Test and Diagnosis of Hard-to-Detect Faults in FinFET SRAMs

Cardoso Medeiros, G.

DOI
10.4233/uuid:94469c76-9ee1-48b3-981f-f0d3d427b1d5

Publication date
2022

Document Version
Final published version

Citation (APA)
Cardoso Medeiros, G. (2022). *Test and Diagnosis of Hard-to-Detect Faults in FinFET SRAMs*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:94469c76-9ee1-48b3-981f-f0d3d427b1d5

Test and Diagnosis of Hard-to-Detect Faults in FinFET SRAMs

GUILHERME CARDOSO MEDEIROS

TEST AND DIAGNOSIS OF HARD-TO-DETECT FAULTS IN FINFET SRAMS

TEST AND DIAGNOSIS OF HARD-TO-DETECT FAULTS IN FINFET SRAMS

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
vrijdag 10 juni 2022 om 15:00 uur

door

Guilherme CARDOSO MEDEIROS

Mestre em Engenharia Elétrica
Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, Brazilië
geboren te Porto Alegre, Brazilië.

Dit proefschrift is goedgekeurd door de

promotor: Prof. dr. ir. S. Hamdioui
copromotor: Dr. ir. M. Taouil

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. S. Hamdioui,	Technische Universiteit Delft, promotor
Dr. ir. M. Taouil,	Technische Universiteit Delft, copromotor
Onafhankelijke leden:	
Prof. dr. S. di Carlo	Politecnico di Torino, Italy
Prof. dr. G. di Natale	Université Grenoble Alpes, France
Prof. dr. G. Q. Zhang	Technische Universiteit Delft
Dr. ir. J. S. S. M. Wong	Technische Universiteit Delft
Prof. dr. L. C. N. de Vreede	Technische Universiteit Delft, reservelid
Overig lid:	
Dr. L. M. B. Pöhls	RWTH Aachen, Germany

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 722325.

Keywords: memory test, manufacturing test, SRAM, FinFET, manufacturing defect, fault model, test solution, design-for-testability, diagnosis
Printed by: IPSKAMP printing, Enschede, the Netherlands
Front & Back: designed by Guilherme Cardoso Medeiros

ISBN 978-94-6366-565-0

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

Dedicated to those who have used their time to make me
a better student, researcher, and human being.

SUMMARY

The *Fin Field-Effect Transistor* (FinFET) technology became the most promising approach to enable the downscaling of technological nodes below the 20 nm threshold. However, the introduction of new technology nodes for embedded memories such as SRAMs, especially for even smaller nodes such as 10 and 5 nm, gives rise to new manufacturing failure mechanisms; these could both impact the yield and the outgoing product quality in the absence of appropriate diagnosis and test solutions. Therefore, there is a need for effective yet cost-efficient test and diagnosis solutions. This thesis contributes to fault modeling, test development, and diagnosis of FinFET based SRAM.

Fault Modeling: although a lot has been done on memory fault modeling, the focus has been mainly on *easy-to-detect* (ETD) faults; these are faults for which the detection can be guaranteed by march algorithms, i.e., writing and reading the memory. The behaviors of *hard-to-detect* (HTD) faults, which are the main focus of this thesis, are by far much more complex; these faults may even require dedicated hardware to facilitate their detection, i.e., *design-for-testability* (DFT) circuits. This thesis provides a complete analysis of hard-to-detect faults in FinFET based SRAMs and validates them using circuit simulations. Deriving realistic fault models is the key enabler for high-quality test solutions. Hence, it requires a deep understanding of how the defects may impact the memory.

Test Development: existing test solutions focus on the detection of ETD faults. They do not require special tests; their detection is guaranteed by writing and reading the memory. Nevertheless, HTD faults are *not* deterministic, i.e., they have a random nature. Therefore, their detection is not guaranteed by simply applying operations – additional solutions that employ special stressing are needed. This thesis proposes multiple test solutions based on the results obtained during the fault modeling stage. First, it discusses test solutions for *random read faults* (RRFs). These faults' detection is improved by optimizing test algorithms to provide the maximum stress possible. An insufficient fault coverage was observed by only applying testing algorithms; thus, dedicated DFTs are developed. A similar procedure is carried out for *undefined state faults* (USFs); these faults' coverage can only be improved by using dedicated DFTs. Finally, a test methodology for parametric faults is proposed; the methodology consists of monitoring the memory's parameters and comparing measurements from different parts of the memory. However, its implementation does not tolerate *process variation* (PV) effects. Therefore, a novel way to implement the methodology is needed.

Diagnosis: production test solutions do not provide any information regarding fault location or nature. However, during the development of a chip, more information regarding failure mechanisms is often necessary. Thus, diagnosis procedures are employed to understand what must be improved in the chip's design and manufacturing process. This thesis proposes a diagnosis methodology that hierarchically diagnoses *all types* of faults (e.g., static, dynamic, ETD, HTD) coming from *all parts* of the memory (e.g., cell array, decoders, peripherals). First, a comprehensive algorithm is applied to detect faults in the entire memory. Once the faulty block is identified, another algorithm is applied to discern whether the fault is static or dynamic. Finally, multiple short algorithms are applied to identify the fault model. The proposed methodology is easily extensible; new diagnostic capabilities can be easily added by integrating new diagnostic algorithms without impacting existing capabilities. Furthermore, it is platform-independent; it does not rely on a specific memory implementation or architecture.

SAMENVATTING

De *Fin Field-Effect Transistor* (FinFET) werd de meest veelbelovende technologie om transistorschaling voorbij de 20 nm-drempel mogelijk te maken. De introductie van kleinere transistortechnologieën, vooral voor nog kleinere technologieën zoals 10 en 5 nm nm, voor ingebedde geheugens zoals SRAM's, leidt echter tot nieuwe bezwijkmechanismen bij de fabricage. Deze mechanismen kunnen zowel het productierendement als de uitgaande productkwaliteit beïnvloeden bij gebrek aan geschikte diagnose- en testoplossingen. Er is daarom behoefte aan effectieve en kostenefficiënte diagnose- en testoplossingen. Dit proefschrift draagt bij aan foutmodellering, testontwikkeling en diagnose van op FinFET's gebaseerde SRAM's.

Foutmodellering: hoewel er veel is gedaan aan geheugenfoutmodellering, ligt de nadruk vooral op *eenvoudig-te-detecteren* (ETD) fouten; dit zijn fouten waarvan de detectie kan worden gegarandeerd door marcheeralgoritmen, d.w.z. simpelweg het schrijven en het uitlezen van het geheugen. Het gedrag van *moeilijk-te-detecteren* (MTD) fouten, die de focus van dit proefschrift zijn, is veel complexer. Om deze fouten te detecteren, kan het zelfs nodig zijn om speciale detectiehardware, d.w.z. *ontwerp-voor-test* (OVT) circuits, te gebruiken. Dit proefschrift biedt een volledige analyse van moeilijk-te-detecteren fouten in op FinFET gebaseerde SRAM's en valideert deze fouten door middel van circuitsimulaties. Het afleiden van realistische foutmodellen is de belangrijkste factor voor hoogwaardige testoplossingen. Het is daarom vereist om een diep begrip te hebben van hoe defecten het geheugen kunnen beïnvloeden.

Testontwikkeling: bestaande testoplossingen richten zich op het opsporen van ETD-fouten. Ze vereisen geen speciale tests; hun detectie wordt gegarandeerd door naar het geheugen te schrijven en het vervolgens uit te lezen. MTD-fouten hebben een willekeurig karakter en zijn daarom *niet* deterministisch. Hierdoor kunnen deze fouten niet gedetecteerd worden door gebruik te maken van simpele geheugenoperaties - er zijn aanvullende oplossingen nodig die speciale stresscondities gebruiken. Dit proefschrift presenteert meerdere testoplossingen op basis van de resultaten die zijn verkregen uit de foutmodellering. Ten eerste worden testoplossingen voor willekeurige leesfouten besproken. De detectie van deze fouten wordt verbeterd door testalgoritmen te optimaliseren om de maximaal mogelijke stress te bieden. De foutendekking die behaald wordt door enkel testalgoritmen te gebruiken is echter te laag. Hierom worden speciale OVT circuits ontwikkeld. Iets soortgelijks wordt gedaan voor ongedefinieerde toestandsfouten, omdat de dekking van deze fouten alleen kan worden verbeterd door speciale OVT's te gebruiken. Ten slotte wordt een testmethodologie voor parametrische fouten geïntroduceerd. De methodologie bestaat uit het controleren van geheugenparameters en het vergelijken van metingen van verschillende delen van het geheugen. De implementatie hiervan tolereert echter geen effecten van procesvariaties. Daarom is er een nieuwe manier nodig om de methodologie te implementeren.

Diagnose: productietestoplossingen geven geen informatie over locatie of aard van de fout. Tijdens de ontwikkeling van een chip is echter vaak meer informatie over bezwijkmechanismen nodig. Daarom wordt diagnostisering gebruikt om te begrijpen wat er moet worden verbeterd in het ontwerp en in het fabricageproces van de chip. Dit proefschrift presenteert een diagnostiseringsmethodologie die op hiërarchische wijze *alle soorten* fouten (bijv. statische, dynamische, ETD, MTD) diagnosticeert in *alle delen* van het geheugen (bijv. in de geheugenmatrix, decoders, randcircuits). Eerst wordt een uitgebreid algoritme toegepast om fouten in het gehele geheugen te detecteren. Zodra het defecte blok is geïdentificeerd, wordt een ander algoritme toegepast om te bepalen of de fout statisch of dynamisch is. Ten slotte worden meerdere korte algoritmen toegepast om het foutmodel te identificeren. De voorgestelde methodologie is gemakkelijk uit te breiden; nieuwe diagnostische mogelijkheden kunnen eenvoudig worden toegevoegd door nieuwe diagnostische algoritmen te integreren zonder dat de bestaande mogelijkheden beïnvloed worden. Bovendien is de methodologie platformonafhankelijk; het is niet afhankelijk van een specifieke geheugenimplementatie of architectuur.

CONTENTS

1

INTRODUCTION

The Fin Field-Effect Transistor (FinFET) technology became the most promising approach to enable the downscaling of technological nodes below the 20 nm *threshold. Nevertheless, this technology also brings forward behaviors that must be well understood to achieve the quality demanded by FinFET devices' applications. Therefore, effective yet cost-efficient test solutions are of great importance to ensure high-quality FinFET products. The main topics of this dissertation are the modeling of FinFET Static Random Access Memories (SRAM) faults and the development of high-quality test and diagnosis solutions for FinFET SRAMs. This chapter serves as a brief introduction to this dissertation. We start by introducing the FinFET technology. Second, we highlighting the role of VLSI test, its importance, and basic concepts. Then, we present the state of the art in both memory testing and memory diagnosis. Following, we explain the research topics explored throughout this Ph.D. project. Next, we present this dissertation's main contributions to advance the state-of-the-art in the field of FinFET SRAM testing and diagnosis. Finally, we detail the thesis organization.*

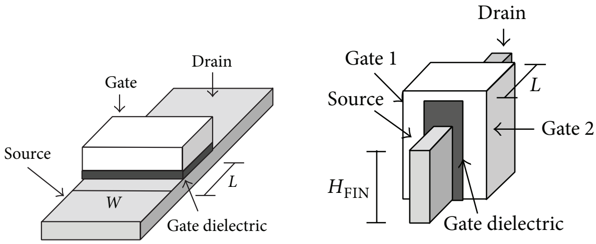
1.1. THE FINFET TECHNOLOGY

In the last few decades, the semiconductor industry has undergone tremendous improvements mainly attributed to the aggressive downscaling of transistors in *Integrated Circuits* (ICs); it is expected that the transistors integrated per unit area roughly double every two years [1]. Such a trend was first observed in 1965 by Gordon Moore [2] and became known as *Moore's Law*. While not an actual law but rather an observation or projection, it has become widely accepted and serves as a road map for the semiconductor industry due to its innovative and economic benefits. Two distinct types of transistor downscaling can be defined:

- **Conventional Scaling**: the transistor's dimensions are downscaled by a specific factor (typically $k = 0.7$). In *constant field scaling*, the supply voltage is also downscaled by the same factor. Voltage downscale is not always feasible as it might impact the device's performance, e.g., frequency, leakage. Thus, *constant voltage scaling* refers to only scaling down the device's dimensions [3].
- **Innovative Scaling**: new techniques and implementation strategies are used to enable further downscaling. It includes new enhancement techniques, e.g., channel strain engineering to improve the movement of electrons and holes in the channel [4], the introduction of new materials, e.g., using high-k materials as the gate-dielectric to reduce the gate leakage [5], [6], and finally, new transistor structures, e.g., *Silicon-on-Insulator* (SOI) [7] and *Fin Field-Effect Transistor* (FinFET) [8].

The continuous downscaling of devices posed a significant challenge, particularly below the 32 nm threshold. One of the most prominent challenges was related to short channel effects, i.e., the gate cannot completely control the channel due to the short distance between drain and source, resulting in higher sub-threshold leakage, threshold voltage (V_{th}) roll-off, and punch-through between the drain and source [9]. While some innovative scaling enabled the downscale down to 28 nm [10], [11], it became clear that a new device, i.e., a new physical structure, was necessary to push *Complementary Metal Oxide Semiconductor* (CMOS) devices towards the 20 nm threshold and lower. Thus, the FinFET device was proposed. The device aims to extend the gate's control over multiple sides of the channel by elevating the latter vertically into the gate, as shown in Fig. 1.1. This 3D structure reduces leakage and enables constant field downscaling even further.

In 2011, Intel reported, for the first time, the successful manufacturing of commercial 22 nm FinFET-based SRAM devices [13] – at that point, still coined as "tri-gate" CMOS devices. The following year, they launched their first processors (desktop and mobile) using the FinFET technology in their Ivy Bridge microarchitecture [14]. By 2014, all major chip manufacturers, e.g., GlobalFoundries, TSMC, Samsung, had started using the FinFET technology in their state-of-the-art products, e.g., 16 and 14 nm devices. Now, more than ten years since the release of the first-ever FinFET-based processor, the FinFET technology has undergone many innovative scalings. New materials have been used, e.g., germanium-based channels [15], and new structures have been proposed, e.g., Negative Capacitance FinFETs (NC-FinFETs) [16], *Gate-all-around* (GAA) nanowire FETs [17].



(a) Planar CMOS (b) FinFET

Figure 1.1: Structural comparison between (a) planar CMOS and (b) FinFET [12].

Both Samsung and TSCM have entered volume production of 5 nm[1] FinFET-based chips [20], [21]. As the semiconductor industry begins its quest towards even smaller FinFET devices, it is still uncertain to what extent the FinFET and the CMOS technology can be downscaled.

1.2. VLSI TEST PHILOSOPHY

This section introduces the VLSI test philosophy together with fundamental concepts and terminologies. It first defines the role of VLSI tests within the electronic testing scope. Then, it presents a classification of VLSI tests. Finally, the concepts of test escape and yield loss are introduced.

1.2.1. POSITION AND ROLE OF VLSI TESTS

Very large scale integration chips are essential in any modern electronic system. For example, consider a typical smartphone. It contains multiple chips that will carry out the many functions expected from such a device, e.g., voice and video encoding, internet connection, touch screen. The most critical chip in a smartphone – and any electronic system, in fact – is the system-on-chip (SoC). As the name suggests, an SoC combines multiple functionalities into a single silicon chip. Therefore, it contains a variety of modules to carry out its functionalities. An example of a state-of-the-art SoC is the A14, the first commercially available product manufactured on a 5 nm process node [22]. Released in 2020, the A14 contains a *central processing unit* (CPU), *neural processing unit* (NPU), *graphics processing unit* (GPU), 5G modem, on-chip memories, among other modules. The A14 integrates 11.8 billion transistors in a single chip of 88 mm, with a transistor density of 134 million transistors per mm^2 [23] using TSMC's 5 nm process. Naturally, fabricating such an intricate VLSI chip is a complicated and time-consuming process prone to manufacturing defects. Therefore, to guarantee the quality and reliability of semiconductor chips, it is essential to rigorously test them in different ways at different phases of their lifetime.

---

[1] At this point, referring technology nodes by their size has no relation to transistor's physical features anymore. It is only a commercial/marketing term to designate a new, improved technology node with increased transistor density, increased speed, and reduced power consumption [18], [19].

The manufacturer unquestionably carries out the most critical testing steps. They are responsible for defining the semiconductor chip's specifications and subsequently designing[2] and mass producing them. By the end of the manufacturing process, it is expected that the manufacturer has conducted various manufacturing tests to eliminate defective parts and guarantee that the chips being shipped to customers are indeed performing as designed for its life start, i.e., t=0, and with the demanded quality requirements. The quality of VLSI chips is evaluated using a metric called *defective part per million* (DPPM). For instance, ten DPPM means that statistically, ten parts out of one million parts are defective. The manufacturer's effort to test the chip varies significantly depending on the chip quality requirements demanded by the application, e.g., an electronic component used in a plush toy has much less severe demands than an electronic component used in the automotive or avionic fields. Thus, the testing procedure applied for the latter case is much more strict than for the former.

Besides guaranteeing the application's demands, the manufacturer also has financial motivations to deliver a high-quality chip. First, there is an exponential increase in the cost of detecting a defective chip after being integrated into increasingly more complicated systems. A widely accepted rule of thumb in test economics in the electronics industry is the rule of ten [24]. It suggests that if a defective chip is not detected by chip-level testing, finding it at the printed circuit board level costs ten times more than at the chip level. This cost factor continues to apply when the defective chip is incorporated into higher-level systems. Second, selling defective chips to customers and receiving them back harms the manufacturer's reputation and potential customers and income. In a worst-case scenario, a system failure due to a defective chip may lead to a catastrophic accident with loss of human lives, resulting in lawsuits and class actions.

The above emphasizes the importance of VLSI tests during the development and manufacturing phase, i.e., before t = 0, carried out by the manufacturer. There are two aspects of VLSI testing: *fault detection* and *fault diagnosis*. The role of fault detection is to detect whether something went wrong, while fault diagnosis is to determine what went wrong precisely and where the process needs to be altered. Both testing processes involve applying test patterns to the circuit and comparing the circuit's response with a precomputed expected response. If a product is designed, fabricated, tested, and fails the test, then there must be a particular cause for the failure [24]:

- The test procedure was incorrect or inappropriate;
- The manufacturing process was faulty or imprecise;
- The chip's design was incorrect;
- The chip's specification was inaccurate.

Therefore, the correctness and effectiveness of VLSI testing, i.e., fault detection and diagnosis, are of utmost importance to deliver high-quality products. Since testing carried out by the manufacturer, i.e., before t = 0, plays a critical role in determining the chip

---

[2] A semiconductor chip may be designed by a company and manufactured by another. It is a typical business model in the semiconductor industry to have fabless companies specializing in designing and manufacturing companies that focus solely on manufacturing process technology.

quality, it receives a substantial financial investment, which leads to research opportunities. Due to this reason, this thesis will be focused on this domain, i.e., fault detection and diagnosis throughout the development and manufacturing of semiconductor chips.

1.2.2. CLASSIFICATION OF VLSI TESTS

Various tests will be performed throughout developing a VLSI chip to determine what leads to a chip failure. Depending on the test objectives and the development stage of a VLSI chip, different test procedures are applied [24]:

1. **Characterization**: also known as design debug or verification testing. This form of testing is performed on any new design before sending it to production. Its primary objective is to verify the design's correctness and whether it meets its specifications, i.e., determine the exact limits of the device operating values. Functional tests alongside comprehensive AC and DC parametric measurements are performed at this stage to determine the limits of chip operation conditions such as supply voltage, temperature, and speed. Shmoo plots are used to identify the conditions in which the test has passed and failed. Secondary objectives include measuring chip characteristics for setting final specifications and determining specific details for the final production test program.

2. **Production**: every manufactured chip must undergo productive tests; it is a sign-off test right before being shipped out of the manufacturer. Its primary objective is to enforce the quality requirements by determining whether the chip under test meets all specifications. Production tests are based on go/no-go decisions. Therefore, they are less comprehensive than the previous characterization tests. Furthermore, fault diagnosis is not attempted. The tests at this stage may not cover all of the chip's functions, but they must guarantee a high coverage of modeled faults such that defective chips can be identified and eliminated out with high confidence. The time spent testing each chip must be brief to minimize equipment cost, as every chip must be tested. Therefore, a great deal of effort is focused on optimizing the testing routine to be fast and effective.

3. **Burn-In**: while production testing guarantees that the chip under test met design specifications at $t = 0$, it does not guarantee that they perform their functions as long as expected when getting to actual usage. Burn-in tests ensure the reliability of those chips by applying production tests once again, but this time under high temperatures and increased voltage supply continuously or periodically. This additional stress will force weak chips to fail at an accelerated speed. Burn-in tests can isolate two types of failures: infant mortality and freak failures. Infant mortalities are screened out by a short-term burn-in (10-20 hours) in standard or slightly stressed conditions; they are often caused by weak defects, *process variation* (PV), or a combination of both. Freak failures are devices with the exact failure mechanisms as reliable devices but require long burn-in time (100-1000 hours) in stressed conditions. Compared to production tests, burn-in tests are much more expensive and time-consuming. Therefore, a manufacturer must consider economics and make a trade-off between test overheads and chip reliability depending on the target application.

4. **Incoming Inspections**: these tests are performed on purchased semiconductor chips right before integrating them into a system. The primary goal of incoming inspection tests is to avoid placing a defective chip in a system, where the cost of testing and diagnosis may far exceed the cost of incoming inspection. Compared to previous manufacturing tests, e.g., production, incoming inspections are performed with much less effort and time; usually, only a certain number of selected samples of purchased VLSI chips undergo incoming inspection; this amount depends on the chip quality and system requirements. However, this practice is gradually disappearing, as companies nowadays expect the received chips to be high-quality and are often pressured by time to market requirements.

1.2.3. TEST ESCAPES AND YIELD LOSS

During the manufacturing process of semiconductor chips, they may be affected by physical defects. Thus, by the end of manufacturing, a chip can be either **OK** or **not OK** ($\overline{\textbf{OK}}$). After manufacturing, a production test, i.e., a short and go/no-go decision-making process, is applied to every manufactured chip. There are two possible outcomes of this test: the chip can either **pass** the production test and is afterward shipped to the customer, or it can fail the test and is discarded or used for yield learning and design improvement. Naturally, it is expected that all **OK** chips will **pass**, and all $\overline{\textbf{OK}}$ chips will **fail**. However, due to inaccurate test procedures, other outcomes may arise:

① **OK**, **pass**: chips that have passed the test and are indeed defect-free.

② $\overline{\textbf{OK}}$, **pass**: chips that have passed the test but are defective, i.e., *test escapes*.

③ **OK**, **fail**: chips that have failed the test but are defect-free, i.e., *yield loss*.

④ $\overline{\textbf{OK}}$, **fail**: chips that have failed the test and are indeed defective.

