeL CA model. Moreover, the gap is 22.9% (i.e., PD pairs in TrUnDeL CA models, but UD in analog CA models). TrUnDeL successfully identifies 53.9% of the UD (s, df) pairs. The remaining 46.1% are categorized as PD.

Also the NOR is a P28 FDSOI library cell. It has 2 inputs, 1 output and 4 transistors. 39 defects are analyzed, resulting in 186 (s, df) pairs (i.e., 96 static and 90 dynamic). 52.1% UD pairs are identified, with 0 misclassifications and 18.3 % gap. Finally, the last primary block of the decoders is the TG with 2 inputs, 1 output and 2 transistors. In total, 18 defects are analyzed, resulting in 40 pairs (i.e., 24 static and 16 dynamic). 42.5% UD pairs are identified with 0 misclassifications and 22.5% gap.

Another important module of the SRAM memory is the WD. The WD is composed of inverter gates, NOR gates and the WD core. Fig. 7.b reports the WD core results. The WD core has 2 inputs and 2 outputs and is composed of 8 transistors. With 28 defects identified, 76 tuples are analyzed (i.e., 30 static and 46 dynamic). TrUnDeL identifies 34.2% UD (s, df) pairs with 0 misclassifications and 42.1% gap.

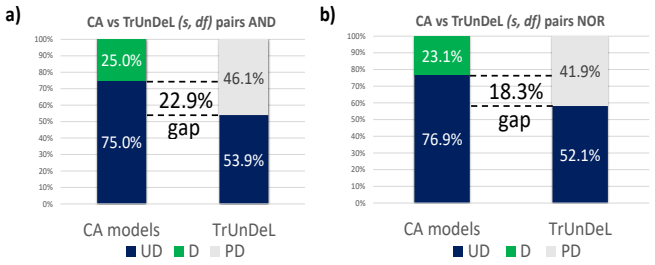

Fig. 6. TrUnDeL vs analog CA AND and NOR.

The next primary block we present is the bitcell that composes the array of the SRAM memory. The analog CA vs TrUnDeL CA comparison for the bitcell is shown in Fig. 8. As we can see, we distinguish two different comparisons for the read and write operations. This is because BL and BLB nets are bidirectional. ATPG does not support bidirectional signals, so we need two different architectures for the bitcell, consequently two distinguished CA models [11]. In the read

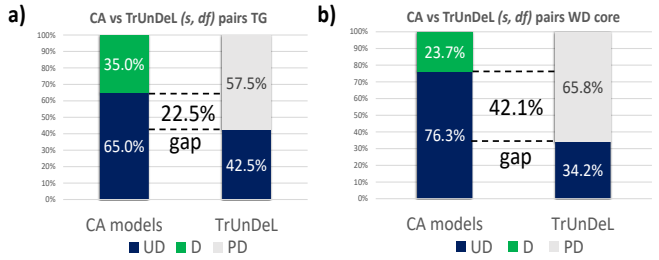

Fig. 7. TrUnDeL vs analog CA TG and WD core.

architecture, S and SB are inputs, and BL and BLB are outputs. In the write architecture, BL and BLB are inputs, while S and SB are outputs. Fig. 8.a shows the results for the bitcell read. The cell has 2 inputs, 2 outputs and 6 transistors. 6 defects are analyzed, resulting in only 12 static (s, df) pairs. 16.7% UD pairs are identified with 0 misclassifications and 33.3% gap. Fig. 8.b shows the results for the bitcell write. Again, the cell has 2 inputs, 2 outputs and 6 transistors. 20 defects are analyzed, resulting in 44 (s, df) pairs (i.e., 24 static and 20 dynamic). 18.2% UD pairs are identified with 0 misclassifications and 40.9% gap.

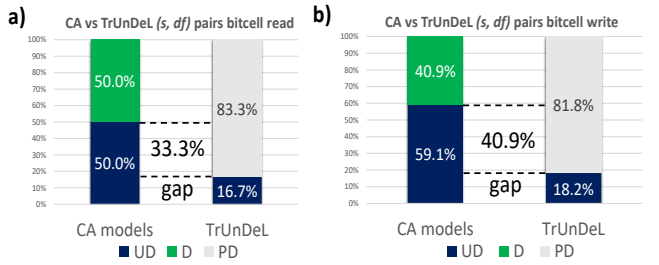

Fig. 8. TrUnDeL vs analog CA bitcell read and write.