Set ② contains defective chips that have passed the test; these are known as test escapes. They will be delivered to customers, along with the defect-free chips in set ①. A defective chip passes a production test because the said test was not developed to detect the fault caused by the defect affecting the chip. In other words, the test program was not complete and did not consider all the defects that could have emerged from the manufacturing process. If not detected by incoming inspections, these defective chips will be mounted onto electronic systems and subsequently distributed to the market. The may lead to user complaints, and in the worst case, to accidents and loss of human lives. Some of these defective chips will be sent back to their manufacturer; these chips are known as *customer returns*. Manufacturers use customer returns to diagnose and understand failure roots and improve the chip design and manufacturing process. Furthermore, customer returns have a significant influence on the business-to-business relationship and may damage the reputation of established chip manufacturers.

Set ③ contains the chips that have somehow failed the production test; these are known as yield loss. They are due to excessively rigorous tests that will also cause defect-free chips to fail, e.g., an I_{DDQ} test with inappropriate margins may overkill some defect-free chips by mistakenly identifying the increased leakage current due to PV as defects.

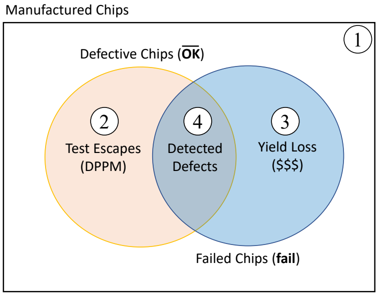


Figure 1.2: Relation between test escapes and yield loss.

This set of chips directly increase the average cost of manufacturing a chip, as defect-free chips that could be sold to customers are being tossed away. Additionally, rejecting good chips also indicates that the test program needs to be adjusted so that this set is minimized as much as possible in future batches.

All chips in sets ③, ④, alongside the customer returns in ②, have not met design specifications and thus should be directed to failure analysis. Investigating and understanding the failure mechanisms of chips is essential for the yield learning process, i.e., fixing and improving the design and manufacturing process to maximize the number of chips belonging to ①. Fig. 1.2 shows the relation between all sets, illustrated in a Venn diagram. Sets ② and ③ are mainly due to inappropriate test programs. Thus, performing failure analysis on these chips will help identify failures that are not covered by the test program and subsequently enhance it to cover these failures. From an economic point of view, the two circles, i.e., $\overline{\textbf{OK}}$ and **fail**, must be as closely overlapped as possible to reduce test escapes and yield loss. Since these lease to higher-quality devices and a cheaper manufacturing process, identifying the real defective chips is the unchanging goal of R&D investment in VLSI tests.

1.3. THE STATE OF THE ART IN MEMORY TESTING

The requirements involved in testing combinational (i.e., logic) and sequential (i.e., memory) circuits are very different. When testing combinational circuits, the test program applies one or two test vectors to detect each fault. On the other hand, when targeting faults in sequential circuits, the test program may be required first to bring the circuit into a state in which the fault may be sensitized and observed. Nevertheless, a memory contains millions (if not billions) of states, and so exhaustively testing all possible states in a memory chip is an impossible task. Therefore, dedicated and specific memory testing techniques have to be developed and applied to guarantee the quality of memory chips.

Memory testing has gone through a long development process. Before the 1980s, test procedures applied to memories were considered *ad-hoc* due to their absence of formal fault models and proofs [25]. These tests have a very long test time (in the order of $O(n^2)$, where n is the number of bits in a memory) for relatively small fault coverage. Examples of ad-hoc tests are the Zero-One test, GALPAT test, and Walking 1/0 test [25], [26].

In the early 1980s, memories grew exponentially in size, i.e., capacity. Thus, to reduce the test's time and cost per memory chip, test development focused on investigating the possible faults that can occur in the memory. This led to the introduction of many fault models. Their main advantages are that they enable a formal fault coverage proof, and they significantly reduce the test time to the order of $O(n)$, i.e., linear with the size of the memory. Some key functional fault models introduced at that point were the *Stuck-At Fault* (SAF), the *Address decoder Fault* (AF) [27], the *Coupling Fault* (CF) [28]–[30], and the *Neighborhood Pattern Sensitive Fault* (NPSF) model [31], [32]. March tests became the dominant type of tests for SAFs, AFs, and CFs [26], [29], [33]; while special linear tests were designed for NPSFs [26], [34].

The above functional fault models were abstractions based on the memory's behavior rather than on actual memory designs and defects. The lack of connection with real memory chips led to the introduction of *Inductive Fault Analysis* (IFA) [35], [36]. IFA is a systematic procedure to predict the faults in an integrated circuit by injecting spot defects in a simulated circuit netlist; this enables the development of fault models based on simulated defects in actual memory designs. The introduction of IFA led to a variety of new functional fault models [33], such as *State Coupling Fault* (CF_{st}), *Data Retention Fault* (DRF), and *Stuck Open Fault* (SOF).

In the early 1990s, memories experienced an extraordinary increase in size; consequently, test programs with linear time became less and less acceptable. Furthermore, the increased use of embedded memories turned testing even more complex by limiting the controllability of inputs and the observability of outputs. *Built-in-self-test* (BIST) was the proposed solution to overcome this problem [37]–[41]. BIST solves the controllability issues of embedded memories and alleviates test requirements regarding test speed and the number of input and output (I/O) pins. An additional advantage of BIST is its at-speed testing, i.e., at the maximal clock period, allowing for a higher fault coverage, especially for faults with complex timing characteristics. An additional technique used to improve memory testing was *Design-for-Testability* (DFT) circuits, which are auxiliary circuits introduced into the memory to reduce test time and enable active tests for cell stability faults such as DRFs [42].

All test solutions previously discussed were *functional* solutions, i.e., they aim at observing faults by checking for incorrect functionalities, such as an unexpected output. *Parametric* memory test solutions, which rely on measuring the memory's parameter and identifying out-of-the-ordinary measurements, have also been proposed. A widely used parametric test is the $\mathrm{I_{DDQ}}$ test, which measures the quiescent power supply current while the memory is idle [43], [44] For example, if there is a short connection in the memory, the current may be higher than usual, and a discrepancy will be detected, thus enabling the detection of defects that do not cause any functional impact and cannot be detected with functional test solutions.

In the late 1990s, experimental results based on DPPM screening of many tests applied to a large number of memory chips indicated that the existing fault models were not sufficient to explain many of the detected faults [45], [46], indicating the existence of additional fault models not yet covered. Therefore, there was a need to find innovative fault modeling techniques. This stimulated the introduction of a new fault modeling approach based on linear resistor defect injection and SPICE simulation [47], [48]. The new approach led to the establishment of faults models still used in today's test solutions, such as read destructive faults, write disturb faults, transition coupling faults, and read destructive coupling faults.

As the CMOS technology scaled down to more advanced nodes, the industry has kept the same approach of modeling manufacturing defects through resistances. Nevertheless, these advanced nodes introduce new materials, fabrication steps, and failure mechanisms. It became widely recognized that a growing number of defects and the increased variability in the device's characteristics represent a significant challenge to the overall quality and reliability of the system, especially when considering high-quality levels, e.g., in the range of defective parts per billion (DPPB) [3]. Furthermore, it is known that faults in these chips are dominated by transient, intermittent, and weak faults rather than hard and permanent faults [49] As the existing approaches only target resistive defects (e.g., opens and bridges) at the terminals and interconnect of devices, its effectiveness in detecting defects in deep-scaled technologies such as FinFET remains uncertain.

1.4. THE STATE OF THE ART IN MEMORY DIAGNOSIS

Memory fault diagnosis has not been given as much interest as the testing of memory chips. However, with the continuous technology downscaling and the stricter quality requirements imposed by customers, it became necessary to understand new fault mechanisms. This led to the introduction of the first diagnostic methodologies in the early 1990s. These first approaches were probability-based fault analysis methodologies [50], [51]. By performing many random experiments, fault sites could be narrowed down by appropriately overlapping the faulty areas and distinguishing them by comparing the pass/fail data with statistically generated fault probabilities. These methods are not deterministic, have a low fault coverage, and have a significantly long test time (in the order of $O(n^2)$. Thus, there was still a need to improve the diagnosis methodologies applied to memory chips.

This improvement came from using signatures for specific sets of fault models. A signature-based diagnosis methodology applies diagnostic tests and uses the results to identify the fault; it provides a unique signature for each targeted fault. When applying the diagnostic test on the memory, all the failed read operations are recorded alongside failing memory cells' addresses. Signatures are generated based on the fail status of each operation and are then grouped into a fault dictionary. Faults are then diagnosed by distinguishing faults with unique signatures. Signature-based diagnosis schemes are, up to this day, the most efficient and most well-received manner to diagnose memory faults. Many solutions using this methodology were published in the 2000s [52]–[56], which enabled the diagnosis of static faults in the memory cell array. Some of these solutions also included the use of *Design-for-Diagnosis* circuits, i.e., additional hardware introduced into the memory with the sole purpose of facilitating diagnostic tests. More recently, the diagnosis of dynamic faults in the memory array also became a research topic [57].

However, existing solutions based on signatures also have some drawbacks. These solutions have a pre-defined dictionary; thus, their signatures are hardwired to a predefined diagnostic test. Consequently, if the memory is affected by a fault not considered in the fault dictionary, the diagnosis phase fails to provide any result or may even provide a wrong response. Any modification to the set of targeted faults needs a new diagnostic test with a new set of fault signatures. This increased complexity of march tests can be excessive if used for industrial purposes. Furthermore, they also assume knowledge of every read operation's pass/fail status of a diagnostic test, which is not always available in testing equipment [56]. Finally, most solutions only focus on memory array cell faults, even though a fault can occur in any part of the memory system, e.g., address decoders, cells, sense amplifiers, write drivers. As the existing approaches' scope is limited, it becomes impossible to determine which memory component is defective; such information is critical for guiding the designers on improving the design and the manufacturing process. Thus, identifying which memory block is defective leads to considerable time-saving during the yield ramp-up phase. While some signature-based solutions proposed to diagnose other parts of the memory chip [58], [59], they still suffer from limitations such as limited fault scope and extensibility. Therefore, an appropriate methodology to diagnose memory chips that overcome the limitations of signature-based techniques is still missing.

1.5. RESEARCH TOPICS

The performed research in this thesis focuses on three main areas:

1. Fault Modeling
2. Test Development
3. Memory Diagnosis

Each of these topics is explained in further detail next.

1.5.1. FAULT MODELING

As the name suggests, fault modeling consists of observing, analyzing, and modeling faulty memory behavior to generate fault models. These fault models are used to develop test solutions later in the testing process. Therefore, developing accurate and realistic fault models that capture a memory cell's faulty behavior in the presence of a defect is the key to high-quality tests. To this end, the following four topics are explored in this thesis.

1) Complete fault space: it is necessary to investigate whether existing fault models developed from planar CMOS apply to FinFET SRAMs, considering all the components in a memory design and possible unique faults in FinFET SRAMs. In other words, the complete fault space dedicated to FinFET SRAMs has to be defined; it should contain all possible faults that may occur in FinFET SRAM chips.

2) Fault analysis procedure: based on the circuit simulation platform, a sound fault analysis validation procedure must be developed. This procedure must include defect injection, i.e., inserting defect models into the memory, and stimuli generation, i.e., generating operation sequences, to verify the faulty behavior and validate the defined fault space.

3) Circuit simulation platform: typically, fault modeling is performed by SPICE-based circuit simulations. Therefore, a practical circuit simulation platform has to be built. It must include a complete SRAM design with the central cell array and all necessary peripherals, such as decoders, write drivers, and sense amplifiers. Write and read functions should be verified.

4) Accurate and realistic faults: finally, the above procedure must be applied to all injected defects to obtain accurate and realistic faults. This step aims to clearly distinguish what could theoretically occur in the memory and what indeed occurs in defective memories.

1.5.2. TEST DEVELOPMENT

Test development consists of developing strategies targeting the faults identified during the fault modeling procedure. In this step, the following three topics are explored to generate optimal test solutions for FinFET SRAMs.

1) Appropriate March algorithms: March tests are the most commonly used test solutions for memory testing. We aim to develop March algorithms that cover all the previously-observed fault models in FinFET SRAMs. Optimization of the tests by applying appropriate stressing conditions for efficient test time will also be explored.

2) DFT and BIST solutions: March solutions are not always able to provide satisfactory test results. For example, they may be too long or have inadequate or insufficient fault coverage. In this phase, we will address the issue of how to improve the test procedure by introducing special hardware into the memory to increase the fault coverage and reduce test time.

3) Validation of test solutions: this phase aims at validating and evaluating the proposed solutions. Experiments will be conducted in the form of simulations to quantify the gains provided by the additional hardware circuits. Furthermore, variation analysis will assess whether the additional testing circuits lead to yield loss.

1.5.3. MEMORY DIAGNOSIS

Production tests provide a simple pass/fail result. During the development of a chip, more information regarding failure mechanisms is often necessary. Thus, diagnosis procedures are employed to understand better what must be improved in the chip's design and manufacturing process.

1) Diagnose Methodology: diagnosis is a complex procedure; it must be able to accurately identify fault patterns and make the appropriate correlation with faults in the logic level and defects in the physical level. An incomplete or improper methodology will lead to inaccurate reports, significantly impacting development costs and time. Therefore, appropriate methodologies must be developed.

2) Diagnostic algorithms: diagnostic algorithms have very different goals than testing algorithms. Unlike the latter, diagnostic algorithms do not focus on detecting the most faults in the shortest time. Instead, they aim to detect specific sets of faults and mask others. Applying multiple diagnostic algorithms provides a clear picture of all the faults occurring in the memory chip. Thus, a more in-depth analysis must be performed when developing diagnostic algorithms.

3) Validation of the diagnose solution: finally, the methodology and the developed algorithms must be validated. Experiments will be conducted in the form of simulations with injected defects in all parts of the memory. The methodology is validated by assessing whether the generated reports provide accurate details regarding the faults caused by the injected defects.

1.6. CONTRIBUTIONS OF THE THESIS

Over the entire course of this PhD project, we have addressed research issues at all three phases of test development, as presented in the previous section. The main contributions of this thesis can be summarized into five items as follows.

1. ***A Survey on different types of FinFET SRAMs and their manufacturing process and failure mechanisms.*** We first surveyed the manufacturing process of FinFET devices in the literature, and defined four critical steps: the substrate foundation, shaping the fin structure, placing the gate on top of the fin, and forming the source and drain contact. We have also investigated the failure mechanisms in each of these steps, how they affect the final physical structure of the FinFET device, and how they impact the device's functionality. Finally, we investigated and classified the different types of FinFET SRAMs based on their topology, e.g., layout, number of fins, types of FinFET device employed. This survey is yet unpublished and is being prepared for submission as a review paper.

2. ***The definition and validation of the complete FinFET SRAM fault space [60].*** A fault space is the set of all possible faults that can occur in a given circuit; it is complete if it contains *all* faults. We have explored the fault space of FinFET SRAMs and have defined a complete set of functional and parametric faults. Functional faults may impact the memory's logic level, such as an incorrect output. On the other hand, parametric faults only impact the memory's electric level; this can translate to electric deviations in the memory's parameters. We have proposed a fault space validation methodology consisting of six steps: i) netlist generation, ii) defect injection, iii) sweep defect size, iv) stimulus generation, v) circuit simulation, and vi) behavior inspection. We then apply this methodology using SPICE simulations and defect injection to validate the fault space and confirm which faults can indeed occur in FinFET SRAM chips.

3. ***The development of test strategies to improve the detection of functional HTD faults in FinFET SRAMs.*** Such faults may have a functional impact, e.g., an incorrect output. Test solutions targeting functional HTD faults aim at improving the likelihood of triggering such faults and thus enable the detection of these faults. Thus, they must apply additional stress to push the memory to operate in severe conditions. We have proposed several test solutions, namely solutions that focus on the detection of random read faults [61], [62] and undefined state faults [63]. These solutions have used appropriate algorithm-related stressing conditions, i.e., a thought-out sequence of write and read operations, and additional hardware inserted into the memory, i.e., DFT circuits, that apply additional environment-related stressing conditions.

4. ***The development of a test strategy based on monitoring schemes to improve the detection of parametric HTD faults in FinFET SRAMs.*** These faults lead to severe parametric deviations in the memory, such as increased leakage current and reduced noise margins. Therefore, they do not cause any functional behavior. Consequently, functional test solutions cannot detect parametric faults. In order to detect these faults, monitoring schemes must be employed. They aim at monitoring a specific parameter, e.g., current flow, bit line swing, and identify when the monitored parameter is outside the expected bounds of operation. We have proposed two solutions, one aiming at detecting increased power consumption [64], and another aiming at detecting a reduced bit line swing [65]. Both employ on-chip analog sensors to monitor memory parameters. A neighborhood comparison logic is then used to compare the measured parameters from different parts of the memory and identify discrepancies

5. ***The development of a new hierarchical diagnosis methodology for embedded memories, including FinFET SRAMs*** [66], [67]. This methodology aims at speeding up the fault localization process during diagnosis by dividing the memory into five functional blocks: the row decoder, the column decoder, the read path, the write path, and the memory array. A 3-step hierarchical framework is then applied to obtain more information regarding the fault: i) its location, i.e., the faulty block, ii) its nature, i.e., static or dynamic, and iii) its model. The hierarchical memory diagnosis methodology covers fault in the entire memory chip and has a vast fault space as it includes static and dynamic faults from all parts of the memory. The methodology is easily extensible; new diagnostic capabilities can be easily added by integrating new diagnostic algorithms without recompiling existing diagnosis signatures. Furthermore, it is platform-independent; it does not rely on a specific memory implementation or architecture, and it does not require any specific test equipment.

1.7. THESIS ORGANIZATION

The contributions mentioned above advancing the state of the art in FinFET SRAM testing and diagnosis will be elaborated in detail in the remainder of this thesis, organized as follows.

Chapter 2 introduces the FinFET SRAM technology and its models. First, we introduce the fundamentals related to FinFET devices, including their electrical equations, their physical structure, and key physical parameters. We then investigate the manufacturing process of FinFET devices. The overall process is separated into four main phases. For each phase, we describe the manufacturing steps required to fabricate the processed structure. We also discuss the potential failure mechanisms that could arise from these manufacturing steps. Then, we introduce the background on FinFET SRAMs; we start by describing the classic SRAM model, covering different abstraction levels. We then discuss the advantages of FinFET SRAM cells over traditional planar CMOS SRAMs, followed by classifying the different types of typologies for FinFET SRAMs. We conclude this chapter by discussing the outlook for the FinFET technology.

Chapter 3 presents the fault modelling for FinFET SRAMs. We start by introducing a classification of faults based on their impact on the memory and their sensitization conditions, e.g., functional vs. parametric faults, static vs. dynamic faults, single-cell vs. coupling faults. We then define the fault space for FinFET SRAMs. The fault space is first defined from static faults coming from the memory array, decoders, write, and read paths. Then, the same is performed for dynamic faults. After, we introduce a fault space validation methodology. Finally, the defined fault space for memory array FinFET SRAM faults is validated using the proposed validation methodology. The simulation setup is detailed, baseline metrics are determined, and the results are laid out in defect size ranges in which faults are observed.

Chapter 4 presents the proposed solutions to test FinFET SRAMs. We first classify the memory array faults based on their detection conditions, i.e., hard-to-detect and easy-to-detect faults. We then present a framework for test development; it includes a discussion on test target and fault observation and identification methods. Furthermore, it also discusses the different stressing conditions that can be applied to the memory circuit. We briefly discuss the existing test solutions and their limitations. Following, we present the proposed test solutions. We start by presenting the test solutions to improve the detection of functional HTD faults; these include algorithms and DFT circuits. Then, we present the DFT circuits proposed to improve the coverage of parametric HTD faults. Finally, we discuss the outlook for test solutions and what still needs to be improved to push the quality of FinFET SRAMs even further.

Chapter 5 introduces a new hierarchical memory diagnosis methodology for embedded memories, including FinFET SRAMs. We define the key differences between detection and diagnosis and classify the existing diagnosis schemes. We then present the diagnosis approach: its methodology, fault space, and three levels of diagnosis. We apply the proposed in various case studies, proving the approach's capabilities to diagnose both easy-to-detect and hard-to-detect faults in all parts of the memory chips. We conclude this chapter by discussing the proposed approach and comparing it with other existing diagnosis solutions.

Finally, Chapter 6 concludes this thesis by providing a summary and an outlook to future research directions.

2

THE FINFET SRAM TECHNOLOGY & MODELS

The FinFET technology has undergone a long evolution since its first experimental manufacturing in 1989. Since then, significant breakthroughs have been made in device design, circuit layout, and manufacturing. Thanks to technology advancements in these fields, FinFETs have been commercialized by several semiconductor companies for over a decade.

The primary purpose of this dissertation is to develop realistic fault models and efficient tests and diagnoses for FinFET SRAMs. It is, therefore, of interest to understand in detail the structure of such devices and memories and the way they operate. This chapter describes the FinFET SRAM technology and models. First, we introduce the FinFET's fundamentals, including its working principles and critical physical parameters. Second, we examine the FinFET manufacturing process and the associated defects in each step. Additionally, we classify the different types of transistors based on their manufacturing steps. Third, we discuss the application of FinFETs to design SRAMs; we introduce the different memory structure models, the advantages of using FinFETs over planar CMOS, and the different SRAMs that can be designed using FinFETs. Finally, we discuss the FinFET technology outlook; we explore the upcoming challenges in technology miniaturization and examine how the industry can use multi-gate devices to tackle these challenges.

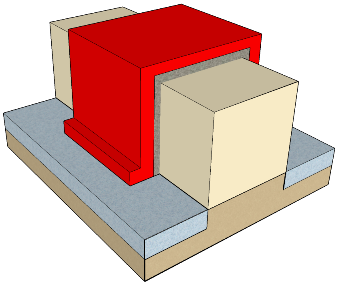


Figure 2.1: General structure of a FinFET.

2.1. FINFET FUNDAMENTALS

FinFETs are quasi-planar, multi-gate devices consisting of vertical silicon stripes, i.e., “fins”, wrapped by a gate structure. A general representation of this structure is depicted in Fig. 2.1, which shows the gate structure (red material) wrapping the fin by all three sides. The first design similar to FinFETs was the DELTA (Fully-Depleted Lean Channel Transistor) [68], a double-gate MOS transistor proposed in 1989. Afterward, other transistor structures were proposed aiming to surpass the scalability limitations of the CMOS technology [69]. Due to its superior electrical properties, the FinFET technology became the most promising approach to continue CMOS scaling and was consequently adopted by major semiconductor companies.

In nanometer CMOS technologies, the gate cannot fully control the channel due to the short distance between drain and source, resulting in the well-known *Short Channel Effects* (SCEs): higher sub-threshold leakage, threshold voltage (V_{th}) roll-off, and punch-through between the drain and source [9]. In planar technologies, two approaches are adopted to reduce SCE. First, the oxide thickness is thinned out to increase the gate-to-channel capacitance. However, this approach is limited as downscaling the oxide can trigger direct tunneling current or cause reliability issues such as *Time Dependent Dielectric Breakdown* (TDDB). Second, it is possible to engineer the channel doping and create a larger potential barrier between source and drain, reducing the charge sharing between the two terminals. However, high channel doping reduces carrier mobility, increases *Gate Induced Drain Leakage* (GIDL), and causes deviations in the V_{th} of devices due to *Random Dopant Fluctuation*. FinFETs have intrinsic superior channel control due to their multi-gate structure, effectively reducing SCE and eliminating the need for high doping devices. Consequently, the absence of doping on FinFET’s body suppresses RDF, resulting in a more homogeneous V_{th} among different transistors of the same circuit [70].

The most significant technological parameters of a FinFET are its fin’s height (H_{FIN}) and width (T_{FIN}), its gate (and channel) length (L_g), and the number of fins (N_{FIN}). Other parameters, such as gate oxide thickness (T_{OX}), buried oxide thickness (T_{BOX}), body doping, gate/source doping, gate workfunction (Φ_G), among others, complete the typical technological parameters of a FinFET [71]. As the gate of a FinFET device is wrapped around its silicon fin, up to three distinct channel sides can be created: two on the side and one on the top; an essential characteristic of this transistor is that current flows from

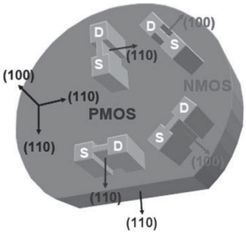


Figure 2.2: FinFET transistors with contrasting channel orientations [73].

all its sides. Thus, the effective channel width of a FinFET can be defined by Eq. 2.1, where N_{FIN} is the number of fins on the transistor. Once the height and width of a fin are fixed for a specific design, the only way to change W_{eff} and increase the channel's current drive capability is by using more fins.

$$W_{\text{eff}} = (2 \times H_{\text{FIN}} + T_{\text{FIN}}) \times N_{\text{FIN}} \tag{2.1}$$

Besides its intrinsic structural advantages, FinFETs also benefit from its manufacturing process aspects. Besides its compatibility with the CMOS manufacturing process, FinFETs can also be engineered to improve the mobility of electrons and holes using different channel orientations. Electrons have the highest mobility along the ⟨100⟩ plane, while holes mobility is the highest along the ⟨110⟩ plane [9]. Thus, NMOS fabricated in the ⟨100⟩ plane and PMOS fabricated in the ⟨110⟩ plane present superior characteristics to its counterparts fabricated in other planes. Nevertheless, fabrication of planar devices in different planes other than the ⟨100⟩ is considered difficult due to increased PV and interface traps [72]. However, manufacturing FinFET devices in contrasting channel orientations can be achieved by rotating devices 45° based on the wafer's orientation. Fig. 2.2 [73] depicts this procedure on a ⟨110⟩ wafer: PMOS devices are manufactured parallel or perpendicular to the wafer's notch, while NMOS devices are tilted 45° to the wafer's notch. By manufacturing FinFET devices in these specific planes, designers can significantly improve the carrier mobility of these devices. Fig. 2.3 [72] shows the I_{ds} vs V_{ds} curve for devices designed in different channel orientations. For NMOS devices, an improvement of 12% in I_{ds} is observed in devices fabricated in the ⟨100⟩ plane compared to ⟨110⟩. For PMOS, devices manufactured along the ⟨110⟩ plane exhibit 18% improved I_{ds} compared to devices in the ⟨100⟩ plane.

2.2. FinFET Manufacturing Process & Defects

This section discusses the manufacturing process of FinFETs. Even though the manufacturing process of FinFETs is compatible with the process used in planar devices, new unique steps are still necessary. Furthermore, these unique steps can create new FinFET-specific defects in the structure of devices, i.e., unintended differences from the intended design [26]. Therefore, it is crucial to understand the FinFET manufacturing process and its unique defects to develop accurate defect models that correctly represent all the faults in a FinFET-based circuit.

The manufacturing process of FinFET devices can be divided into four different stages

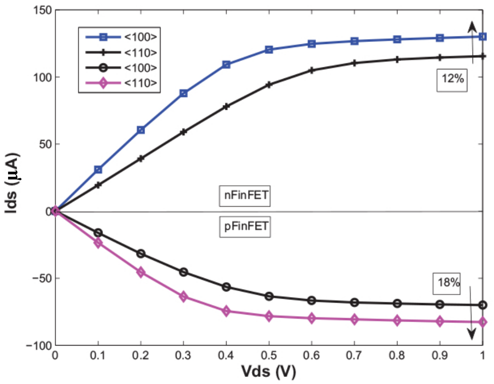


Figure 2.3: I_{ds} vs. V_{ds} characteristics of FinFET transistors with different channel orientations [72].

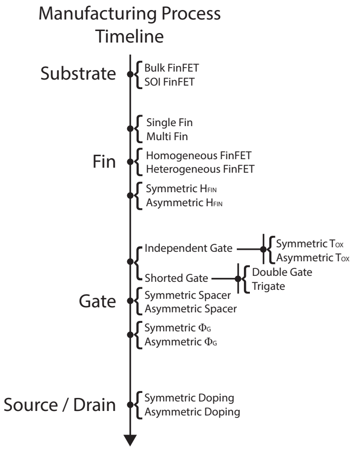
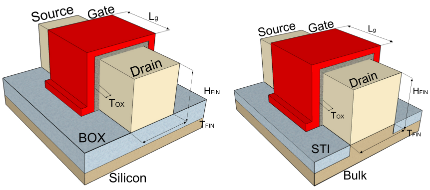


Figure 2.4: Types of FinFET transistors, divided by order of manufacturing.

based on the structure being fabricated: Substrate Deposition, Fin Patterning, Gate Stacking, and *Source/Drain* (S/D) Formation. Accordingly, it is also possible to classify different FinFET structures based on this distinction, as depicted in Fig. 2.4. In summary, the resulting FinFET device is the combination of each bullet point in the manufacturing process timeline. Each of these stages and the defects that may occur during the processing steps are discussed next.

2.2.1. STEP 1: SUBSTRATE

One of the essential characteristics of a FinFET concerns its wafer substrate. There are two types of substrates in FinFET devices: **bulk** and ***silicon on insulator*** **(SOI)**; they are illustrated in Fig. 2.5. Their only difference is the connection between fins and substrate. Because bulk devices do not have such a connection, they present a better substrate heat



(a) (b)

Figure 2.5: Structure and main technology parameters of FinFET transistors: (a) SOI FinFETs and (b) Bulk FinFETs.

transfer than SOI devices. Initial research proposed FinFETs using the concept of *Fully-Depleted* (FD)-SOI [68] owing to its less complex manufacturing process, more suitable to academic settings. At that point in time, university equipment could not provide a satisfactory oxide etching precision, an essential step in fabricating the fins of bulk FinFETs. Later on, with the development of research in FinFETs and the industry engagement, FinFETs in bulk structures became feasible.

MANUFACTURING STEPS

An SOI substrate consists of a *buried oxide* (BOX) layer on top of the silicon. Fins are built on top of the BOX and are physically isolated from each other and the substrate. The manufacturing process of SOI FinFETs is similar to planar SOI structures, the main difference being the thickness of the outer silicon layer and buried oxide. In planar FD-SOI devices, thin layers of silicon and oxide are used to create a *Ultra Thin Body and Buried Oxide* (UTBB) FD-SOI. For FinFET devices, thicker layers of Si and oxide are required to create the fin's three-dimensional structure and the BOX, respectively. The surface of the BOX is polished via *Chemical Mechanical Planarization* (CMP) to remove the oxide excess and planarize the surface.

Contrarily, bulk FinFETs do not require any manufacturing step; FinFET devices are built directly on top of the wafer. Therefore, bulk devices have lower wafer costs. Nevertheless, these devices also require additional manufacturing steps and lithography masks, which increases cycle time and overall cost [9]. They also demand a more controlled manufacturing process, especially to guarantee a uniform H_{FIN} within the entire wafer. Thus, the overall cost of producing Bulk devices is higher than producing SOI FinFETs; yet, both Bulk and SOI FinFETs are comparable in performance, cost, and yield [12]. Nevertheless, when analyzing the FinFET structures embraced by major foundries (e.g., Intel, Samsung, TSMC), it is clear that the bulk structure is favored. For commercial products below the 7 nm threshold, only bulk FinFETs are available [74].

MANUFACTURING DEFECTS

Manufacturing defects in the substrate are related to defects in either the Si wafer or the BOX of SOI devices. Since Si wafers have also been used for planar devices, no unique defects have emerged with the FinFET technology. However, a new defect mechanism affecting the buried oxide of SOI FinFETs has emerged, named as **SOI FinFET BOX Breakdown** [75]. Chemicals used for post-CMP processing can interact with the substrate. In

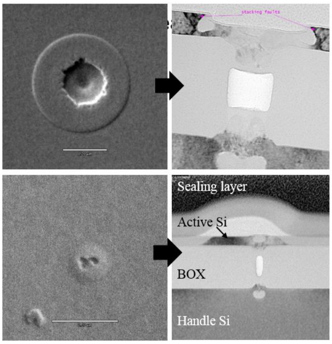


Figure 2.6: Top-down (left), and cross-section (right) images of BOX breakdown defects [75].

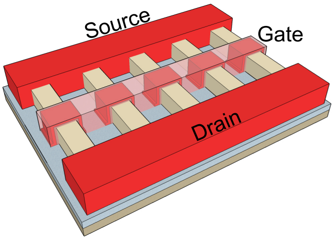


Figure 2.7: A five-fin FinFET.

severe cases, this interaction can generate an electric discharge through the BOX, causing a breakdown. This breakdown may damage the substrate resulting in dimples and craters, as shown in Fig. 2.6 [75]. This defect mechanism can be alleviated or eliminated by optimizing the post polish clean chemistry [75].

2.2.2. STEP 2: FIN

As previously discussed, the fins dimension dictates a FinFET's effective channel width. Typically, all devices in the circuit are designed (and manufactured) with the same L_g, T_{FIN}, and H_{FIN}. Therefore, the only way to increase a FinFET's drive (i.e., its I_{on}) is by adding more fins to its structure, i.e., **single-fin** FinFETs or **multi-fin** FinFETs. A multi-fin FinFET consisting of five fins (i.e., $N_{\text{FIN}} = 5$) is depicted in Fig. 2.7. The FinFET's I_{on} is directly related to its W_{eff} and linearly proportional to the number of fins used to design the transistor.

The performance of a FinFET device can be improved by modifying its channel material, leading to two distinct types of FinFETs: **homogeneous** and **heterogeneous** FinFETs. The first refers to the classical CMOS concept in which the same material (usually Si) is used for both the substrate and channel, while the second alludes to FinFET transistors on Si substrate but with materials like Ge or III-V compounds as channels. The primary motivation behind using such materials is their improved carrier mobility. Table 2.1 lists the electrical properties of Si, Ge, and some of the most popularly adopted compounds in semiconductors [76]–[79]. Because of their inherent superior mobility

Table 2.1: Properties of materials and compounds used in semiconductors.

	Si	Ge	GaAs	InAs	InSb
Electron Mobility $[\text{cm}^2\text{V}^{-1}\text{s}^{-1}]$	1600	3900	9200	40000	77000
Hole mobility $[\text{cm}^2\text{V}^{-1}\text{s}^{-1}]$	430	1900	400	500	850
Band Gap [eV]	1.12	0.66	1.42	0.36	0.14

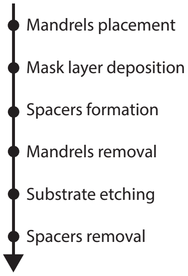


Figure 2.8: Fin patterning process flow of FinFETs on SOI wafer.

properties, heterogeneous FinFETs show higher levels of performance and lower power consumption at low operating voltage [80]. Ge p-channel FinFET devices, such as the ones proposed by TSCM [15], have shown excellent subthreshold characteristics and adequate SCE control. Therefore, they are a valid option to improve the performance of FinFETs. On the other hand, they are also more complex and expensive to manufacture because of structural integration with Si and epitaxial growth steps [81].

Finally, fin asymmetry within a single chip has also been proposed. As mentioned earlier, all fins are typically manufactured with the same physical structure. However, it has been proposed to manufacture fins with disparate heights [82], leading to **symmetric** and **asymmetric** H_{FIN}. The performance of PMOS FinFETs can be improved by increasing H_{FIN}, as they have inherently lower performance than NMOS FinFETs owing to their smaller carrier mobility. However, manufacturing such asymmetrical devices requires additional masks and etching steps, which may increase process variations and the overall fabrication costs.

Manufacturing Steps

The fins are the first structure to be built. They can be manufactured on top of SOI or bulk wafers, directly impacting the manufacturing flow; building the former is more straightforward than the latter as it involves fewer steps and requires less precision. A simplified version of the fin patterning process on SOI wafer [9] is illustrated in Fig. 2.8. In summary, it consists of manufacturing spacers and subsequently partially removing them to etch the fins. A detailed manufacturing flow showing the produced physical structure is depicted in Fig. 2.9.

At the start of fin patterning, the SOI wafer comprises the Si layer, the BOX layer, and a second Si layer on top of the oxide. The thickness of this outer Si layer determines the device's H_{FIN}. The first step of fin patterning on SOI wafers is placing the mandrels, which are structures used to generate the spacers. Designers can adjust the width of mandrels

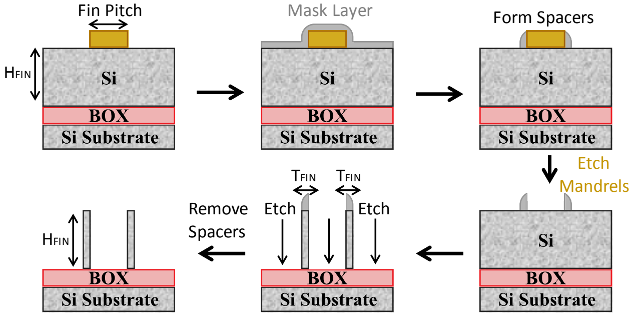


Figure 2.9: Manufacturing process of fins on an SOI FinFET.

to engineer the distance between fins on a transistor, known as the fin pitch. However, inconsistencies in the width of adjacent mandrels (and thus fin pitch) can impact the transistor's behavior as it further augments mismatches in the materials deposited over the fins, leading to defects later in the manufacturing process. A mask layer is then deposited and later trimmed; the remaining mask layers on the sides of mandrels are known as a spacer. The process of removing mandrels to manufacture spacers is known as *self-aligning double patterning*, as one single structure generates the mask to manufacture the pattern of two fins. The width of spacers defines the T_{FIN}; any deviations will cause disparities in T_{FIN} and impact the fin's performance. Once the mandrels are removed, spacers can be used as hard masks to etch the outer Si layer down to the BOX. This step does not require much precision, as the BOX will serve as a physical boundary. The last step is to remove the spacers, leaving thin, tall stripes of silicon on top of the BOX.

The process to manufacture fins on bulk substrates is longer and requires more precision [9]. A process flow is shown in Fig. 2.10; it concerns a technique known as the subtractive method. In this method, the fin is formed by etching down the Si substrate. Fig. 2.11 depicts a simplified version of this fin patterning process; the initial steps to form the spacers are omitted as they are the same as for SOI substrate. First, a thin cap layer of Silicon Nitride (SiN) is deposited on top of the Si substrate, followed by an amorphous silicon hard mask. SADP is then used to manufacture spacers on top of the hard mask. Then, the wafer is etched down to form the fins; the space between fins is known as *shallow trench isolation* (STI). After this etching step, the spacers are removed from the top of the fins. Next, oxide is grown from the Si substrate, followed by CMP to remove excess oxide and polish the surface. The thin hard mask is removed, and the oxide is etched down to uncover the three-dimensional structure of the fin. This final step defines the H_{FIN} of the device and, therefore, must be precise and well-controlled. Otherwise, it may negatively impact the performance of the transistor as its physical parameters will be deviated from what was designed.

A more complex fin patterning process can be used to manufacture fins with different materials from its substrate. Such process' flow, known as *replacement* method [83], is depicted in Fig. 2.13. In this method, the fin is formed by removing the existing fin structure and growing a new fin in the same place using a new material. Fig. 2.13 depicts

Figure 2.10: Subtractive fin patterning process flow of FinFETs on bulk wafer.

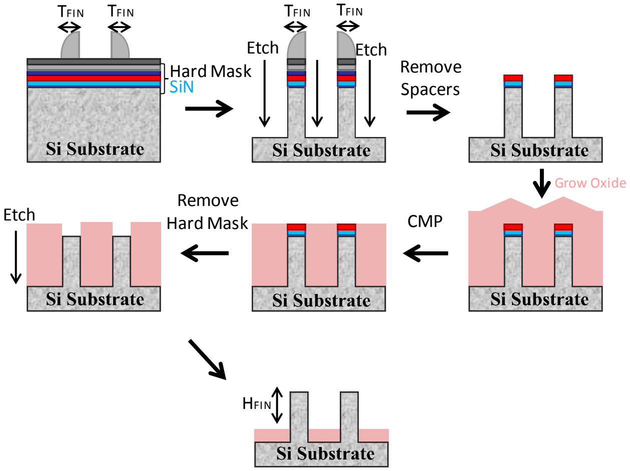


Figure 2.11: Manufacturing process of fins on a bulk FinFET using the subtractive fin method.

Figure 2.12: Replacement fin patterning process flow of FinFETs on bulk wafer.

a simplified version of this fin patterning process. A thicker SiN layer is required for this technique as it will be used as an oxide mold, which allows fins to be grown using the substrate as a base. Besides that, the initial part of the replacement process is similar to the subtractive process: spacers structures are manufactured on top of the fin so that the trenches between fins can be etched down, and oxide is grown and later polished using CMP. Such steps are omitted from Fig. 2.13. The oxide mold is formed by removing the thick SiN cap layer, leaving a fin-shaped void with a silicon substrate bottom. Fins are then formed through epitaxial growth. This technique can be used to grow not only silicon but also other materials such as Germanium or III-V elements, resulting in heterogeneous devices integrated on Si. CMP then removes the extra grown material. Finally, the oxide is etched, exposing the three-dimensional structure of the fin.

Once the fins are patterned on top of the wafer, asymmetries can be manufactured through additional masks and etch steps. Fig. 2.14 depicts the final steps of manufacturing asymmetric H_{FIN} devices on a bulk wafer; note that the same process is used for SOI wafers. In summary, the asymmetry is created by applying a mask on top and around fins designed with a standard H_{FIN}. An oxide etch is performed to deepen the STI and reveal even more of the fin structure, thus increasing H_{FIN}. The STI around fins covered by the mask is not etched, and thus their H_{FIN} is not affected by this process.

Manufacturing Defects

Inconsistent or faulty processes can heavily impact the manufacturing of fin structures, significantly altering the performance of the resulting FinFET device. Naturally, FinFETs can be affected by ***process variation*** **(PV)**; all parts of the FinFET structure, e.g., fin, gate, contacts, can suffer from PV mismatches. Regarding fins, PV can cause many issues [85], such as the ones listed below and illustrated in Fig. 2.15:

- Surface roughness in the sidewalls of fins resulting in degraded V_{th};
- Fin width variation under the gate leading to variations in T_{FIN} and therefore W_{eff};

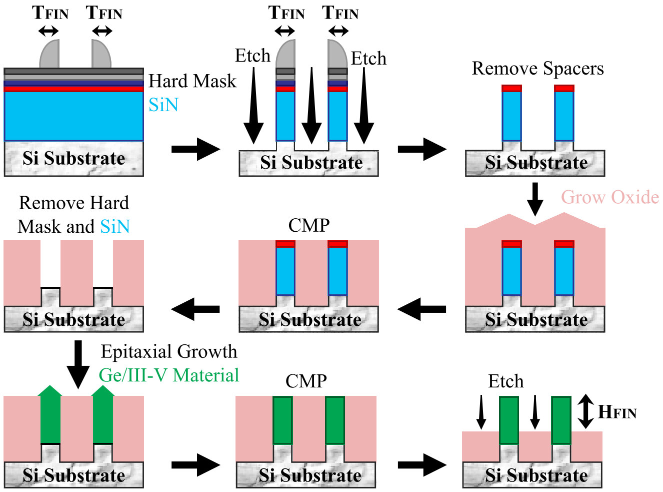


Figure 2.13: Manufacturing process of Ge fins on a Si bulk FinFET using the replacement fin method.

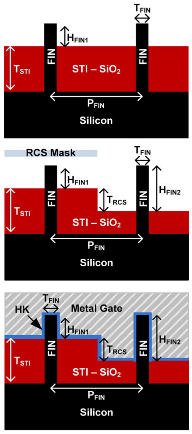


Figure 2.14: Manufacturing process of asymmetric H_{FIN} fins on a Si bulk FinFET [84].

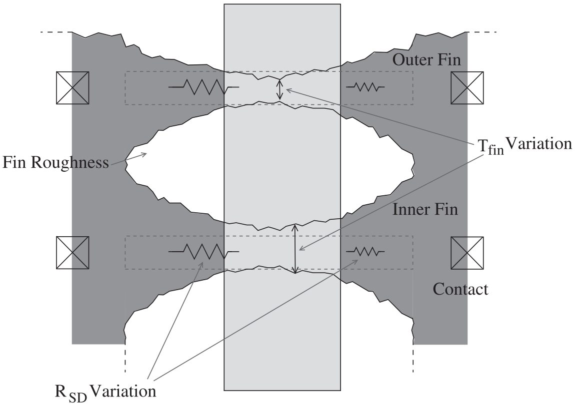


Figure 2.15: FinFET-specific PV mismatch effects in the structure of fins [85].

- Asymmetric variation of the source-drain series resistances R_{SD} due to variations in T_{FIN} outside the gate affecting the drain current (I_D) and transconductance (g_m);
- Structural disparity between inner and outer fins due to different optimal proximity rules of the lithography process causing divergences in the performance of neighbor fins.

It has been shown [86] that surface roughness in the sidewalls of fins is one of the dominant V_{th} variability sources in FinFET transistors; it also has the most significant impact on drive-current variability. Furthermore, variations on T_{FIN} are among the most significant causes of increased leakage current (I_{off}) [87], together with gate variations. Finally, T_{FIN} fluctuations will also cause variations on the parasitic resistance of fins, leading to variations on I_{on} [88].

The etching process to manufacture the fin structure may also cause defects. They may come from plasma etching, a widely-used technique during the manufacturing process of nano-scaled devices [89]. Such a process has a crucial role in defining critical fin dimensions such as T_{FIN}, H_{FIN}, and Fin Pitch; it is also used in the gate manufacturing process [9]. Despite the advancements in plasma processing, it still may cause the degradation of material properties. These degradation mechanisms are referred to as *Plasma (Process)-Induced Damage* (PID); the most relevant type of PID to FinFETs is the ***Plasma-Induced Physical Damage*** **(PPD)**. PPD is the defection creation mechanism caused by the bombardment of ions on silicon [90]. While etching Si to form the fin structures, some ions may impact the fins' sidewalls by sputtering or creating damage by penetrating the Si substrate and undergoing lateral straggling; such effects are depicted in Fig. 2.16. PPD can degrade material properties, affecting carrier mobility and creating latent defects that can function as carrier traps in the channel of FinFET devices, leading to operational speed deterioration and creating aging reliability issues.