The last primary block we consider is the Sense Amplifier (SA), with the comparison results shown in Fig. 9.a. The SA has 2 inputs, 2 outputs and 15 transistors. 38 defects are analyzed, resulting in 84 (s, df) pairs (i.e., 66 static and 18 dynamic). 14.2% of UD pairs are identified and the gap is 42.9%. However, in this case we have 2 misclassifications, as shown in Fig. 9.b.

The two misclassifications are complementary and have the same cause. Let's consider the first misclassified (s, df) pair in detail. It is a Short between Drain and Source (SDS) of Mn1 during read 0 (i.e., the $(Read0, SDS\_Mn1)$ pair). The sense amplifier is an analog block while TrUnDeL has a digital behavior. During read 0, BL starts to discharge from Vdd to Gnd by a small amount that we call Δ, resulting in a voltage value on BL equal to Vdd-Δ. The SA is designed to quickly perform the read operation and as soon as the Δ discharge is complete, it reads 0 on the output SA_0. TrUnDeL indeed has a digital behavior, so BL discharges completely to Gnd, and then the sense amplifier reads 0 on SA_O. This means that for the analog behavior, transistor Mn3 is active while for TrUnDeL transistor Mn3 is inactive. This leads to the $(Read0, SDS\_Mn1)$ pair misclassification.

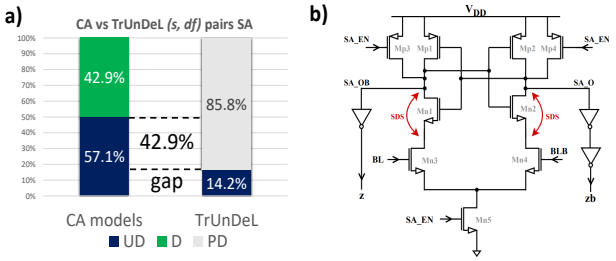

Fig. 9. TrUnDeL vs analog CA SA and misclassifications.

In terms of runtime, generating TrUnDeL CA models for all SRAM primary blocks takes about 15 seconds (using 1 CPU and 8 MB of memory), whereas generating analog CA models takes approximately 1 hour (using a single SPICE license). TrUnDeL is significantly faster because it uses a switch-level graph-based solution that avoids analog simulations, unlike the classical analog CA model generation, which relies on exhaustive and time-consuming analog simulations. This is the runtime gain for a simple 4x4 SRAM case study. For a real case SRAM, the runtime gain would change as the complexity and the number of primary blocks increase.

B. Fault Coverage comparison

Once all the CA models are generated for each primary block, the ATPG run generates the 4x4 SRAM test patterns. Then, the validation flow compares the FC achieved by test patterns generated by using TrUnDeL CA models with the FC achieved by using analog CA models. This subsection presents this FC comparison results module by module, as shown in Table III.

TABLE III
TRUNDEL FC VS ANALOG FC MODULES.

		TrUnDeL		*Analog*	
	#_defects	*FC*	*patterns*	*FC*	*patterns*
Write_driver	41	85%	2	85%	2
Sense_ampl	29	100%	2	100%	2
Array_read	128	100%	64	100%	64
Array_write	256	75%	32	100%	64
Row_decoder	100	48%	7	60%	9
Clm_decoder	220	80%	10	82%	17

The validation flow has been applied to all the modules that exploit TrUnDeL CA models excluding precharge module analysis, as all the necessary test patterns to test precharge defects are already generated by analyzing all the other modules [15]. Moreover, during the analysis, the ATPG does not consider all the equivalent defects to avoid useless run. The first module presented in the table is the write driver. A total of 41 defects are analyzed. A FC of 100% is achieved using either analog CA models of the write driver core and NOR, or the TrUnDeL CA models. To obtain this fault coverage, ATPG generates 2 patterns using both analog and TrUnDeL CA models.

The second module that we analyze is the sense amplifier. In this case, both the TrUnDeL and analog FC are the same: 100% achieved by the same 2 patterns. We analyzed the sense amplifier despite the 2 misclassifications. The misclassifications affect the TrUnDeL CA model, modifying the DDM, where a given (s, df) entry pair is UD instead of PD. However, this does not affect the ATPG behavior for the sense amplifier since other defects in the sense amplifier are mapped to the necessary stimuli that cover the misclassified UD defects. Therefore, all the necessary patterns to fully test the SA are generated by the ATPG and the FC is not reduced.

The next analyzed module is the array. In Table III, we have two cases: array_read that exploits the bitcell_read primary block, and array_write that exploits the bitcell_write primary block. Consequently, the ATPG uses two different verilog SRAM architectures one for read and one for write. The result for array read is 100% FC for both TrUnDeL and analog CA models in ATPG process, for 128 defects and generating 64 patterns in both cases. For array write, 256 defects are analyzed. In this case, the FC achieved by using TrUnDeL CA models is 75% with 32 patterns generated, while the analog one is 100% with 64 patterns generated.