The Si etching process results in high aspect-ratio fin structures built from the substrate of the wafer or on the top of buried oxide. While narrow and tall fins enhance

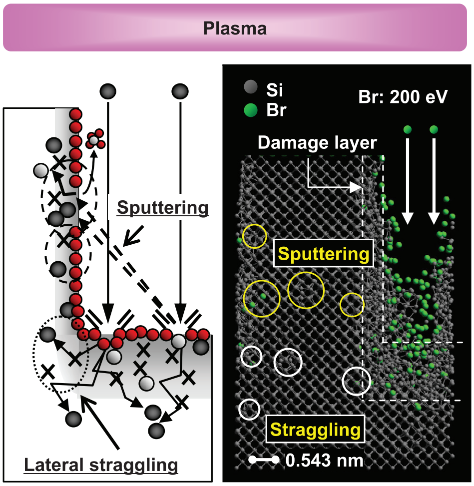
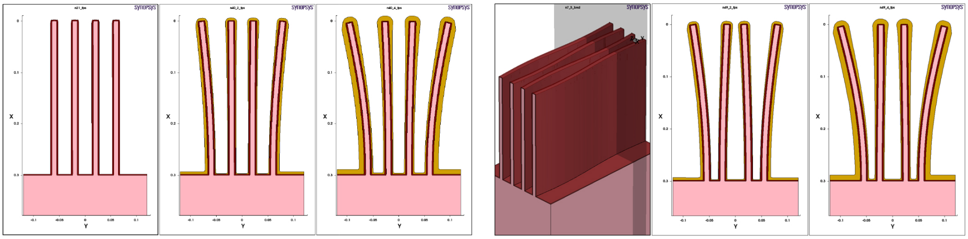


Figure 2.16: PPD creation in a fin structure [89].

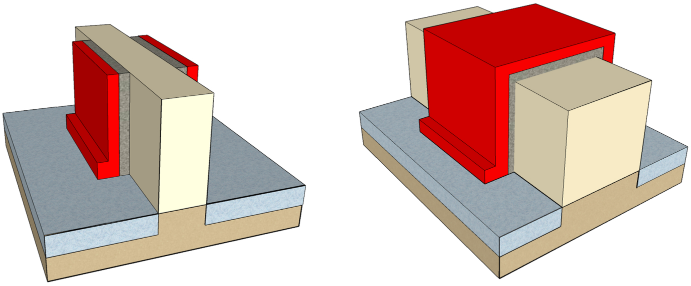


(a) (b)

Figure 2.17: Progressive bending of fins due to (a) non-uniformly distributed cap layer stress and (b) irregularly fin pitch. [91].

electrical characteristics, they are also physically weaker due to their higher ratio between height and width; this may lead to **stress-induced failure** mechanisms, such as fin bending. This behavior affects the device's carrier mobility and can eventually lead to the collapse of fins. This stress has different origins [91]: different thermal expansion coefficient of materials, intrinsic stress of deposited cap layers, or irregular fin pitch caused by lithography issues during SADP and fin patterning. Figure The fin bending behavior is shown in 2.17 [91]; Fig. 2.17a depicts the bending behavior due to cap layer stress, while Fig. 2.17b illustrates the behavior due to fin pitch mismatch.

Finally, fins may also be affected by traditional **open** defects, i.e., a discrepancy in the connection between two nodes [24]. In summary, these defects are material discrepancies, e.g., **cuts**, **bumps**, **voids**, due to airborne particles or lithography inaccuracies created during fin patterning. Although open defects have been intensely investigated in planar transistors, new behaviors are observed when simulating FinFETs with open defects owing to the unique structure and, therefore, new unique defect spots. Cuts on



(a) Independent-Gate (IG) FinFET (b) Shorted-Gate (SG) FinFET

Figure 2.18: FinFET gate structures.

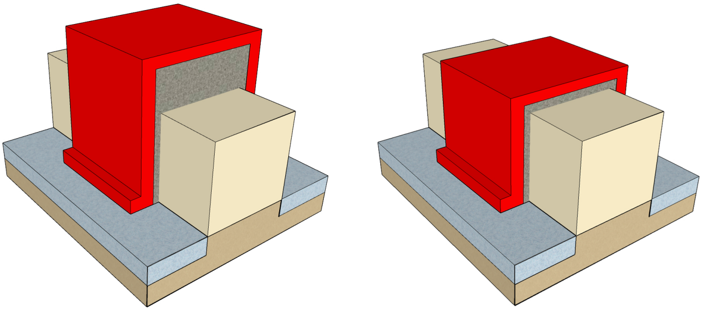
fins will lead to a reduced current going through these devices. Note that the overall impact on multi-fin FinFETs depends on the relative number of defective fins [92]; a device can be regarded as fault-free if a relatively small number of fins are defective, but it will suffer from stuck-open or delay faults if the number of defective fins is large enough.

2.2.3. STEP 3: GATE

The functionality of a FinFET is heavily dependent on its gate configuration. Besides engineering W_{eff}, i.e., the fin's size, engineers also have much freedom regarding gate structure. For example, one may use biasing signals in parts of the gate or change how the gate wraps the fins; this can be achieved by modifying the gate's physical structure and how side gates interact. Two types of FinFETs arise from this distinction: ***shorted-gate*** **(SG)**, also known as tied-gate, and ***independent-gate*** **(IG)**; both structures are depicted on Fig. 2.18. IGs FinFETs are transistors in which the top part of the gate is removed, either by CMP or etching, resulting in isolated and independent side gates. The resulting FinFET is a four-terminal device, i.e., source, drain, and two gates; thus, different signals or voltages can be applied in the gates of a single device.

While channel control is reduced due to no control on top of the channel, IGs offer flexibility as a trade-off. Designers can modulate the $\mathrm{V_{th}}$ of the device by voltage biasing one of the gates. In PMOS devices, the biasing voltage V_{hi} is related to the supply voltage, while the biasing voltage V_{low} in NMOS devices is related to ground. The transistor is reverse-biased if the bias voltage V_{hi} (V_{low}) is above the supply voltage (below ground); this reduces leakage at the price of increased device delay [93]. Contrarily, the transistor is forward-biased if the bias voltage V_{hi} (V_{low}) is below the supply voltage (above ground), reducing delay at the expense of leakage. The area footprint is another critical aspect of IG FinFETS as they require two separate gate contacts, demanding more wafer surface area. This additional area overhead also has a direct impact on the layout of the transistor, which in turn may impact the placement and routing of the circuit [12].

Likewise to fins, asymmetry in the gate structure can also be used to fine-tune the final IG FinFET. IG FinFETs with **asymmetric** T_{OX} have been proposed aiming at having higher I_{on} than while showing the same I_{off}[94]. In these devices, a slightly thicker oxide layer is deposited in the gate used to bias, leading to a significantly improved sub-



(a) Double Gate (DG) FinFET (b) Trigate FinFET

Figure 2.19: FinFET shorted-gate structures.

threshold slope. As it demands two different gate signals, this asymmetry can only be used in IG FinFETs.

In SG devices, on the other hand, all gates covering the fin are shorted; the final result is the traditional three-terminal CMOS transistor. Thus, only one signal is used to control the gate. SG FinFETs offer better electrostatic characteristics, with improved drive strength and channel control [93]. SGs can be further divided based on the T_{OX} on top of the fin. Initially, FinFETs were proposed with a thicker dielectric layer on top of the fin to prevent the formation of parasitic inversion channels at the top corners of the device [69]. Because of the increased T_{OX} on the top surface, two gates are formed on the sides of the fin, creating a ***double-gate* (DG)** structure. In this structure, the fin thickness does not contribute to W_{eff} as there is no connection between the top gate and the channel. In 2012, Intel introduced a variant of FinFET transistors with a reduced dielectric thickness on the top surface of the fin known as **trigate** FinFET [95]. The structural difference of DG and Trigate FinFETs is illustrated in Fig. 2.19. Due to the additional current conduction at the top surface, the width of the fin also contributes to W_{eff}.The top gate also contributes to a smaller gate-source/drain parasitic capacitance; nevertheless, this is countered by the increased parasitic resistance [96].

Additional asymmetry techniques that can be applied to both IGs and SGs have been proposed. **Asymmetry in the gate spacer length** was proposed [97] aiming at manufacturing a FinFET with improved short-channel characteristics owing to an extra underlap in the drain side of the gate; lower SCE, lower drain-induced barrier lowering (DIBL), lower subthreshold swing (SS), and reduced subthreshold leakage have been reported. Another similar technique is to implement **asymmetry in the gate space material** [98] aiming at enhanced speed and reduced leakage current. Instead of using SiO_2 as the only spacer material on the source side, these devices have an inner high-k (such as HfO_2, $k = 25$) and an outer low-k (SiO_2, $k = 3.9$) spacer structure. The heterogenous spacer doubled the I_{on} and reduced the I_{off} and DIBL. Note that both techniques eliminate the symmetrical relation between source and drain current flow. Therefore, they do not adhere to the conventional interchangeable source-drain concept of CMOS technology [12].

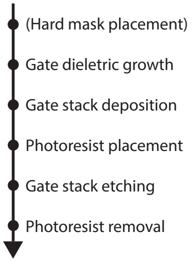


Figure 2.20: Gate stacking process flow of a FinFET.

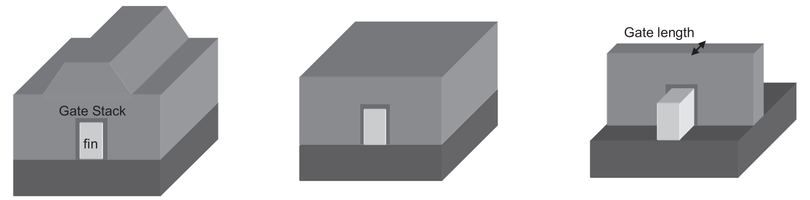
Finally, the gate structure of FinFETs can be manufactured with selective doping to create devices with **asymmetric gate workfunctions** (Φ_G) [99], [100]. This technique is a simple way to improve performance as it only requires a few additional doping steps during manufacturing. Devices with asymmetric Φ_G showed promising short-channel characteristics, with leakage current reduction in two orders of magnitude and with I_{on} only slightly lower than that of symmetrical Φ_G devices.

Manufacturing Steps

The manufacturing process of FinFET gates is named gate stacking; it consists of growing the dielectric and depositing gate materials. The gate stacking manufacture flow is shown in Fig. 2.20. If one wishes to manufacture an IG FinFET, a hard mask must be placed on top of the fin before any material is deposited to ensure two separate gate structures later in the process. A vital aspect of the gate stacking process is selecting a fitting gate metal. Since FinFET transistors have intrinsic or lightly doped channels, designers rely on gate materials with appropriate Φ_G to tune the device's $\mathrm{V_{th}}$. Some metal nitrides such as TiN, TiSiN, TaN, and TaCN, whose work function ranges between 4.4 eV to 4.7 eV, have been used as they are suitable to the high-temperature front-end fabrication process of multi-gate devices [9]. Once the gate material is appropriately selected, a layer thicker than the fin's height is deposited over the wafer. A subsequent CMP step removes the lumps on top of the fins and flattens the gate surface. A photoresist layer is placed on top of the gate stack to pattern the gate structure; designers define L_g by engineering the width of this photoresist. The gate material not covered by the photoresist is then etched out, forming the gate structure. This etching process must have a high selectivity to the fin material to avoid any structural damage in the fins. Once the gate structure is formed, the photoresist material is removed. Sketches representing the gate material deposit, CMP, and gate etch steps are shown in Fig. 2.21 [9].

Manufacturing Defects

Defects on the gate structure are due to an inconsistent or faulty manufacturing process. Likewise fins, gates also suffer from **PV**. The gate structure may be misaligned, resulting in a deviated $\mathrm{V_{th}}$ [69]. Furthermore, variations in the granularity of the metal gate are one of the dominant $\mathrm{V_{th}}$ variability sources in FinFETs [86], [88]. Finally, deviations in L_g lead to a more pronounced variation in a device's I_{off} [87].



(a) Gate material deposition. (b) CMP to remove gate material lumps on top of fins. (c) Gate material etch to form gate structures.

Figure 2.21: Gate stacking steps to manufacture SG FinFETs [9].

The gate structure may also be affected by traditional defects. For example, **open** defects in IG FinFETs may cause delay and leakage problems that no traditional fault model can adequately capture; therefore, hybrid combinations of models are needed [49], [71]. A similar conclusion is obtained when analyzing the traditional **gate-to-channel oxide short defects** in FinFETs. Such defects are usually small dielectric voids that directly connect the gate and the channel. They may decrease the saturation drain current, create a negative drain current when the drain voltage is low, and exponentially increase I_{off} [101]. Experiments have shown that their behavior is much more complex for FinFET devices than for planar CMOS [101], [102], and new models are needed to represent this defect's impact accurately.

2.2.4. STEP 4: SOURCE & DRAIN

The source and drain structures are located at the far ends of a FinFET's fin; they connect the transistor to the upper interconnect layers, together with the gate. Before manufacturing the structures, the fin section used for the source and drain should be doped along the sidewalls to avoid variable parasitic resistances. This doping process can either be uniform or asymmetrical, leading to *asymmetrically doped* (AD) FinFETs. According to experiments, these devices have improved short-channel characteristics (lower DIBL, SS, and subthreshold current) due to reduced electric fields from the less doped terminal, resulting in a significant reduction in leakage current [103].

MANUFACTURING STEPS

The manufacturing process for the source and drain contacts is the final step to manufacturing a FinFET device; it involves doping the fins, protecting the gate structure, and forming the raised source and drain structure. This manufacture flow is shown in Fig. 2.22, and the manufacture steps are illustrated in Fig. 2.23 [9]. First, the source and drain extensions are doped. Then, spacers are formed alongside the sidewalls of the gate and fin; the sidewall spacers on the fins are subsequently removed to expose the fin area used for the source and drain structure. These structures are manufactured, i.e., raised, using selective epitaxy; such a raised structure helps to reduce the parasitic resistance associated with thin fins [104]. The final source and drain structure for different sidewall orientations are shown in Fig. 2.24 [9].

The source and drain structure are the last structure unique to FinFETs. A represen-

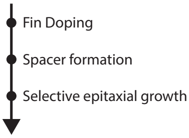
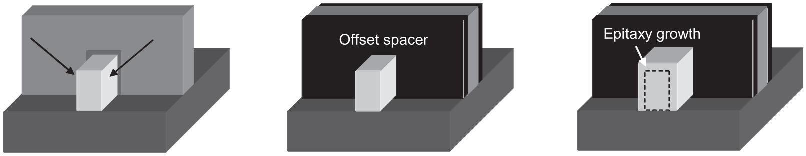


Figure 2.22: Manufacturing process flow of a FinFET's source and drain.



(a) Source and drain extension doping implant.
(b) Gate spacer formation.
(c) Epitaxial growth of raised source and drain.

Figure 2.23: Manufacturing steps of source and drain of FinFETs [9].

tation of a FinFET device up to this point in the manufacturing process is illustrated in 2.25; it concerns a symmetric FinFET built on a bulk substrate, with a single-fin under a shorted trigate. Subsequent manufacturing steps are related to local and global interconnections and do not differ from planar CMOS.

MANUFACTURING DEFECTS

Different materials can be used in the raised source and drain structure to improve carrier mobility, similar to the technique used on fins. The epitaxial growth of SiGe in the source and drain provides stress to boost mobility and enhance the performance of PMOS FinFETs. However, this epitaxy process may result in the growth of SiGe at unwanted locations [105]. Such residues may create **bridge** defects, i.e., the unwanted connection between two circuit nodes. However, it is worth mentioning that process optimizations can be developed to eliminate the SiGe residues.

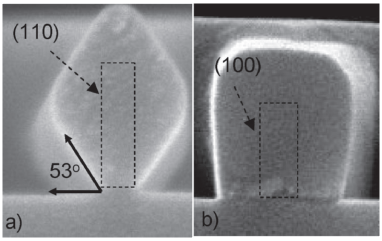


Figure 2.24: Cross sectional SEM images on fins after epitaxial growth on (a) ⟨110⟩ side wall surface and (b) ⟨100⟩ sidewall surface [9].

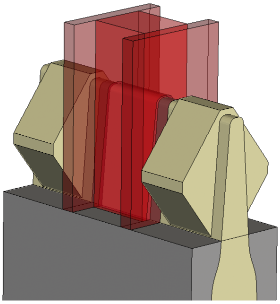


Figure 2.25: An illustrative FinFET structure before contacts are placed on top of the gate, source, and drain.

2.3. FINFET-BASED SRAMS

Static Random-Access Memories (SRAMs) are groups of memory cells designed to store logic values that can be retained at any time. They are named “static” as they do not require periodic refresh signals to preserve their stored data, and “random access” as they allow direct access to any memory location, rather than the access in a fixed sequence [106]. This section presents SRAMs as one type of application for FinFET technology. We first present the different types of SRAM models, e.g., behavioral, functional, logical, electrical, layout. We then discuss the benefits of designing SRAMs using the FinFET technology instead of traditional planar CMOS. Finally, we explore the different FinFET SRAM designs and examine their advantages and limitations.

2.3.1. SRAM MODELS

A model presents physical phenomena, processes, and systems logically and objectively while maintaining a degree of abstraction. Models can also be used to describe a memory system and divide them into different levels of abstraction. Traditionally, memory models have been divided into five levels: behavioral, functional, logical, electrical, and layout [26], as shown in Fig. 2.26. They help simplifying the explanation and treatment of systems by explicitly presenting information relevant only to the discussion about the system at that level while hiding irrelevant information.

The layout model is the closest to the actual physical system; it assumes complete knowledge of the memory chip’s layout. On the other hand, the behavioral model considers the entire system as a black box and only deals with the memory’s inputs and outputs. Therefore, as we move up in the hierarchy levels, the models become less representative of the physical world and more related to the way the system behaves, or in other words, less material and more abstract. Next, we explore each of these levels, starting from the highest abstraction level.

BEHAVIORAL MODEL

This model is displayed as the highest abstraction level in Fig. 2.26; it is based on the system specification. At this level, the only information given is the relation between input and output signals while treating the system as a black box. A model at this level

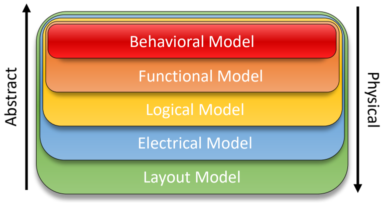


Figure 2.26: Memory models and levels of abstraction.

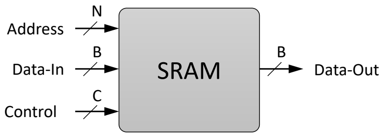


Figure 2.27: General SRAM behavioral model.

usually uses timing diagrams to convey information about the system's behavior; it describes how the model interacts with the external world, such as memory read and write operations.

The most common memory behavioral model is a box with inputs and outputs, as shown in Fig. 2.27. The memory system receives an address, control, and data-in values from the exterior via the inputs and produces data-out values via the outputs. More specifically, these inputs consist of C controls, N address lines, and B input data lines, where B is the word-width of the memory; the output consist only of a B-bit data-word. Usually, the data-in and data-out lines are combined to form bidirectional data lines, thus reducing the number of required chip pins.

FUNCTIONAL MODEL

The functional model characterizes the functions the system needs to fulfill to operate appropriately. The system is divided into several interacting subsystems, each with a specific function; each is a black box with its own behavioral model. The collective operation of the functional blocks results in the proper operation of the system as a whole. A typical SRAM functional model consists of a memory cell array, a row address decoder, a column address decoder, read/write circuits, and data flow and control circuits, as depicted in Fig. 2.28. We explain these functional blocks in detail next.

1. **Memory array:** The memory cell array is the heart of the SRAM. It consists of cells organized in an array structure; the memory bit storage capacity is given by $R \times C$, where R is the number of rows and C is the number of columns. The memory array can have distinct logic and physical organizations. For example, a memory chip of 1 Kbit can be logically seen as 1000 addresses with a word size of 1 bit, while it is physically organized as a 1000×10 matrix. The word width also plays a role in deciding the physical organization of the memory cell array. While the number of rows can be any integer, the number of columns should be an integer multiple of

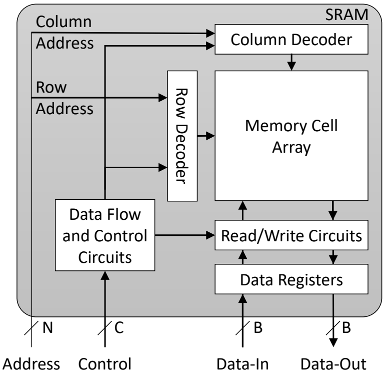


Figure 2.28: SRAM functional model.

the word width; there is always an integer number of memory words in one row, i.e., C *mod* B = 0.

2. **Address decoders:** Due to its matrix organization, the memory cell address is divided into row address bits and column address bits. Therefore, the memory system contains both row and column decoders. While the high order bits select the appropriate row, the low order bits select the appropriate column. Combined, they point to a unique memory cell, if B = 1, or to a unique word if B >1; in this latter case, more than one column, i.e., B cells, are accessed together at a given time.

3. **Read/write circuitry:** They can be divided into *sense amplifiers* (SAs), *write drivers* (WDs), and *precharge* (PC) circuits. During a read operation, the content of the selected memory cells is read and amplified by SAs, loaded into the data register, and outputted through the data-out pins. During a write operation, the data on data-in pins is loaded into data registers and written into the selected memory cells using WDs. As mentioned before, it is possible to combine data-in and data-out lines to form bidirectional data lines.

4. **Control & timing circuitry:** these circuits are responsible for generating the signals that coordinate all memory blocks. More specifically, they coordinate the execution of operations by generating control signals such as read enable, write enable, sense amplifier enable, and others.

LOGICAL MODEL

This model is based on the logic gate representation of the system. At this level, simple boolean relations and logic equations are used to establish the desired system functionality. Logical models are usually used to describe digital circuits. The memory itself has few digital circuits; they only account for a small part of the memory chip. More specifically, the only digital circuits in a memory system are the decoders. Therefore, we present logical models only for these circuits.

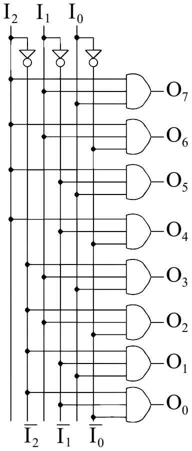


Figure 2.29: Logical model of a 3 to 8 decoder.

Address decoders are used to select particular memory cells in a memory array. A row address decoder decodes the row address and activates a specific WL, while a column address decoder decodes the column address and activates a specific pair of BLs. This allows subsequent read/write operations to target the selected cell(s) in the memory array. Typically, an address decoder block consists of multiple smaller decoders. Pre-decoders are used initially to decoder the address inputs. Then, post-decoders combine the pre-decoders inputs with timing and enable signals from the data flow and control circuits. Nevertheless, pre- and post-decoders have the same logic implementation, i.e., decoders based on NOT and AND gates. Even though their detailed implementation depends on their size, e.g., 3 to 8, 2 to 4, their general structure follows the same pattern: an n number of inputs (I) are inverted with NOT gates and then combined in multiple ways with AND gates to generate a total of 2^n outputs (O). Fig. 2.29 shows a straightforward implementation of a decoder using NOT and AND gates. It consists of a 3 to 8 decoder: three inputs (e.g., addresses) and eight outputs (e.g., WLs).

ELECTRICAL MODEL

This model is based on the basic electrical components that build up the system; it is the electrical equivalent circuit that represents a system's behavior and contains details about the internal structure at the electrical level. The components are mostly transistors, resistors, and capacitors in an SRAM. At this level, we are not only concerned with the logical interpretation of an electrical signal but also the actual electrical value of it (e.g., voltage, current, and resistance). Since this thesis is primarily concerned with experimental analysis at the electrical level of SRAMs, i.e., SPICE-based electrical-level circuit simulations, this memory model is discussed in more detail. More specifically, we discuss the key electric circuits of a memory system, i.e., the memory cell, the decoders, the SA, and the WD.

1. **Memory cell:** an SRAM cell is a bistable circuit that can be driven into one of two possible logic states: '1', referred to as "true", or '0', referred to as "false". It can re-

tain its state as long as a power supply is provided. One of the most used SRAM cell designs is composed of six transistors. Such a design, named 6T, is used through the experiments in this thesis. An electrical-level representation of a 6T memory cell is shown in Fig. 2.30. Two cross-coupled inverter gates are designed by connecting transistors M0, M1, M2, and M3. The output of each inverter is one of the cell's storing nodes; the logic value stored in the cell corresponds to the digital representation of the voltage on Q ('1' for V_{DD}, '0' for 0 V).

In addition, transistors M4 and M5 serve as pass gates; they connect the internal nodes of the cell to the rest of the memory system. They are controlled by a wordline *WL* signal, which is used to access a particular memory cell. Besides their connection to the cell's storing nodes, they are also connected by complementary *bitlines* (BLs), which are signals used to read/write the memory cell. Therefore, a memory cell can be accessed via its WL and BLs.

Memories can operate in three distinct modes: hold mode, read mode, and write mode. In hold mode, no operations are being performed on the cell. The word line is off, and the cell has no connection with the rest of the memory array. In read mode, BL and $\overline{BL}$ are precharged to a high level (V_{DD} or '1'). When the WL is enabled, the memory cell starts pulling down one of the bit-lines depending on the value stored. The voltage difference between the complementary BLs is sensed and amplified by the read circuit, and the appropriate value is loaded in the data register. Note that SRAM's read process is non-destructive, i.e., the cell retains its data after the read operation. During write mode, BL and $\overline{BL}$ are driven to complementary data values; when WL is enabled, the cell is forced to the state presented on BLs as these lines are driven with more force than the force with which the cell retains its information.

2. **Memory decoder:** as previously mentioned, pre- and post-decoders consist of NOT and AND gates. At the electrical level, these gates are described using transistors, as shown in Fig. 2.31. A NOT gate (Fig. 2.31a) is composed of 1 PMOS and 1 NMOS; the output is a voltage that represents the inverted logical value of the input, i.e., $Out = \overline{In}$. Moreover, an AND gate is obtained by combining a NAND (Fig. 2.31b) and a NOT gate. The output is the voltage representing the logical output of the boolean function $Out = A \times B$.

3. **Sense amplifier:** this circuit *senses* the voltage on a column's BLs and *amplifies* it to a logic level, i.e., '0' or '1'. In this thesis, we use the SA design depicted in 2.32. Initially, both nodes A and B are precharged to V_{DD}. The *column mux* signal (CMUX) is enabled together with WL; it comes from the data flow and control block. Consequently, one of the SA's nodes (either A or B) is discharged; this phase is known as the *sensing* phase. Once there is enough BL discharge, the WL and CMUX are disabled, and the signal *sense amplifier enable* (SAE) is activated. This next phase is known as the *amplification* phase; the node that was slightly discharged during sensing is then fully discharged, while the other node stays at V_{DD}. Both storing nodes are then used as inputs for an SR latch, which serves as the interface for the output pin.

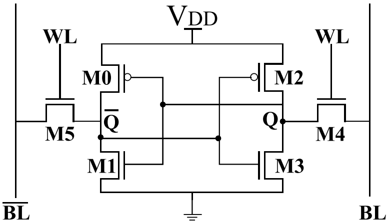
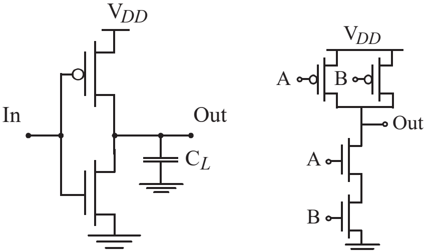


Figure 2.30: The electrical model of an SRAM 6T cell.



(a) NOT gate. (b) NAND gate.

Figure 2.31: Electrical model of a NOT gate and a NAND gate [107].

4. **Write driver:** this circuit *writes* a memory cell by *driving* the new value into the cell and eventually overpowering it. Furthermore, the WD circuit can also be coupled with the PC, thus saving area. The electrical model of the WD used in this thesis is shown in Fig. 2.33. The gate inputs concern the WD for BL; for $\overline{BL}$, the data input in the NAND-2 is inverted, while the data input in the AND-3 is not. When the enable WD signal is activated, data to be written is passed from data-in pin to the complementary BLs; BL contains *Data_In* while the complementary $\overline{BL}$ contains the $\overline{Data\_In}$. Once the WL is activated, the WD forces its value into the cell. Once the operation is finished, the memory goes back to idle mode. The PC signal is activated, and both BLs are kept in V_{DD}.

LAYOUT MODEL

The layout level is the last abstract memory model. It represents the memory in terms of geometric shapes that correspond to the patterns of metal, oxide, or semiconductor layers that make up the components of an integrated circuit. Furthermore, it describes all components' physical structure, location, and dimensions. As the semiconductor technology scales down, the layout model became even more relevant for companies, as it now also defines the three-dimensional structure of the FinFET device. Due to the proprietary nature of this information and its high complexity, semiconductor manufacturers rarely disclose the layout models of their chips. Nevertheless, some experimental layouts by Intel [13], Samsung [108], and GLOBALFOUNDRIES [109], and academic models [110] have been developed and published. For example, Fig. 2.34 [110] shows the layout level of a FinFET 6T SRAM cell designed on a 14 nm technology.

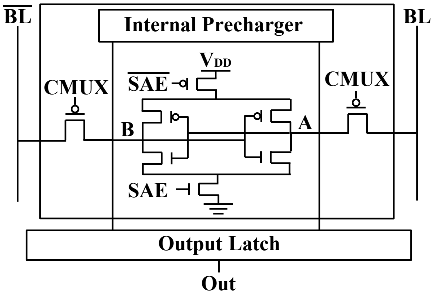


Figure 2.32: The electrical model of an the sense amplifier used in this thesis.

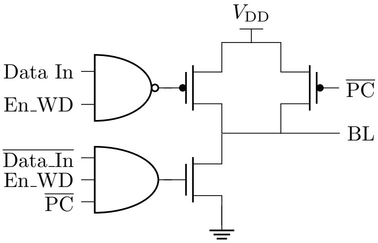
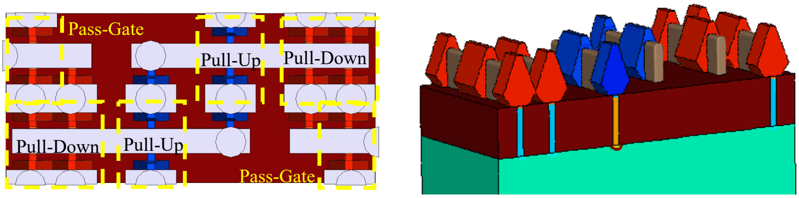


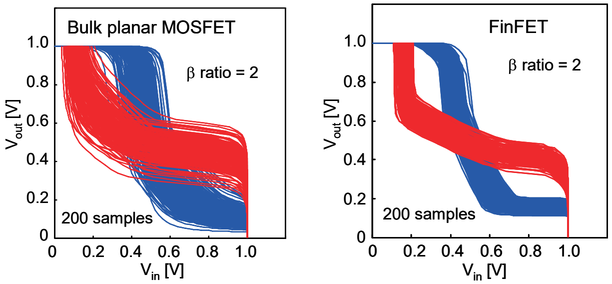
Figure 2.33: The electrical model of an the write driver and precharger used in this thesis.



(a) Top view.

(b) Isometric view, without any contact structures.

Figure 2.34: Layout model of a FinFET SRAM [110].



(a) SNM for planar bulk CMOS SRAM.

(b) SNM for trigate FinFET SRAM.

Figure 2.35: SRAM butterfly curves for bulk planar and FinFET. The FinFET SRAM exhibits superior SNM thanks to smaller V_{th} variation due to its undoped channel [88].

2.3.2. FINFET SRAMS ADVANTAGES OVER PLANAR CMOS SRAMS

Memory cells are usually designed to be as small as possible to achieve the highest density possible. However, when approaching the 20 nm node, many aspects impose sizing restrictions, prevent further scaling down and improvements [107]. One aspect is power consumption. Lowering the supply voltage proves to be the best alternative to save power. However, conventional CMOS SRAMs are limited to scaling power supply due to the random variations of V_{th} caused by random dopant fluctuation. This is not the case for FinFET SRAMs, as high doping is not required in FinFETs owing to their enhanced SCE, which expressive reduces random dopant fluctuation [93]. Therefore, variations on V_{th} are less significant, allowing V_{DD} to be scaled down.

Another aspect is process variation and overall device stability. The stability and performance of SRAM cells can also be improved by using FinFETs. Experiments have shown that the *static noise margin* (SNM) of FinFET cells is superior, and its distribution is more uniform than cells with planar CMOS, owing to the smaller V_{th} variation due to its undoped channel and slower V_{th} roll-of with L_g [88]. These experiments' results are depicted in Fig. 2.35, which shows the SNM butterfly curves of SRAM cells designed with CMOS and FinFET under PV. As can be seen, the curves from FinFET cells are much more homogeneous when compared to their planar counterpart. Furthermore, it has also been shown that other SRAM metrics, such as read SNM and *write margin* (WM), also benefit from using the FinFET technology [111]. It is clear that FinFET SRAMs can bring many inherent advantages in robustness, power consumption, and reliability, thus outperforming the traditional planar CMOS SRAM.

2.3.3. FINFET SRAM DESIGN

The versatility of FinFET designs implies a multitude of SRAM cell designs. Many authors have proposed pushing the manufacturing process and SRAM design to their limits to obtain high-performance cells. Kawasaki et al. [112] discussed the challenges of cell-size scaling and the mismatches in V_{th}. Basker et al. [109] explored memory cells with aggressively scaled fin pitch and their design challenges. Guillorn et al. [113] investigated the limits of device patterning to manufacture high-density FinFET memory cells. Kazuhiko et al. [114] designed and manufactured an SRAM cell with reduced leakage current and dynamic power consumption by using IG FinFETs. Tawfik et al. [115] presented and characterized the metrics of different SRAM cell designs using IG FinFETs and low-$\mathrm{V_{th}}$ SG FinFETs. Bhoj et al. [116] classified 6T FinFET SRAMs into three distinct configurations: fully shorted-gate, partial or fully independent-gate, and multiple-Φ_G shorted-gate configurations; they evaluated each configuration regarding key memory performance indicators, e.g., read power noise margin, read current (I_{READ}), I_{off}.

Some authors have also proposed to use asymmetric FinFETs to improve SRAM performance. Sachid et al. [84] presented a 6T memory cell in which some transistors were asymmetrical H_{FIN} devices; such a memory cell was manufactured by Chen et al. [117], who reported improvements of 25% in the SNM. Goel et al. [97] designed SRAM cells using FinFETs with asymmetric gate spacer length. The authors exploited the asymmetry provided by these devices to achieve 11% and 6% improvement in read SNM and WM, respectively, at the cost of 7% increased access time and 7% increased cell area. Moreover, they also reported a 57% I_{off} reduction in these cells. Moradi et al. [103] proposed memory cells using AD FinFETs. However, the improvements brought forth by using devices with different doping concentrations may not pay off; while there were improvements in SNM, RSNM, WM, write access time, and I_{off}, they are not significant trade-offs for the 20% higher read access time. Finally, Pal et al. [98] used FinFETs with asymmetric gate space materials to design SRAM cells. They proposed to use the dual-k spacer either in the source or drain side of the transistor. Both cases showed improvements in SNM and RSNM. Nevertheless, cells with dual-k spacers on the source side exhibited an almost 20% reduction in WM due to stronger PU devices.

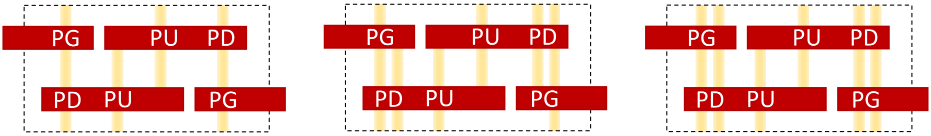
Despite all these experiments, the most significant FinFET SRAM design choices are fairly simple; they come from the fin distribution and cells with varying angle orientations. As previously mentioned, memories are designed with the highest density possible; this implies that cells must be as small as possible. Therefore, when designing a FinFET SRAM cell, one may choose different fin configurations. In summary, three different configurations are used by the industry [13], [118], [119]. They are further described below and depicted in Fig. 2.36.

- **High Density (HD)**: in this configuration, the *pull-up*, *pass-gate*, and *pull-down* (PU, PG, and PD, respectively) are all designed with only one fin. Therefore, its configuration is defined as 1:1:1. This configuration provides the best density at the cost of cell performance.

- **Low Power (LP)**: in this configuration, the PD transistors are designed with two fins; its configuration is defined as 1:1:2. The additional fin in the PD improves the PD to PG drive current ratio, thus enhancing the cell's read margin.

- **High Performance (HP)**: the HP cell has a 1:2:2 configuration. Besides the read margin enhancements, it also has improved write capability. **This is the FinFET SRAM design used throughout this thesis.**

Changing and optimizing the crystal surface orientation of the PU, PG, and PD transistors is also a feasible and straightforward approach to improve the performance of

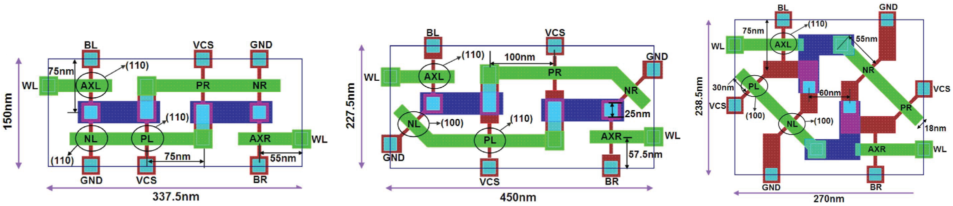


(a) High density (HD) configuration

(b) Low power (LP) configuration

(c) High performance (HP) configuration

Figure 2.36: Simplified FinFET SRAM 6T cell layouts.



(a) PU, PD, and PG transistors, all with (110) orientation.

(b) (110) PU, (100) PD, and (110) PG transistors.

(c) (100) PU, (100) PD, and (110) PG transistors

Figure 2.37: Layout of single-fin FinFET 6T SRAM cells with varying crystal orientations [120].

FinFET SRAMs. As previously mentioned, the mobility of PMOS and NMOS devices is directly related to their physical orientation in the wafer. As the FinFET technology enables manufacturing devices in different planes other than ⟨100⟩, cell performance can be optimized by modifying the physical layout of cells. Different 6T layouts are depicted in Fig. 2.37 [120]. Many works investigating surface orientation in FinFET memory cells have been published in literature [70], [120]–[124]. Metrics such as area, cell stability, access time, and process variation were evaluated. Although some authors presented positive results regarding cells with mixed surface orientations, the area penalty of this approach is still a significant drawback, making it infeasible for high-density memories.

2.4. FINFET TECHNOLOGY OUTLOOK

The semiconductor industry today faces the difficult challenge of extending the CMOS technology. To continue improving the performance of circuits and developing new applications, the industry explores different paths of action. The *Semiconductor Industry Association* (SIA) has been discussing possible directions since 2005; it established three paths [125], as shown in Fig. 2.38:

- **More Moore** focuses on the shrinking of technological nodes; it is achieved by geometrical scaling of devices and by designing technologies that enable high performance, low power, high reliability, low cost, and high design productivity. The advancement of the FinFET technology lies in this direction.
- **More than Moore** focuses on the functional diversification of devices by integrating multiple systems, e.g., sensors, actuators, oscillators, RF communication, power

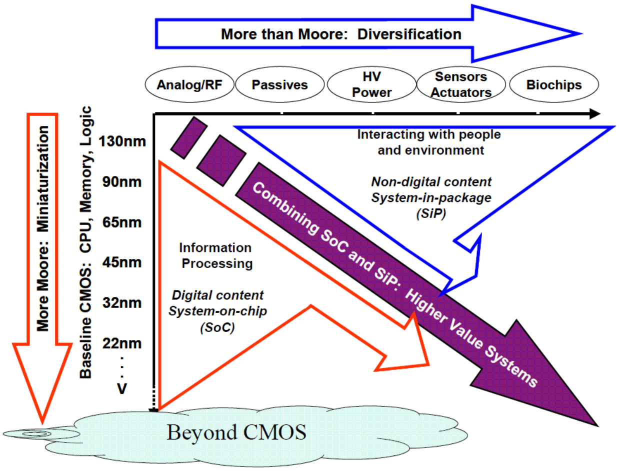


Figure 2.38: More Moore, More than Moore, and Beyond CMOS [125].

control, in the same package.

- **Beyond CMOS** deals with emerging devices and materials and focuses on new information processing elements or technologies. Carbon-based nano-electronics, spin-based devices, ferromagnetic logic, atomic switches, and nano-electro-mechanical-system switches are a few examples of Beyond CMOS.

In this section, we explore the outlook of the FinFET technology based on the 2021 edition of the *International Roadmap for Devices and Systems*^{T} *M* [126]. First, we present and discuss the novel transistor structures likely to overtake FinFETs. Then, we list the FinFET challenges related to More Moore; we first present near-term (2021-2025) issues, followed by long-term (2026-2034) issues.

2.4.1. NOVEL TRANSISTOR STRUCTURES

FinFET remains the key device architecture to sustain More Moore scaling; some authors estimate it is likely to sustain miniaturization until 2025 [127], [128]. Nevertheless, the industry has already been working on the manufacture of *Gate-all-around* (GAA) FETs, the structure to replace FinFETs. A concept sketch of a GAA is shown in Fig 2.39 [129]; it consists of a channel surrounded by the gate on *all* its sides. The channel could have either the form of a sheet or a wire, resulting in GAA nanosheet FETs and GAA nanowire FETs, respectively [130]. This difference is further illustrated in Fig. 2.40, which depicts the technology transition from planar to FinFET to GAA. GAA structures offer excellent electrostatics and short-channel control. Furthermore, they can be fabricated with minimal deviation from FinFETs, and avoid some of the patterning challenges associated with scaled technologies [17].

The first introduction of GAA nanosheets into the market is expected for 2022. Fig. 2.41 [131] shows the IRDS More Moore roadmap concerning the structure of devices. By 2030, it is expected to see not only fully-stacked fins but also fully-stacked transis-

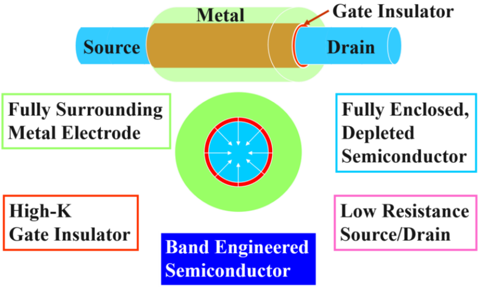


Figure 2.39: Sketch concept of the “ideal FET” device, i.e., a *Gate-All-Around* (GAA) transistor [129].

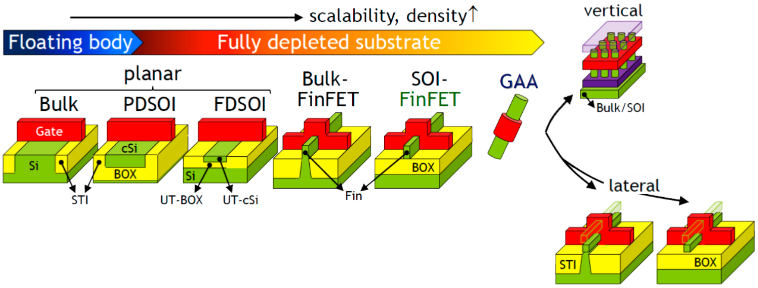


Figure 2.40: Transistor technology transition, from planar to GAA [130].

tors. From then on, the semiconductor industry hopes to increase density much beyond what device scaling alone would permit and ultimately pave the way for vertical integration [132]. Even though we have focused on the physical structure of FinFETs, many other characteristics are also expected to evolve, e.g., high-mobility channels, strain engineering, reducing parasitic device resistance and capacitance. A thorough discussion of the listed characteristics and many others can be found in the IRDS 2021 More Moore [131].

2.4.2. NEAR-TERM ISSUES

The transition from FinFET to GAA is expected to start soon. With this transition, near-term issues are expected to arise; in its latest report (2021), the IRDS defines this period as from 2021 - 2025 [131]. We briefly presented some of the issues; a more in-depth discussion can be found in the IRDS 2021 More Moore report.

POWER SCALING

The scaling down of FinFET’s voltage and capacitance has significantly slowed down, and unfortunately, there are no viable solutions for power reduction. Introducing GAA devices is a remedy to reduce the supply voltage; yet, not in a sustainable manner that enables continuous scaling. Power scaling is also limited due to the loading capacitance scaling slow down. This capacitance is increasingly becoming impacted by the parasitic components of the device with continuous dimension scaling. Therefore, introducing

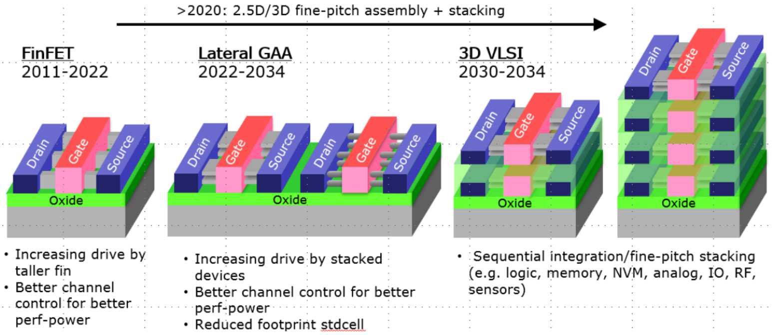


Figure 2.41: Device architectures and their applications [131].

low-*k* materials, new contact access schemes, and local interconnect schemes that allow lower parasitics is needed.

PARASITICS SCALING

When introducing GAA devices, one of the primary issues concerns the increased parasitics from stacked devices and maintaining control over it. Due to GAA's structure, high-aspect-ratio contacts are needed to access the bottom gate contact, increasing the contact resistance and the fringe capacitance between the gate and drain/source. Interface resistance will also require new silicidation schemes that conformally wrap the source/drain.