The last two modules are the decoders. For the row decoder, with 100 defects, an FC of 48% is achieved using TrUnDeL CA models with 7 generated patterns. A FC of 60% is achieved using analog CA models with 9 generated patterns. For the column decoder, with 220 defects, the TrUnDeL FC is 80% with 10 generated patterns, while the analog FC is 82% with 17 generated patterns.

C. Modules Fault Coverage discussion

We can divide the memory modules into three categories based on the FC obtained using TrUnDeL CA models on the 4x4 SRAM test case and how this FC can be affected by the size of the memory.

In the first category, we have the decoders. For these modules, the CA ATPG generates fewer test patterns, and the TrUnDeL FC does not reach the analog FC. We still need analog simulations of the PD pairs to reach the maximum possible FC. As the memory size increases, the TrUnDeL FC might decrease due to the increased complexity of the decoders.

In the second category, there is the array write module. This module does not reach the highest possible FC using TrUnDeL CA models, as fewer patterns are generated. Analog simulation of the PD pairs is still needed to achieve the maximum possible FC. The size of the circuit does not affect the FC. This is because the bitcells are replicated more frequently as the size of the architecture increases, while their internal structures remain quite similar.

Finally, in the last category, there are the write driver, sense amplifier, and array read modules. ATPG runs on these modules generate the same number of patterns using TrUnDeL CA models as with analog CA models. Consequently, we achieve maximum FC using TrUnDeL CA models without the need for any analog simulation. The TrUnDeL CA ATPG will be effective for these modules, even when considering larger memory architectures as the primary blocks are simply replicated more frequently.

V. Conclusions and Perspectives

In this paper, we presented the TrUnDeL CA ATPG methodology, an alternative way to generate structural test patterns for a 4x4 SRAM memory test case, exploiting TrUnDeL CA models of the SRAM primary blocks. The TrUnDeL CA models are generated using only TrUnDeL (i.e., a switch-level graph-based solution), without the need for any analog simulations, Drastically reducing the generation time from 1 hour to 15 seconds. The TrUnDeL CA ATPG process is able to achieve a FC close to the one reached by using the analog CA ATPG process for the memory modules. We also analyzed all the TrUnDeL CA models comparing them with the analog CA models. Only SA had 2 misclassifications. This is due to the purely digital nature of TrUnDeL, while SA is an analog circuit.

One perspective is to adapt TrUnDeL to analog circuit to eliminate misclassifications. Another perspective is to reduce the gap, leading to a more effective CA model generation and to an increment of the FC.

References

[1] X. Xhafa *et al.*, "Sram periphery testing using the cell-aware test methodology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10.1109/TCAD.2024.3506854.

[2] G. Mongelli, E. Faehn, D. Robins, P. Girard, and A. Virazel, "A fast and efficient graph-based methodology for cell-aware model generation," in *2024 IEEE International Test Conference (ITC)*, pp. 270–279, 2024.

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

[4] S. Hamdioui, "Testing embedded memories: A survey," in *Mathematical and Engineering Methods in Computer Science: 8th International Doctoral Workshop*, pp. 32–42, Springer, 2013.

[5] 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.

[6] 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.

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

[8] F. Hapke *et al.*, "Cell-aware test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 9, 2014.

[9] Z. Gao *et al.*, "Defect-location identification for cell-aware test," in *2019 IEEE Latin American Test Symposium (LATS)*, pp. 1–6, IEEE, 2019.

[10] Z. Gao *et al.*, "Application of cell-aware test on an advanced 3nm cmos technology library," in *2019 IEEE International Test Conference (ITC)*, pp. 1–6, IEEE, 2019.

[11] X. Xhafa, A. Ladhar, E. Faehn, L. Anghel, G. Di Pendina, P. Girard, and A. Virazel, "On using cell-aware methodology for sram bit cell testing," in *2023 IEEE European Test Symposium (ETS)*, pp. 1–4, 2023.

[12] X. Xhafa, E. Faehn, P. Girard, and A. Virazel, "A structural testing approach for sram address decoders using cell-aware methodology," in *2024 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–4, 2024.

[13] G. Mongelli, X. Xhafa, E. Faehn, D. Robins, P. Girard, and A. Virazel, "A graph-based methodology for speeding up cell-aware model generation," in *2024 IEEE 30th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–6, 2024.

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

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