COST REDUCTION

Naturally, reducing the costs per area unit is still very much desired; this will be facilitated by extreme-ultraviolet (EUV) lithography and design-technology-co-optimization methodologies. EUV challenges related to throughput and yield need careful device dimensioning that optimizes the die cost. Therefore, new process-enhanced designs are needed to scale down the area of standard and bit cells even further. Furthermore, integrating those design constructs might require new materials to allow better etch selectivity and self-deposition.

INTEGRATION ENABLEMENT FOR SRAM-CACHE APPLICATIONS

Bit cell area scaling is slowing down due to the slow-down of the fin and gate pitch scaling. GAA devices bring an opportunity to reduce the SRAM area significantly through optimized layouts that eliminate the critical design rules impacting the area.

INTERCONNECT SCALABILITY

The interconnect scalability suffers from two main aspects: interconnect resistance, and interconnect materials. Interconnect resistance has now entered an exponential increase regime because of the non-ideal scaling of the barrier for Cu and increased scattering at the surface and grain-boundary interfaces. Therefore, there is a need for new barrier materials and Cu alternative solutions. Furthermore, time-dependent dielectric breakdown limits the minimum space between the adjacent lines for a given low-*k* dielectric, thus restricting the possible interconnect materials even further.

2.4.3. LONG-TERM ISSUES

Once GAA FETs are successfully employed, the industry might shift its attention to long-term issues; in its latest report (2021), the IRDS defines this period as from 2026 - 2034 [131]. We briefly presented some of the issues; a more in-depth discussion can be found in the IRDS 2021 More Moore report.

POWER SCALING

Once GAAs are successfully introduced, the only remaining solution to overcome power scaling issues is to use *steep-subthreshold* (SS) devices to enable complementary SoC functions, i.e., replacing mainstream CMOS aiming at reducing power. However, most steep-SS device candidates do not have an adequate performance comparable to CMOS at nominal supply voltages. Therefore, to maximize the performance of steep-SS devices, new architectures are necessary to attain the performance through parallelization.

USE CASES OF VERTICAL DEVICE STRUCTURES

Vertical device structures could enable performance scaling and functional diversification. However, using these devices will also lead to routing congestion and increased parasitics. Therefore, there is a need for new logic schemes and architectures that maximize the advantage of the 3D capability.

THERMAL ISSUE DUE TO INCREASED POWER DENSITY

GAA devices have limited heat conductance due to confined architecture. Thus, it is straightforward that 3D stacking causes thermal challenges. Increased pin density due to aggressive standard cell height scaling and increased drive by stacked devices put significant pressure on the power density.

COST REDUCTION WITH 3D INTEGRATION

A significant challenge will be managing the cost, yield, and process complexity of 3D integration. Using vertical devices separated by the interconnect increases the wafer cost and the number of masks, i.e., process complexity, adding pressure to defect control. Architectures need to be refined to reduce the interconnect complexity between tiers and simplify integration and function per tier, e.g., I/O in one tier SRAM in another tier.

INTEGRATION OF NON-CU METALLIZATION TO REPLACE CU

Finally, integrating non-Cu interconnects may bring some issues to the manufacture of GAAs. Such interconnects must meet all the requirements imposed by electromigration and time-dependent dielectric breakdown. Furthermore, they also must be compatible temperature-wise with devices used in 3D integration.

3

FAULT MODELING FOR FINFET SRAMS

In Chapter 2, we discussed the memory functional model, which combines the memory specifications' with its internal subsystems structure. For testing purposes, this model is further simplified; the so-called 'reduced functional memory model' consists of four subsystems: the address decoder, the memory array, the write logic, and the read logic. Since the vast majority of memory devices contain these subsystems, the reduced functional fault model is, up to a large extent, independent of specific memory implementations.

This chapter presents the fault modeling for FinFET SRAMs concerning the previously mentioned blocks. First, we introduce the fault primitive concept and a classification scheme based on the fault's sensitization and impact. Second, we present the fault space definition for each block; first, for static faults, and then for dynamic. Following, we propose a methodology to validate the memory array fault space. Finally, we present the fault modeling results for memory array faults using the proposed validation methodology.

Parts of this chapter have been published in TVLSI'21 [60]

3.1. FAULT CLASSIFICATION BASED ON SENSITIZATION AND IMPACT

Manufacturing defects may lead to unexpected and undesired behaviors. These faulty behaviors, i.e., faults, can only be detected using appropriate test solutions, which requires accurate fault modeling. For SRAMs, fault modeling comprises two steps: fault space definition and fault space validation. However, before defining the fault space, it is necessary to explore and classify all possible types of memory faults. This section proposes a classification for memory faults based on their impact on the memory and sensitization requirements. We first propose three ways to distinguish memory faults. Then, we introduce the fault primitive concept, a method to describe memory faults.

3.1.1. PARAMETRIC VS. FUNCTIONAL FAULTS

Manufacturing defects may impact the memory system in two ways. First, it may cause operations to fail, e.g., a write operation cannot flip the cell's content, or a read operation returns the opposite expected value. These faulty behaviors are known as **functional faults**, i.e., they affect the memory's functionality. These faults directly impact how the memory operates, as they will cause operations to fail. Therefore, their detection is straightforward; it is only necessary to analyze the read operations' outputs.

On the other hand, a defect may only lead to a deviation in the memory's electrical parameters, usually referred to as *specs*. For example, a defect can increase I_{off} or reduce the cell's SNM. While still faulty, such behaviors will not cause the memory to fail; read and write operations are still executed as expected. These faulty behaviors are known as **parametric faults**, i.e., only the cell's parameters are affected.

3.1.2. STATIC FAULTS VS. DYNAMIC FAULTS

Memory faults require specific conditions to be triggered, e.g., a read fault requires a read operation. In some cases, no operations are necessary to trigger a fault. In others, a fault is only triggered after multiple consecutive operations. Test engineers must be aware of these two scenarios to develop high-quality test solutions. A **static fault** refers to faults that are sensitized with *at most* one operation. For example, a read operation causes a specific cell's content to flip. Furthermore, note that static faults also include faults that do not require any operation, e.g., the state of a specific cell is always stuck at '1'.

Contrarily, a **dynamic fault** is a fault that is only triggered after *at least* two operations. For example, a read operation fails if and only if it was applied immediately after a transition operation. Dynamic faults have become a significant test issue for deep-scaled memories; FinFET memories are much more prone to dynamic faults, while in planar CMOS memories, the vast majority of faults were static [133]. Moreover, the number of operations required to trigger a dynamic fault depends on many factors, such as defect size, temperature, frequency. In order to alleviate testing efforts, it is desired to apply appropriate stressing conditions so that this number is the smallest possible.

3.1.3. SINGLE CELL VS. COUPLING FAULTS

A fault may be further based on how many cells are involved in the process of triggering it. Faults previously discussed have been all examples of **single-cell faults**; they only

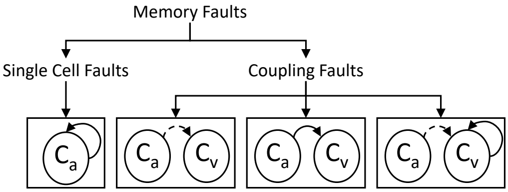


Figure 3.1: Single cell and coupling faults.

involved the faulty cell itself. However, in **coupling faults**, more than one cell is involved – the aggressor cell (C_a) and the victim cell (C_v). These faults, also known as *multi-cell* faults, may occur in three distinct situations, as shown in Fig. 3.1. First, the state of C_a may influence the state of C_v in this case, no operations are needed in C_a and C_vṠecond, an operation in C_a may influence the state of C_vḞinally, an operation in C_v may fail due to C_vś current state.

3.1.4. FAULT PRIMITIVE CONCEPT

The concept of *Fault Primitives* (FPs) [134] have been traditionally used to describe all memory array faults that might lead to incorrect functional behavior, i.e., functional memory array faults. An FP is denoted by the three-tuple notation $\langle S/F/R \rangle$ as follows:

- ***S*** denotes the sequence of operations that sensitizes the fault. If the fault involves only one cell, i.e., a single-cell fault, then S describes the state or operations in C_v only. For such case, S takes the form of $S = x_0 O_1 x_1 ... O_n x_n$, where $x_i \in \{0,1\}$, $i \in \{0,1,...,n\}$, and $O \in \{r,w\}$. '0' and '1' denote logic values, while 'r' and 'w' denote a read and a write operation, respectively. n represents the number of operations necessary to trigger the fault. If $n \leq 1$, the fault is defined as a static fault. If $n \geq 2$, the fault is defined as a dynamic fault.

  On the other hand, if the fault involves multiple cells, i.e., coupling faults, then S describes states or operations in C_a and C_v. It takes the form of $S = S_a\ ;\ S_v$, where S_a describes the sensitizing operations or state of C_a, and S_v describes the sensitizing operations or state of C_v. S_a and S_v are described similarly as to S, i.e., $S_i = x_0 O_1 x_1 ... O_n x_n$, where i is either a or v. Note that the scenario in which both S_a and S_v contain operations require particular memory architectures, e.g., 2-port memories, which are not covered in this thesis.

- ***F*** denotes the stored value in the cell after the fault takes place. While there are only two logic values that a cell can store ('0' and '1'), the voltage in nodes Q and $\overline{Q}$ may be different from V_{DD} or GND in the presence of a manufacturing defect. If the voltage difference between the two nodes is too small (i.e., the voltages on both nodes are almost the same), the cell may be storing an undefined value ('U') [135]. Thus, $F \in \{0, \mathrm{U}, 1\}$. Additionally, a subscript may be used to specify the faulty effect's nature [136]; we use the subscript '$_r$' to denote retention faults [47], [137], i.e., the cell's value flips some time (at least longer than the period of one memory operation) after the cell was stressed.

- ***R*** represents the read output after applying S. More specifically, $R \in \{-,0,1,?\}$. If the final operation in S is not 'r', then $R = -$. Furthermore, '0' and '1' denote the logic values outputted b the SA. Finally, $R = ?$ denotes a particular case in which the SA's input is too small for it to sense the cell's content correctly, resulting in a random readout value.

For example, the fault $\langle 1r1/0/? \rangle$ denotes a failed read '1' operation that flips the cell's content to '0' and returns a random readout value. Similarly, both the FPs $\langle 0w1/0_r/- \rangle$ and $\langle 0w1/0/- \rangle$ represent a failed write transition from '0' to '1'. The first FP is modified with a temporal component to indicate that the cell's impact occurs after a time interval longer than one memory operation, i.e., the cell's content goes back to '0' after some time following the write operation. On the other hand, the second fault does not include a temporal component; the cell's content is still '0' right after the write operation.

3.2. FAULT SPACE DEFINITION – STATIC FAULTS

The fault space definition is the first step in fault modeling. A fault space is defined by modeling possible faults that can occur in a given circuit; it is complete if it contains *all* faults that could occur in this circuit [26]. In this section, we explore the fault space of static memory faults, i.e., fault sensitized by *at most* one operation.

3.2.1. STATIC MEMORY ARRAY FAULTS

These faults can be directly pointed to a specific cell or group of cells. They can be divided into functional single-cell, functional coupling, and parametric faults.

FUNCTIONAL SINGLE-CELL FAULTS

Static functional single-cell faults comprise *all* faults affecting one single cell that is triggered by *at most* one operation and *may* impact the cell's functionality. These faults can be described using FPs that combine all possible S, F, and R. Therefore, the space is complete if it contains *all* possible combinations. Table 3.1 lists the complete fault space for static, functional, single-cell faults, alongside the name of each fault. Note that we do not include the *nat* modifier as it does not change the name of a fault. The name of a fault is also defined based on the combination of its S, F, and R. For $n = 0$, FPs are named as

$$\{ini\}F\{fin\}_{\{nat\}}, \tag{3.1}$$

while for $n = 1$ FP are described as

$$\{out\}\{opn\}\{opd\}\{eff\}F\{fin\}_{\{nat\}}. \tag{3.2}$$

Each field in the name template (3.1) and (3.2) describes a specific fault characteristic as follows:

- ini describes the cell's initial state; $ini \in \{0,1\}$.

- fin describes the cell's final state; $fin \in \{0,\text{U},1\}$.

Table 3.1: Complete space of static, functional, single-cell faults.

#	S	F	R	FP Notation	FP Name	Functional Fault Model
1	0	1	-	$\langle 0/1/- \rangle$	S0F1	State Fault
2	0	U	-	$\langle 0/U/- \rangle$	S0FU	
3	1	0	-	$\langle 1/0/- \rangle$	S1F0	
4	1	U	-	$\langle 1/U/- \rangle$	S1FU	
5	0w1	0	-	$\langle 0w1/0/- \rangle$	W1TF0	Write Transition Fault
6	0w1	U	-	$\langle 0w1/U/- \rangle$	W1TFU	
7	1w0	1	-	$\langle 1w0/1/- \rangle$	W0TF1	
8	1w0	U	-	$\langle 1w0/U/- \rangle$	W0TFU	
9	0w0	1	-	$\langle 0w0/1/- \rangle$	W0DF1	Write Disturb Fault
10	0w0	U	-	$\langle 0w0/U/- \rangle$	W0DFU	
11	1w1	0	-	$\langle 1w1/0/- \rangle$	W1DF0	
12	1w1	U	-	$\langle 1w1/U/- \rangle$	W1DFU	
13	0r0	0	1	$\langle 0r0/0/1 \rangle$	iR0NF0	Incorrect Read Non-Destructive Fault
14	1r1	1	0	$\langle 1r1/1/0 \rangle$	iR1NF1	
15	0r0	0	?	$\langle 0r0/0/? \rangle$	rR0NF0	Random Read Non-Destructive Fault
16	1r1	1	?	$\langle 1r1/1/? \rangle$	rR1NF1	
17	0r0	1	1	$\langle 0r0/1/1 \rangle$	iR0DF1	Incorrect Read Destructive Fault
18	0r0	U	1	$\langle 0r0/U/1 \rangle$	iR0DFU	
19	1r1	0	0	$\langle 1r1/0/0 \rangle$	iR1DF0	
20	1r1	U	0	$\langle 1r1/U/0 \rangle$	iR1DFU	
21	0r0	1	0	$\langle 0r0/1/0 \rangle$	dR0DF1	Deceptive Read Destructive Fault
22	0r0	U	0	$\langle 0r0/U/0 \rangle$	dR0DFU	
23	1r1	0	1	$\langle 1r1/0/1 \rangle$	dR1DF0	
24	1r1	U	1	$\langle 1r1/U/1 \rangle$	dR1DFU	
25	0r0	1	?	$\langle 0r0/1/? \rangle$	rR0DF1	Random Read Destructive Fault
26	0r0	U	?	$\langle 0r0/U/? \rangle$	rR0DFU	
27	1r1	0	?	$\langle 1r1/0/? \rangle$	rR1DF0	
28	1r1	U	?	$\langle 1r1/U/? \rangle$	rR1DFU	

- nat is an optional modifier based on the fault's nature. For this work, the subscript '$_r$' is used to specify retention faults, i.e., the fault's impact on the cell's final state fin occurs after a time interval longer than one memory operation.

- out describes the read operation's output; $out \in \{\mathrm{i}, \mathrm{r}, \mathrm{d}\}$, where 'i' means an incorrect read output, 'r' a random read output, and 'd' a deceptive read output (i.e., $ini \neq fin$). This field is disregarded if a write operation is the last operation in S.

- opn describes the last operations in S; $opn \in \{\mathrm{W},\mathrm{R}\}$, where 'W' stands for a write operation while 'R' stands for a read operation.

- opd describes the operand of the operation opn; $opd \in \{0, 1\}$.

- eff describes the operation's effect on the faulty cell; $eff \in \{\mathrm{T},\mathrm{D},\mathrm{N}\}$, where 'T' denotes a transition operation (i.e., $S{=}0w1$ or $S{=}1w0$), 'D' denotes a destructive operation, and 'N' denotes a non-destructive operation.

Table 3.1 also groups the faults into eight different *functional fault models* (FFMs) [135], i.e., non-empty sets of FPs with similar properties. These FFMs can be described as follows:

1. **State Fault:** the logic value stored in a cell flips or is destroyed before accessing the cell. It is a particular case as no operation is required to sensitize it and, therefore, only depends on the initial stored value in the cell. It consists of 4 FPs: $\langle 0/1/\text{-}\rangle$, $\langle 0/\mathrm{U}/\text{-}\rangle$, $\langle 1/0/\text{-}\rangle$, and $\langle 1/\mathrm{U}/\text{-}\rangle$

2. **Write Transition Fault:** a cell fails to undergo a transition, e.g., from ‘0’ to ‘1’ or from ‘1’ to ‘0’, when it is written. This FFM is sensitized by a *transition* write operation and depends on both the initial stored logic value and the value to be written. The cell’s content may either stay the same, or be destroyed, i.e., ‘U’. It consists of 4 FPs: $\langle 0w1/0/\text{-}\rangle$, $\langle 0w1/\mathrm{U}/\text{-}\rangle$, $\langle 1w0/1/\text{-}\rangle$, and $\langle 1w0/\mathrm{U}/\text{-}\rangle$.

3. **Write Disturb Fault:** a non-transition write operation, e.g., $0w0$, $1w1$, either reverses the logic value stored in the cell or destroys it. It also depends on the initial stored logic value and the value to be written. It consists of 4 FPs: $\langle 0w0/1/\text{-}\rangle$, $\langle 0w0/\mathrm{U}/\text{-}\rangle$, $\langle 1w1/0/\text{-}\rangle$, and $\langle 1w1/\mathrm{U}/\text{-}\rangle$.

4. **Incorrect Read Non-Destructive Fault:** a read operation returns the opposite of the expected logic value. The operation does not flip nor destroy the cell’s content. It consists of 2 FPs: $\langle 0r0/0/1\rangle$ and $\langle 1r1/1/0\rangle$.

5. **Random Read Non-Destructive Fault:** a read operation returns a random logic value; this happens when the SA’s input is too small, i.e., below its safe margin. The cell’s content stays the same. It consists of 2 FPs: $\langle 0r0/0/?\rangle$ and $\langle 1r1/1/?\rangle$.

6. **Incorrect Read Destructive Fault:** a read operation either flips the cell’s content to the opposite logic value, e.g., from ‘0’ to ‘1’, or destroys the cell’s content, e.g., from ‘0’ to ‘U’. Furthermore, it also outputs the opposite of the expected logic value. It consists of 4 FPs: $\langle 0r0/1/1\rangle$, $\langle 0r0/\mathrm{U}/1\rangle$, $\langle 1r1/0/0\rangle$, and $\langle 1r1/\mathrm{U}/0\rangle$.

7. **Deceptive Read Destructive Fault:** a read operation either flips the cell’s content to the opposite logic value, e.g., from ‘0’ to ‘1’, or destroys the cell’s content, e.g., from ‘0’ to ‘U’. Nevertheless, it does output the expected logic value. Therefore, a second read operation is necessary to detect the fault. It consists of 4 FPS: $\langle 0r0/1/0\rangle$, $\langle 0r0/\mathrm{U}/0\rangle$, $\langle 1r1/0/1\rangle$, and $\langle 1r1/\mathrm{U}/1\rangle$.

8. **Random Read Destructive Fault:** a read operation either flips the cell’s content to the opposite logic value, e.g., from ‘0’ to ‘1’, or destroys the cell’s content, e.g., from ‘0’ to ‘U’. Furthermore, the read operation also outputs a random value. It consists of 4 FPs: $\langle 0r0/1/?\rangle$, $\langle 0r0/\mathrm{U}/?\rangle$, $\langle 1r1/0/?\rangle$, and $\langle 1r1/\mathrm{U}/?\rangle$.

FUNCTIONAL COUPLING FAULTS

The space of functional coupling faults comprise *all* faults in which an aggressor cell C_a has a direct influence in a faulty behavior in a victim cell C_v; this faulty behavior may lead to incorrect functional behavior. Likewise to single-cell faults, coupling faults can also be described using FPs that combine S_a, S_v, F, and R. Note that the cases in which S_a and S_v contain operations executed at the same time are not covered in this thesis; they are therefore not included in the table. Table 3.2 lists the fault space for static, functional, coupling faults; it consists of 80 different FPs [135]. The table does not include FP names,

Table 3.2: Complete space of static, functional, coupling faults.

#	S_a	S_v	F	R	FP Notation	Functional Fault Model	#	S_a	S_v	F	R	FP Notation
1	0	0	1	-	⟨0;0/1/-⟩	State Coupling Fault (CFst)	5	1	0	0	-	⟨1;0/0/-⟩
2	0	0	U	-	⟨0;0/U/-⟩		6	1	0	U	-	⟨1;0/U/-⟩
3	0	1	0	-	⟨0;1/0/-⟩		7	1	1	0	-	⟨1;1/0/-⟩
4	0	1	U	-	⟨0;1/U/-⟩		8	1	1	U	-	⟨1;1/U/-⟩
9	0w1	0	1	-	⟨0w1;0/1/-⟩	Disturb Coupling Fault (CFds)	21	0w1	1	0	-	⟨0w1;1/0/-⟩
10	0w1	0	U	-	⟨0w1;0/U/-⟩		22	0w1	1	U	-	⟨0w1;1/U/-⟩
11	1w0	0	1	-	⟨1w0;0/1/-⟩		23	1w0	1	0	-	⟨1w0;1/0/-⟩
12	1w0	0	U	-	⟨1w0;0/U/-⟩		24	1w0	1	U	-	⟨1w0;1/U/-⟩
13	0w0	0	1	-	⟨0w0;0/1/-⟩		25	0w0	1	0	-	⟨0w0;1/0/-⟩
14	0w0	0	U	-	⟨0w0;0/U/-⟩		26	0w0	1	U	-	⟨0w0;1/U/-⟩
15	1w1	0	1	-	⟨1w1;0/1/-⟩		27	1w1	1	0	-	⟨1w1;1/0/-⟩
16	1w1	0	U	-	⟨1w1;0/U/-⟩		28	1w1	1	U	-	⟨1w1;1/U/-⟩
17	0r0	0	1	-	⟨0r0;0/1/-⟩		29	0r0	1	0	-	⟨0r0;1/0/-⟩
18	0r0	0	U	-	⟨0r0;0/U/-⟩		30	0r0	1	U	-	⟨0r0;1/U/-⟩
19	1r1	0	1	-	⟨1r1;0/1/-⟩		31	1r1	1	0	-	⟨1r1;1/0/-⟩
20	1r1	0	U	-	⟨1r1;0/U/-⟩		32	1r1	1	U	-	⟨1r1;1/U/-⟩
33	0	0w1	0	-	⟨0;0w1/0/-⟩	Write Transition Coupling Fault (CFtr)	37	1	0w1	0	-	⟨1;0w1/0/-⟩
34	0	0w1	U	-	⟨0;0w1/U/-⟩		38	1	0w1	U	-	⟨1;0w1/U/-⟩
35	0	1w0	1	-	⟨0;1w0/1/-⟩		39	1	1w0	1	-	⟨1;1w0/1/-⟩
36	0	1w0	U	-	⟨0;1w0/U/-⟩		40	1	1w0	U	-	⟨1;1w0/U/-⟩
41	0	0w0	0	-	⟨0;0w0/0/-⟩	Write Disturb Coupling Fault (CFwd)	45	1	0w0	0	-	⟨1;0w0/0/-⟩
42	0	0w0	U	-	⟨0;0w0/U/-⟩		46	1	0w0	U	-	⟨1;0w0/U/-⟩
43	0	1w1	1	-	⟨0;1w1/1/-⟩		47	1	1w1	1	-	⟨1;1w1/1/-⟩
44	0	1w1	U	-	⟨0;1w1/U/-⟩		48	1	1w1	U	-	⟨1;1w1/U/-⟩
49	0	0r0	0	1	⟨0;0r0/0/1⟩	Incorrect Read Non-Destructive Coupling Fault (CFir)	51	0	1r1	1	0	⟨0;1r1/1/0⟩
50	1	0r0	0	1	⟨1;0r0/0/1⟩		52	1	1r1	1	0	⟨1;1r1/1/0⟩
53	0	0r0	0	?	⟨0;0r0/0/?⟩	Random Read Non-Destructive Coupling Fault (CFrr)	55	1	0r0	0	?	⟨1;0r0/0/?⟩
54	0	1r1	1	?	⟨0;1r1/1/?⟩		56	1	1r1	1	?	⟨1;1r1/1/?⟩
57	0	0r0	1	1	⟨0;0r0/1/1⟩	Incorrect Read Destructive Coupling Fault (CFrd)	61	1	0r0	1	1	⟨1;0r0/1/1⟩
58	0	0r0	U	1	⟨0;0r0/U/1⟩		62	1	0r0	U	1	⟨1;0r0/U/1⟩
59	0	1r1	0	0	⟨0;1r1/0/0⟩		63	1	1r1	0	0	⟨1;1r1/0/0⟩
60	0	1r1	U	0	⟨0;1r1/U/0⟩		64	1	1r1	U	0	⟨1;1r1/U/0⟩
65	0	0r0	1	0	⟨0;0r0/1/0⟩	Deceptive Read Destructive Coupling Fault (CFdrd)	69	1	0r0	1	0	⟨1;0r0/1/0⟩
66	0	0r0	U	0	⟨0;0r0/U/0⟩		70	1	0r0	U	0	⟨1;0r0/U/0⟩
67	0	1r1	0	1	⟨0;1r1/0/1⟩		71	1	1r1	0	1	⟨1;1r1/0/1⟩
68	0	1r1	U	1	⟨0;1r1/U/1⟩		72	1	1r1	U	1	⟨1;1r1/U/1⟩
73	0	0r0	1	?	⟨0;0r0/1/?⟩	Random Read Destructive Coupling Fault (CFrrd)	77	1	0r0	1	?	⟨1;0r0/1/?⟩
74	0	0r0	U	?	⟨0;0r0/U/?⟩		78	1	0r0	U	?	⟨1;0r0/U/?⟩
75	0	1r1	0	?	⟨0;1r1/0/?⟩		79	1	1r1	0	?	⟨1;1r1/0/?⟩
76	0	1r1	U	?	⟨0;1r1/U/?⟩		80	1	1r1	U	?	⟨1;1r1/U/?⟩

as the previously presented nomenclature scheme only applies to functional single-cell faults.

1. **State Coupling Fault (CFst):** C_v is forced into a given logic state if C_a is in a given state. No operation is needed to sensitize CFst; it only depends on the initial stored values in C_a and C_v. It consists of 8 FPs, from #1 to #8.
2. **Disturb Coupling Fault (CFds):** an operation (write or read) performed on C_a causes C_vś content to either flip or be destroyed. Any operation performed C_a is accepted as a sensitizing operation for the fault, be it a read, a transition write, or a non-transition write operation. CFds consists of 24 FPs, from #9 to #32.
3. **Write Transition Coupling Fault (CFtr):** C_v fails to undergo a transition write operation, e.g., from ‘0’ to ‘1’ or ‘1’ to ‘0’, if C_a stores a specific logic value. This fault

is sensitized by first setting C_a in a given state and then applying a write operation on C_v. It consists of 8 FPs, from #33 to #40.

4. **Write Disturb Coupling Fault (CFwd):** given that C_a stores a specific logic value, a non-transition write operation in C_v either reverses the logic value stored in the cell or destroys it. It consists of 8 FPs, from #41 to #48.
5. **Incorrect Read Non-Destructive Coupling Fault (CFir):**a read operation in C_v returns the expected opposite logic value if C_a stores a specific logic value. The operation does not flip nor destroy the cell's content. It consists of 4 FPs, from #49 to #52.
6. **Random Read Non-Destructive Coupling Fault (CFrr):** a read operation in C_v returns a random logic value as output if C_a is in a given state. The cell's content stays the same. It consists of 4 FPs, from #53 to #56.
7. **Incorrect Read Destructive Coupling Fault (CFrd):** a read operation in C_v either flips the cell's content to the opposite logic value, e.g., from '0' to '1', or destroys the cell's content, e.g., from '0' to 'U', if C_a is in a given state. Furthermore, it also outputs the opposite of the expected logic value. CFrd consists of 8 FPs, from #57 to #64.
8. **Deceptive Read Destructive Coupling Fault (CFdrd):** a read operation in C_v either flips the cell's content to the opposite logic value, e.g., from '0' to '1', or destroys the cell's content, e.g., from '0' to 'U', if C_a stores a specific logic value. Nevertheless, the read operation does output the expected logic value. Therefore, a second read operation is necessary to detect the fault. It consists of 8 FPS, from #65 to #72.
9. **Random Read Destructive Coupling Fault (CFrrd):** a read operation in C_v either flips the cell's content to the opposite logic value, e.g., from '0' to '1', or destroys the cell's content, e.g., from '0' to 'U', given that C_a stores a specific logic value. Furthermore, the read operation also outputs a random value. It consists of 8 FPs, from #73 to #80.

PARAMETRIC FAULTS

While it is important to understand functional faults' behavior, it is equally important to understand that manufacturing defects may also lead to parametric faulty behavior. This type of faulty behavior has been discussed in recent years in modeling and simulation of analog faults [138]. For SRAMs, a parametric faulty behavior means that a cell's performance characteristics are outside the expected range due to a small defect or extreme PV, thus failing one of its parametric specs. Despite no functional impact, parametric faults may lead to early in-field failure and must be considered, especially for task-critical applications such as aerospace and automotive.

Unlike functional faults, it is impossible to define a complete fault space for parametric faults in SRAMs due to the number of parameters involved in such a circuit. Therefore, we have selected a set of critical circuit parameters to define parametric faults. Specifically, when one of the selected parameters is out of the predefined spec, we refer to it as a parametric fault. These parameters are listed in Table 3.3. We do not define

Table 3.3: Fault space of static parametric faults.

#	Parameter	Fault Model
29	BLS	Reduced Bitline Swing
30	SNM	Reduced Noise Margin
31	RNM	
32	PCH	Increased Power Consumption
33	PCR	
34	PCTW	
35	PCNW	

(a) Fault 1, No Cell Fault.

(b) Fault 2, No Address Fault.

(c) Fault 3, Multiple Cell Fault.

(d) Fault 4, Multiple Address Fault.

Figure 3.2: Static faulty behaviors in an address decoders [139].

names for parametric faults. Instead, we list the parameter and group them into bigger parameter groups. BLS stands for *Bitline Swing*, i.e., the voltage difference between BL and $\overline{BL}$ during a read operation that is used as the input to the SA. SNM and RNM stand for *Static Noise Margin* and *Read Noise Margin* of the cell, which is the minimum noise voltage able to flip the cell's state during hold mode or a read operation, respectively. Finally, PCH, PCR, PCTW, PCNW denote the *Power Consumption* (PC) during four scenarios: hold mode (H), a read operation (R), a transition write operation (TW), and a non-transition write operation (NW). While we have limited our work to these specific parameters, any parameter could be included in the fault space as long as it has a performance specification.

3.2.2. STATIC DECODER FAULTS

Static address decoder faults refer to those faults in the address decoder sensitized using at most one operation. Faults in the row and column decoder can be modeled similarly, even though faults in each block affect different cell groups. Therefore, in this thesis, we assume that faults in the address decoder fault space can affect both the row and column decoder.

An address decoder may be affected by four types of faulty functional behaviors [139], as shown in Fig. 3.2. They can be explained as follows:

- **Fault 1, No Cell:** no memory cell is accessed with a particular address $\mathrm{A_x}$;
- **Fault 2, No Address:** a particular memory cell $\mathrm{C_x}$ is not accessed by any address;
- **Fault 3, Multiple Cell:** a particular address $\mathrm{A_y}$ accesses more than one memory cell, e.g., $\mathrm{C_x}$ and $\mathrm{C_y}$;
- **Fault 4, Multiple Address:** multiple addresses, e.g., $\mathrm{A_x}$ and $\mathrm{A_y}$, can access a particular memory cell $\mathrm{C_x}$.

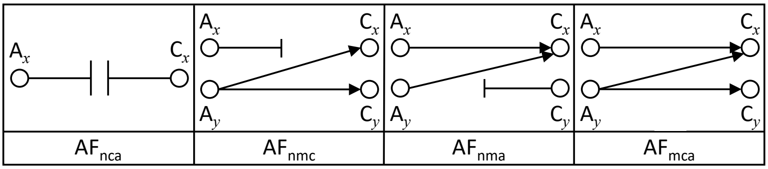


Figure 3.3: Static address decoder faults [140].

However, these faults do not occur by themselves but rather in pairs; therefore, it is a combination of both faulty behaviors. This leads to the traditional static fault space for address decoder [140], as shown in Fig. 3.3. These faults can be defined as follows:

1. **No Cell, No Address (AFnca):** the combination of faults 1 and 2. A particular address A_x does not access its cell C_x. Furthermore, C_x is not accessed by any other address.
2. **No Cell, Multiple Cells (AFnmc):** the combination of faults 1 and 3. A particular address A_x does not access its cell C_x. Nevertheless, a second address A_y accesses both its cell C_y, as well as C_x.
3. **No address, Multiple Addresses (AFnma):** the combination of faults 2 and 4. A particular address A_y does not access its cell C_y, but rather a different cell, e.g., C_x. C_x is also accessed by its own address, i.e., A_x.
4. **Multiple Cells, Multiple Addresses (AFmca):** the combination of faults 3 and 4. A particular address A_y accesses its own cell C_y and a second cell C_x simultaneously. C_x is also accessed by its own address, i.e., A_x.

3.2.3. STATIC WRITE AND READ PATH FAULTS

Static faults in the memory's write and read paths refer to those faults in the peripheral circuitry, e.g., *write driver* (WD), *sense amplifer* (SA), sensitized using at most one operation. Static fault models considered for the write and read path are the traditional models for logic circuits, such as stuck-at-zero and stuck-at-one [24]. Therefore, the static fault space for the write and read path consists of the following four static faults:

1. **Write Driver Stuck-at-one (WD-S-a-1):** the WD circuit only writes the logical value '1' into to-be written cells.
2. **Write Driver Stuck-at-zero (WD-S-a-0):** the WD circuit only writes the logical value '0' into to-be written cells.
3. **Sense Amplifier Stuck-at-one (SA-S-a-1):** the SA cannot read the cell's content correctly or store the sensed value properly; it always outputs '1'.
4. **Sense Amplifier Stuck-at-zero (SA-S-a-0):** the SA cannot read the cell's content correctly or store the sensed value properly; it always outputs '0'.

3.3. FAULT SPACE DEFINITION – DYNAMIC FAULTS

Dynamic faults are timing and related faults. Therefore, the number and order of operations, i.e., S, is crucial to trigger the fault. Furthermore, dynamic faults can also be associated with the four blocks previously discussed, i.e., memory cell array, address decoders, write path, and read. This section explores the fault space of dynamic memory faults, i.e., fault sensitized by more than one operation.

3.3.1. DYNAMIC MEMORY ARRAY FAULTS

These faults can be directly pointed to a specific cell or group of cells. The main difference between static and dynamic memory array faults is that the latter requires more than one operation to be applied sequentially to be sensitized. Once again, they can be divided into functional single-cell, functional coupling, and parametric faults.

FUNCTIONAL SINGLE-CELL FAULTS

Dynamic functional single-cell faults can also be described using the FP notation following the same scheme used for static faults. However, the FP's S now contains a sequence of operations with at least two consecutive operations. The number of operations depends on the technology and stressing conditions; while it has been shown for older DRAM technologies that four operations are enough to detect all dynamic faults [141], recent works have proposed applying up to eight consecutive operations for FinFET SRAMs [57]. Therefore, in this thesis, we embrace a more general approach – we describe dynamic faults sensitized by n-operation sequences. This way, we are not limited by the number of operations, and the fault space definition for dynamic faults becomes less redundant and more straightforward.

Table 3.4 lists the fault space for dynamic functional single-cell faults, alongside their FP name and FFM. The S element is described as $S = ...x_{n-1}O_nx_n$, where '...' is a sequence of $n-1$ operations that will put the cell into the x logic state, $x_i \in \{0,1\}$, $O \in \{r, w\}$, and n the number of operations necessary to trigger the fault. Since the fault model is dictated based on the fault that triggered the fault and the cell's state before and after this operation, the only operation relevant when defining the fault space is the last operation in S. Finally, Table 3.4 does not include dynamic state faults as they cannot exist.

FUNCTIONAL COUPLING FAULTS

Dynamic coupling faults are sensitized by applying more than one operation sequentially to two cells: the aggressor cell and the victim cell. As in the case of single-cell dynamic operations, we define the dynamic fault models as triggered by n-operation sequences. However, these n operations may be distributed in any combination among $\mathrm{C_a}$ and $\mathrm{C_v}$, e.g., all sensitizing operations are sequentially applied to the victim cell, or one operation is applied to the victim cell followed immediately by a second operation to the aggressor cell. In summary, the possible combinations extrapolate with n. For dynamic coupling faults in which the triggering operations, i.e., the last operation in S, is performed in $\mathrm{C_a}$ the sensitizing sequence is defined as $S = S_\mathrm{a} ; S_\mathrm{v} = ...x_{a,n-1}O_nx_{a,n}; ...x_{v,n}$, where $x_{a,n}$ and $x_{v,n}$ denote the state of $\mathrm{C_a}$ and $\mathrm{C_v}$ when applying operation O_n, respectively. '...' denotes past operations in $\mathrm{C_a}$ and $\mathrm{C_v}$. Contrarily, if the triggering operations

Table 3.4: Complete space of dynamic functional single-cell faults.

#	S	F	R	FP Notation	FP Name	Functional Fault Model
1	...0w1	0	-	$\langle ...0w1/0/\text{-}\rangle$	n-W1TF0	Dynamic Write Transition Fault
2	...0w1	U	-	$\langle ...0w1/U/\text{-}\rangle$	n-W1TFU	
3	...1w0	1	-	$\langle ...1w0/1/\text{-}\rangle$	n-W0TF1	
4	...1w0	U	-	$\langle ...1w0/U/\text{-}\rangle$	n-W0TFU	
5	...0w0	1	-	$\langle ...0w0/1/\text{-}\rangle$	n-W0DF1	Dynamic Write Disturb Fault
6	...0w0	U	-	$\langle ...0w0/U/\text{-}\rangle$	n-W0DFU	
7	...1w1	0	-	$\langle ...1w1/0/\text{-}\rangle$	n-W1DF0	
8	...1w1	U	-	$\langle ...1w1/U/\text{-}\rangle$	n-W1DFU	
9	...0r0	0	1	$\langle ...0r0/0/1\rangle$	n-iR0NF0	Dynamic Incorrect Read Non-Destructive Fault
10	...1r1	1	0	$\langle ...1r1/1/0\rangle$	n-iR1NF1	
11	...0r0	0	?	$\langle ...0r0/0/?\rangle$	n-rR0NF0	Dynamic Random Read Non-Destructive Fault
12	...1r1	1	?	$\langle ...1r1/1/?\rangle$	n-rR1NF1	
13	...0r0	1	1	$\langle ...0r0/1/1\rangle$	n-iR0DF1	Dynamic Incorrect Read Destructive Fault
14	...0r0	U	1	$\langle ...0r0/U/1\rangle$	n-iR0DFU	
15	...1r1	0	0	$\langle ...1r1/0/0\rangle$	n-iR1DF0	
16	...1r1	U	0	$\langle ...1r1/U/0\rangle$	n-iR1DFU	
17	...0r0	1	0	$\langle ...0r0/1/0\rangle$	n-dR0DF1	Dynamic Deceptive Read Destructive Fault
18	...0r0	U	0	$\langle ...0r0/U/0\rangle$	n-dR0DFU	
19	...1r1	0	1	$\langle ...1r1/0/1\rangle$	n-dR1DF0	
20	...1r1	U	1	$\langle ...1r1/U/1\rangle$	n-dR1DFU	
21	...0r0	1	?	$\langle ...0r0/1/?\rangle$	n-rR0DF1	Dynamic Random Read Destructive Fault
22	...0r0	U	?	$\langle ...0r0/U/?\rangle$	n-rR0DFU	
23	...1r1	0	?	$\langle ...1r1/0/?\rangle$	n-rR1DF0	
24	...1r1	U	?	$\langle ...1r1/U/?\rangle$	n-rR1DFU	

is performed in $\mathrm{C_v}$ then $S = S_{\mathrm{a}}$; $S_{\mathrm{v}} = ...x_{v,n}; x_{a,n-1}O_n x_{a,n}$. The fault space for dynamic functional coupling faults is listed in Table 3.5.

PARAMETRIC FAULTS

Dynamic parametric faults are spec deviations triggered by performing operations in the cell. For example, a sequence of read operations on the cell may cause a voltage deviation on the storing nodes, thus causing a reduction in the cell's SNM and an increased I_{off}. Once again, the fault space comprises possible parameters that could suffer deviations; defining a complete fault space is unfeasible due to the number of parameters involved in an SRAM circuit. We consider the same parameters as listed previously in Table 3.3. However, dynamic faults receive the $n-$ notation before their designed name to indicate they are triggered after applying n consecutive operations.

3.3.2. DYNAMIC DECODER FAULTS

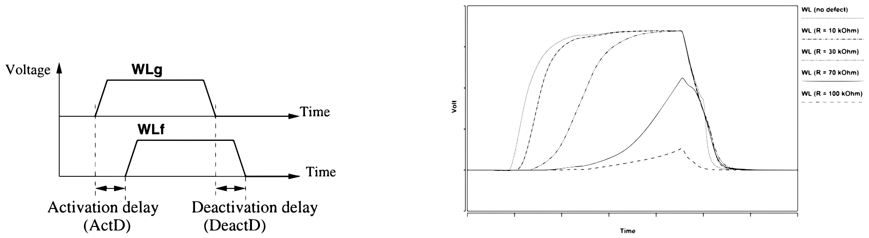
Dynamic address decoder faults cause delays in the selection of rows and columns, leading to *activation delay* (ActD) faults and *deactivation delay* (DeactD) faults [142]. While an ActD fault affects the rising edge of WL or *column select* (CS) signal due to resistive defects, a DeactD fault affects these signals' falling edge. The impact of such faults depends heavily on the memory's organization; we illustrate such dependency by analyzing the impact on the WL signal. If the memory is self-timed, i.e., the WL is only deactivated after it has been enabled for some time, a dynamic address decoder fault only shifts the WL's activation period. Such faulty behavior is shown in Fig. 3.4a [140]; while $\mathrm{WL_g}$ de-

Table 3.5: Complete space of dynamic functional coupling faults.

#	S_a	S_v	F	R	FP Notation	Functional Fault Model	#	S_a	S_v	F	R	FP Notation
1	...0w1	...0	1	-	⟨...0w1;...0/1/-⟩	Dynamic Disturb Coupling Fault (dCFds)	13	...0w1	...1	0	-	⟨...0w1;...1/0/-⟩
2	...0w1	...0	U	-	⟨...0w1;...0/U/-⟩		14	...0w1	...1	U	-	⟨...0w1;...1/U/-⟩
3	...1w0	...0	1	-	⟨...1w0;...0/1/-⟩		15	...1w0	...1	0	-	⟨...1w0;...1/0/-⟩
4	...1w0	...0	U	-	⟨...1w0;...0/U/-⟩		16	...1w0	...1	U	-	⟨...1w0;...1/U/-⟩
5	...0w0	...0	1	-	⟨...0w0;...0/1/-⟩		17	...0w0	...1	0	-	⟨...0w0;...1/0/-⟩
6	...0w0	...0	U	-	⟨...0w0;...0/U/-⟩		18	...0w0	...1	U	-	⟨...0w0;...1/U/-⟩
7	...1w1	...0	1	-	⟨...1w1;...0/1/-⟩		19	...1w1	...1	0	-	⟨...1w1;...1/0/-⟩
8	...1w1	...0	U	-	⟨...1w1;...0/U/-⟩		20	...1w1	...1	U	-	⟨...1w1;...1/U/-⟩
9	...0r0	...0	1	-	⟨...0r0;...0/1/-⟩		21	...0r0	...1	0	-	⟨...0r0;...1/0/-⟩
10	...0r0	...0	U	-	⟨...0r0;...0/U/-⟩		22	...0r0	...1	U	-	⟨...0r0;...1/U/-⟩
11	...1r1	...0	1	-	⟨...1r1;...0/1/-⟩		23	...1r1	...1	0	-	⟨...1r1;...1/0/-⟩
12	...1r1	...0	U	-	⟨...1r1;...0/U/-⟩		24	...1r1	...1	U	-	⟨...1r1;...1/U/-⟩
25	...0	...0w1	0	-	⟨...0;...0w1/0/-⟩	Dyanmic Write Transition Coupling Fault (dCFtr)	29	...1	...0w1	0	-	⟨...1;...0w1/0/-⟩
26	...0	...0w1	U	-	⟨...0;...0w1/U/-⟩		30	...1	...0w1	U	-	⟨...1;...0w1/U/-⟩
27	...0	...1w0	1	-	⟨...0;...1w0/1/-⟩		31	...1	...1w0	1	-	⟨...1;...1w0/1/-⟩
28	...0	...1w0	U	-	⟨...0;...1w0/U/-⟩		32	...1	...1w0	U	-	⟨...1;...1w0/U/-⟩
33	...0	...0w0	0	-	⟨...0;...0w0/0/-⟩	Dyanmic Write Disturb Coupling Fault (dCFwd)	37	...1	...0w0	0	-	⟨...1;...0w0/0/-⟩
34	...0	...0w0	U	-	⟨...0;...0w0/U/-⟩		38	...1	...0w0	U	-	⟨...1;...0w0/U/-⟩
35	...0	...1w1	1	-	⟨...0;...1w1/1/-⟩		39	...1	...1w1	1	-	⟨...1;...1w1/1/-⟩
36	...0	...1w1	U	-	⟨...0;...1w1/U/-⟩		40	...1	...1w1	U	-	⟨...1;...1w1/U/-⟩
41	...0	...0r0	0	1	⟨...0;...0r0/0/1⟩	Dynamic Incorrect Read Non-Destructive Coupling Fault (dCFir)	43	...0	...1r1	1	0	⟨...0;...1r1/1/0⟩
42	...1	...0r0	0	1	⟨...1;...0r0/0/1⟩		44	...1	...1r1	1	0	⟨...1;...1r1/1/0⟩
45	...0	...0r0	0	?	⟨...0;...0r0/0/?⟩	Dynamic Random Read Non-Destructive Coupling Fault (dCFrr)	47	...1	...0r0	0	?	⟨...1;...0r0/0/?⟩
46	...0	...1r1	1	?	⟨...0;...1r1/1/?⟩		48	...1	...1r1	1	?	⟨...1;...1r1/1/?⟩
49	...0	...0r0	1	1	⟨...0;...0r0/1/1⟩	Dynamic Incorrect Read Non-Destructive Coupling Fault (dCFrd)	53	...1	...0r0	1	1	⟨...1;...0r0/1/1⟩
50	...0	...0r0	U	1	⟨...0;...0r0/U/1⟩		54	...1	...0r0	U	1	⟨...1;...0r0/U/1⟩
51	...0	...1r1	0	0	⟨...0;...1r1/0/0⟩		55	...1	...1r1	0	0	⟨...1;...1r1/0/0⟩
52	...0	...1r1	U	0	⟨...0;...1r1/U/0⟩		56	...1	...1r1	U	0	⟨...1;...1r1/U/0⟩
57	...0	...0r0	1	0	⟨...0;...0r0/1/0⟩	Dynamic Deceptive Read Destructive Coupling Fault (dCFdrd)	61	...1	...0r0	1	0	⟨...1;...0r0/1/0⟩
58	...0	...0r0	U	0	⟨...0;...0r0/U/0⟩		62	...1	...0r0	U	0	⟨...1;...0r0/U/0⟩
59	...0	...1r1	0	1	⟨...0;...1r1/0/1⟩		63	...1	...1r1	0	1	⟨...1;...1r1/0/1⟩
60	...0	...1r1	U	1	⟨...0;...1r1/U/1⟩		64	...1	...1r1	U	1	⟨...1;...1r1/U/1⟩
65	...0	...0r0	1	?	⟨...0;...0r0/1/?⟩	Dynamic Random Read Destructive Coupling Fault (dCFrrd)	69	...1	...0r0	1	?	⟨...1;...0r0/1/?⟩
66	...0	...0r0	U	?	⟨...0;...0r0/U/?⟩		70	...1	...0r0	U	?	⟨...1;...0r0/U/?⟩
67	...0	...1r1	0	?	⟨...0;...1r1/0/?⟩		71	...1	...1r1	0	?	⟨...1;...1r1/0/?⟩
68	...0	...1r1	U	?	⟨...0;...1r1/U/?⟩		72	...1	...1r1	U	?	⟨...1;...1r1/U/?⟩

picts the fault-free case, WL$_f$ depicts the behavior of a faulty WL suffering from both ActD and DeactD. However, if the WL deactivation moment is fixed, the ActD's impact changes significantly. Depending on the defect size, the activation may cause the WL to be partially enabled only or not be enabled at all. This behavior is shown in Fig. 3.4b [140], which shows multiple WL curves for different defect sizes; not that for $R > 70\,\mathrm{k}\Omega$, the WL is barely activated.

A key characteristic of dynamic address decoder faults is that only specific address transitions sensitize them. Therefore, the address order in which addresses are accessed is vital to triggering and detecting dynamic address decoder faults. These address transitions are defined as address pairs or address triplets. A *sensitizing address pair* (SAP) consists of a sequence of two addresses, e.g., {A_x, A_y}, which have to be applied in sequence to fulfill the requirements of dynamic address decoder faults. A *sensitizing address triplet* (SAP) consists of two SAPs, e.g., {A_x, A_y} and {A_y, A_x}, that are applied in sequence, i.e., {A_x, A_y, A_x}. The sequence required to sensitize the dynamic fault depends on the defect's location, e.g., pre-decoder, post-decoder, and the addressing scheme of the memory. It is widely accepted that using hamming-distance-based addressing methods significantly increases the sensitization of dynamic decoder faults [140], [142].



(a) Faulty behaviors in a self-timed WL signal.

(b) Faulty behaviors in a WL with fixed deactivation time.

Figure 3.4: Dynamic faulty behaviors in an address decoders [140].

3.3.3. DYNAMIC WRITE AND READ PATH FAULTS

Dynamic faults in the peripheral circuitry faults are timing-related faults in the peripheral circuitry, i.e., WDs, SAs, PCs [143]. These faults are sensitized by applying consecutive, opposite memory operations in the same write or read path; such sequence of operations is known as *back-to-back*. In-between operations in different BL pairs or idle time could reduce the effect of the first sensitizing write operation. The fault space for dynamic write and read paths is defined as follows:

1. **Slow Write Driver Fault (SWDF):** the write driver may be too slow due to defects in its driver or along the write path, resulting in a reduced differential voltage on BLs during the write operation; this may cause the cell not to be written. SWDFs are sensitized by applying two consecutive write operations in the same BL pair writing opposite logic values, e.g., $w1, w0$.

2. **Slow Precharge Fault (SPCF):** the PC may be slower than designed due to manufacturing defects, resulting in it not precharging BL and $\overline{BL}$ to the same voltage level. The partially-charged BLs may eventually cause the SA to misread the cell's content due to its high sensitivity to BL voltage offset errors. SPCFs are sensitized by applying a back-to-back operation sequence consisting of a write operation followed by a read operation, e.g., $w1, r0$; this sequence provides the most stress in the BL pair.

3. **Slow Sense Amplifier Fault (SSAF):** asymmetries caused by defects or extreme process variation may impair the SA's amplification and cause it to produce incorrect results. SSAF's sensitization requires applying back-to-back read operations, e.g., $r1, r0$. Therefore, it is first necessary to write two different cells in the same read path with opposite logic values and then apply the sensitizing read operations.

3.4. FAULT SPACE VALIDATION METHODOLOGY

As previously mentioned, fault modeling is divided into fault space definition and fault space validation. While fault space definition aims at identifying *all* possible faults that could (theoretically) occur, fault space validation uses simulation to predict circuit behavior or even real-life chip testing to verify what faults models occur in reality indeed. In this thesis, fault validation is achieved through a systematic circuit simulation methodology that reproduces numerous operating scenarios with multiple injected defects and stressing conditions. While this methodology is later used to validate the fault space concerning single-cell memory array faults only, the methodology can be used to validate any fault space by adjusting the set of injected defects and stressing conditions. A flowchart representation of the validation methodology is shown in Fig. 3.5; it consists of six steps:

1. **Netlist Generation:** an electrical-level netlist of a memory circuit is generated. This netlist contains the description of all memory blocks, e.g., array, decoders, peripherals. In this thesis, all netlists are described in SPICE. This step is a one-time procedure.
2. **Defect Injection:** a defect is introduced in the memory cell to sensitize faults one at a time. A defect set, i.e., a list with all defects to be injected into the memory, is used to control the defect injection procedure. Throughout this thesis, we have modeled the inject defects as resistor components. Depending on what faults are being targeted, they can be injected at any part of any memory block. Additionally, the number of operations in S and the defect size D_{size} are (re)set (i.e., $n = 0$, $D_{size} = 0\ \Omega$).
3. **Defect Size Sweep:** The resistor component's resistance is swept within the specified defect range. If $n \leq 1$, this range is $[0,+\infty)$, i.e., all possible defect sizes. If $n \geq 2$, this range consists of the defect sizes in which no faults were observed in previous iterations. For each iteration of sweep defect size, D_{size} is increased.
4. **Stimuli Generation:** based on n, the stimuli S is generated. It includes an initial condition and n memory operations. For $n = 0$, S can be $\{0,1\}$; for $n = 1$, S can be $\{0w0, 0w1, 1w0, 1w1, 0r0, 1r1\}$, and so forth. Furthermore, S can also contain operations in other rows, columns, and words, depending of the faults being targeted.
5. **Circuit Simulation:** the netlist including a defect, a defect size D_{size}, and stimuli S is simulated using a SPICE simulator. During the simulation, measuring commands capture memory electrical parameters, e.g., voltage, current, and read outputs at specific time instants. All measurements are logged and saved.
6. **Behavior Inspection:** once all defect sizes and stimuli for a given defect and n are simulated, an (automated) behavior inspection is carried out. This step analyzes all measurements from the circuit simulation step and identifies faults. The result is a report containing all observed faults alongside their respective defect size ranges.

All steps are performed sequentially. A loop between steps 2 and 6 guarantees that the methodology covers all defects, defect sizes, and stimuli combinations.

The Behavior Inspection is the methodology's most critical step as it translates the measured electrical behaviors to the faulty behaviors defined in the fault space. While it is easy to identify when the defect causes an incorrect logic behavior in the memory,

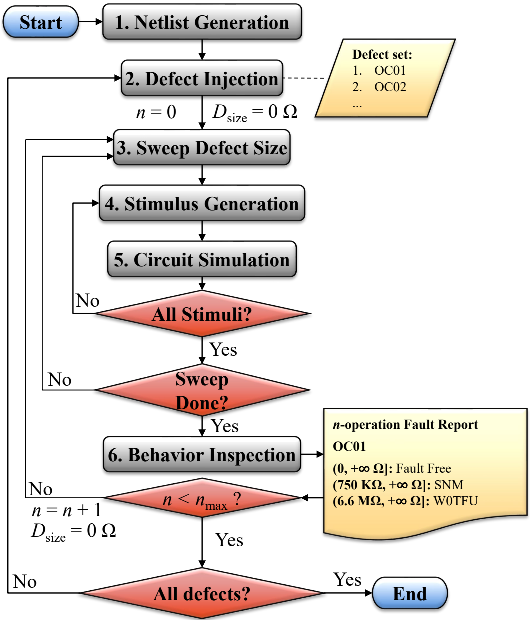


Figure 3.5: Fault space validation methodology.

the occurrence of parametric faults due to severe parametric deviation, random reads, and undefined states is strongly dependent on the memory's specifications. Therefore, it is essential to define the memory's spec in a manner that will accurately reflect the occurrence of faults.

A parametric fault occurs when one of the memory's parameters, e.g., BLS, noise margin, power consumption, fails its specification, i.e., its performance characteristics are outside its expected range. In the context of analog faults, it has been reported that this range is typically three standard deviations (σ) of the mean (μ) of a given parameter [138]. Nevertheless, it is common for memory engineers to design SRAMs that withstand parameter variations of 5σ or even 6σ [144]. In this thesis, we only want to detect *extrinsic* variations, i.e., variations caused by manufacturing defects, that surpass the typical range of *intrinsic* variations, i.e., variations caused by PV. Therefore, we define the threshold between intrinsic and extrinsic variation, i.e., PV-induced variation and defect-induced variation, respectively, as $\pm 6\sigma$. Accordingly, deviations within the $\mu \pm 6\sigma$ range are considered intrinsic variations; parametric deviations outside this range are considered extrinsic variations and consequently parametric faults. A too relaxed threshold causes intrinsic variations to be classified as extrinsic, i.e., a defect-free device is signaled as defective, leading to yield loss. Note that the definition of this range is flexible and can change based on the application; critical applications may not tolerate a variation of 3σ, and thus variations greater than 3σ are already flagged as faulty. Notwithstanding, changing this range modifies only the resistance boundaries in which a defect sensitizes a fault; the overall fault trends are expected to remain the same.

The definition of random reads and undefined faults are dependent on the definition of parametric faults. Because of PV, the SA will be skewed to either output ‘1’ or ‘0’, i.e., there is a voltage offset between the two storing nodes that will bias the output result. A deterministic output is guaranteed if the SA's input is bigger than its offset. Therefore, the spec for SA's input can be determined by estimating the mean offset variation and then extrapolating to match the determined σ. A random read fault occurs when the BLS is smaller than the SA's spec for minimum input. The identification of a random read occurs in two steps. First, the BLS is defined by calculating BLS = $|BL$ - $\overline{BL}|$ when WL = V_{DD}/ 2 during a read operation. Then, the measured BLS is compared to the SA's input specification; if it extrapolates the 6σ variation defined for parametric faults, i.e., BLS < |SA's input spec - 6σ|, the faulty behavior is defined as a random read fault. It is important to note that this fault is related to the SA's input spec, which is, in turn, related to the SA's offset variation; it is not related to the BLS spec.

As for undefined state fault, it occurs when the voltage difference between the cell's storing nodes is smaller than a predefined threshold. Previous works that have dealt with undefined state faults in older SRAM technologies have not defined this threshold [47], [135]. The acceptable ΔV between storing nodes changes from application to application; non-critical applications may tolerate a small ΔV, while critical applications, e.g., automotive and aerospace, may require a greater ΔV. For this work, we identify faults through the perspective of a critical application with more strict requirements. Therefore, we specify this threshold as half V_{DD}. Thus, a cell is in an undefined state if $|V(Q)-V(\overline{Q})| \leq V_{DD}/2$.

Once the behavior inspection for a given defect (including defect size sweep and all n-operation stimuli) has finished, steps 3 to 6 are repeated for $n+1$. The defect size D_{size} is reset to 0 Ω, and the defect size range and S are adjusted. If $n < 2$, the defect size range is set to $[0, +\infty)$. If $n \geq 2$, the defect size range is adjusted based on previous iterations' reports; only defect sizes in which no faults were observed are analyzed. The lower bound of the defect size range is the smallest defect size possible, and the upper bound is the smallest defect size that triggered a fault. For example, consider that for a given defect, faults were observed in the range $[50\ k\Omega, +\infty)$ when $n = 1$. For $n = 2$, the defect size range investigated will then be limited to $[0,\ 50\ k\Omega]$.

Results from previous iterations are also used to limit the stimuli. Since it is infeasible to investigate all possible combinations of memory operations for large values of n, S must be limited. Thus, only operations that trigger unexpected behaviors are used as the base of S. For example, consider that for a given defect, no faults were observed when $n = 1$. However, two unexpected behaviors were observed: a prolonged transition when applying the sequence $0w1$ and a slightly reduced BLS when applying the sequence $0r0$. Such behaviors can be an indication of dynamic faulty behavior. Thus, for $n = 2$, $0w1$ and $0r0$ will be used as the base of S. Only the sensitizing sequences $\{0w1w1, 0w1w0, 0w1r1, 0r0w1, 0r0w0, 0r0r0\}$ are applied to the defective cell; remaining combinations of S for $n = 2$ are excluded from the analysis. This process is repeated until n reaches a user-defined maximum number of operations, i.e., n_{max}. The fault space validation finishes when this process has been repeated for all the defects in the defect set.

3.5. FAULT MODELING RESULTS FOR SINGLE-CELL MEMORY ARRAY FAULTS

In this section, we validate the fault space of single-cell memory array faults through SPICE-based circuit simulations; we validate this specific fault space as it includes a diverse set of FFMs and does not require complex operation sequences involving two or more cells. While this methodology is later used to validate the single-cell memory array fault space concerning only, the methodology can be used to validate any fault space by adjusting the set of injected defects and stressing conditions. First, we introduce our simulation setup, including details about the simulated circuits, the Monte Carlo analysis, and the injected defects. We then present the simulation results from the Monte Carlo analysis to generate the memory's baseline metrics and define the thresholds of parametric variations. Finally, we present the results obtained following the 6-step validation methodology.

3.5.1. SIMULATION SETUP

The memory netlist is described in SPICE using the BSIM-CMG model [145] and the PTM 14 nm FinFET library [146]. The FinFET device configuration is a bulk substrate, trigate, with asymmetries in all structures and materials, while the used bit cell configuration is 1:2:2 for high performance. The memory array comprises 128 rows and 64 columns where each logical word contains 32 bits; hence, two neighboring columns share a single write driver, SA, and prechargers. Capacitive loads are applied to BLs and WLs to emulate a 1 kB memory. The memory works on a nominal supply voltage of 0.8 V and a clock frequency of 2 GHz. Additional timing circuits are used to generate control signals.

The same memory model is used for spec identification and fault modeling. The spec identification aims to identify the memory's baseline metrics and estimate the impact of PV on the memory's parameters. This is achieved through thousands of *Monte Carlo* (MC) simulations, i.e., simulations in which parameters (physical or electrical) are randomly varied. In this work, we have introduced a voltage source on the gate terminal of transistors [147] to emulate the PV's impact on V_{th} variation at time zero [148]. First, we perform MC simulations to estimate the memory's baseline metrics, i.e., the mean μ, and how much PV impacts μ, i.e., the standard deviation σ. We simulated the memory operating under four distinct scenarios: hold mode (i.e., idle), read operation, non-transition write operation, and transition write operation; each scenario is simulated 10,000 times. We measured different operating parameters in each scenario, such as power consumption, BLS, and noise margin. The μ value derived from measurements is defined as the parameter's spec. The σ is then calculated based on the spec and used to define the parametric deviation threshold.

The second analysis comprises injecting defects and inspecting the electrical behavior under different sensitizing sequences. The injected defects consist of twenty-eight single-cell resistive defects, as shown in Fig. 3.6. They are either *Resistive-Open* (RO), *Resistive-Short* (RS), or *Resistive-Bridge* (RB) defects [135]. RO defects are unintended series resistances within an existing connection. They are labeled as *Open Connections* (OC) 01 to 12. RSs and RBs are unintended resistive connections between two nodes. In more detail, RSs are shorts to power nodes (V_{DD} or GND); they are named *Short in the*

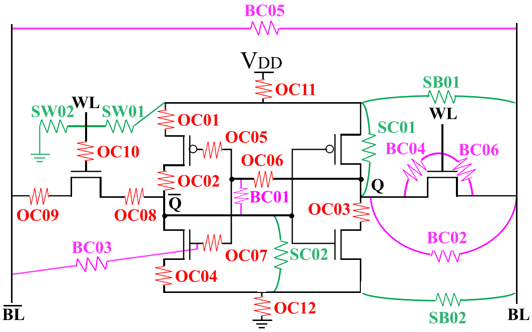


Figure 3.6: Injected resistive defects: opens, shorts, bridges.

Cell (SC) 01 and 02, *Short in the BL* (SB) 01 and 02, and *Short in the WL* (SW) 01 and 02. In contrast, RBs are shorts between any two other cell nodes. They are identified as *Bridges Connections* (BC) 01 to 03. Due to the symmetry of 6T cells, all defects have symmetrical opposites, e.g., OC01 can be injected on the pull-up of either $\overline{Q}$ or Q; defect BC01 is an exceptional case to this rule. Symmetrical opposite defects will lead to a similar faulty behavior to their counterparts and are therefore neglected in our analysis.

Each simulation comprises one defect, one defect size, and one sensitizing sequence S of n operations. We have set an n_{max} of 30 operations, i.e., S contains at most 30 operations. Each combination of a resistive defect and a sensitizing sequence was simulated with (at most) 100 different defect sizes by sweeping the resistance value from 0 Ω to 100 GΩ (representing $+\infty$), logarithmically spaced.

3.5.2. Baseline Metrics and Spec Identification

This section presents the results obtained from the Monte Carlo analyses' measurements; we aim to obtain the mean and standard deviation of the operating parameters, e.g., BLS, SNM, to define the memory's baseline metrics. Table 3.6 shows a summary of the measured parameters, their mean (μ), their standard variation (σ), and the condition to determine a parametric fault. The last represents the threshold between a parameter deviated by (extreme) PV and a deviation caused by a manufacturing defect; it is defined as $\mu \pm 6\sigma$. It is worth noting that the Monte Carlo analysis did not lead to functional faults; only parametric deviations were observed. Furthermore, parametric deviations outside the $\pm 6\sigma$ range were not observed. Therefore, it is safe to assume that the parametric faults identified during the defect injection campaign are caused by manufacturing defects rather than PV. Finally, note that reducing the $\pm 6\sigma$ range could lead to the incorrect indication of PV-induced variation as defect-induced.

3.5.3. Results

This section presents the results from the reports generated on the *Behavior Inspection* step of the fault space validation methodology. We analyze all defects assuming wide size ranges and identify the memory array faults sensitized by each defect. Table 3.7 shows a summary of the observed *static* faults by each defect. It is divided into two segments: the

Table 3.6: Results of the Monte Carlo analysis.

Parameter	Mean (μ)	Standard Deviation (σ)	Condition for Parametric Fault
BLS	168 mV	7.58 mV	BLS < 122.5 mV
SNM	347.2 mV	5.41 mV	SNM < 341.74 mV
RNM	191 mV	8.01 mV	RNM < 141.92 mV
PCH	43.19 pW	4.4 pW	PCH > 70.4 pW
PCR	63.0 µW	2.1 µW	PCR > 75.9 µW
PCTW	102.2 µW	2.7 µW	PCTW > 118.5 µW
PCNW	27.2 µW	6.2 µW	PCNW > 64.1 µW

top part lists the functional single-cell FPs, while the bottom lists parametric faults. All the FFMs previously listed for static single-cell faults have been observed, with at least one FP of each FFM being sensitized; FPs that were not sensitized are not included in Table 3.7. Some functional faults could have been sensitized by changing the simulation setup in two manners. First, by including symmetrical opposite defects: if a defect sensitizes an R0DF1, its symmetrically opposed defect will sensitize an R1DF0. Second, by reducing the defect size sweep step: a correct read may become a deceptive read, and then a destructive read as the defect strength increases; if the sweep step is too big, the intermediate behavior is not observed. Finally, it is worth mentioning that all parametric single-cell faults previously identified have been observed.

To better analyze the validation methodology's result and the occurrence of static single-cell faults, we explore the results of some defects more thoroughly, namely OC01, OC03, OC05, OC08, SW02, and BC01. Table 3.8 lists the results. We categorize the faults sensitized by each defect based on defect size ranges. We define these ranges as *fault classes* (FCs). Each FC contains a set of sensitized faults for a given defect size range. Furthermore, an FC may be a subset of another FC, i.e., if FC-1 $\subset$ FC-2, FC-2 contains all the faults in FC-1, plus other faults not in FC-1. We can see a direct relation between defect size and observed faulty behavior. For smaller defects, only parametric faults are observed. However, as the defect's resistance increases, we start to observe faulty functional behavior.

To depict the simulations' output, we illustrate the electrical waveforms when simulating a cell affected by defect OC08. The waveforms are shown in Figs. A and B; they show the operation sequence $w1$, $r1$, $w0$, $r0$, $w1$, $r1$. Fig. 3.7 illustrates the case for OC08 = 25 kΩ. According to Table 3.8, only the fault rR0NF0 should be observed. Indeed, when a $r0$ is performed in the cell, the $\overline{BL}$ is barely discharged, characterizing a random read fault. On the other hand, Fig. 3.8 illustrates the case for OC08 = 50 kΩ. We can see that the cell fails to go to '0' during the $w0$ operation, thus characterizing a W0TF1, as indicated in Table 3.8.

We then increase the number of consecutive operations (i.e., $n+1$) to investigate the occurrence of dynamic faults. Only resistive-open defects led to dynamic faults; Table 3.9 outlines the results for 3 exemplary defects. Defect OC01 sensitizes dynamic parametric faults for defect sizes much smaller than the faults sensitized for $n \leq 1$. This faulty behavior originates from the disturbance generated when writing the defective memory cell. Dynamic faults were also observed for OC03. While smaller defects may require up to 23 consecutive operations to sensitize a functional fault, bigger defects require a

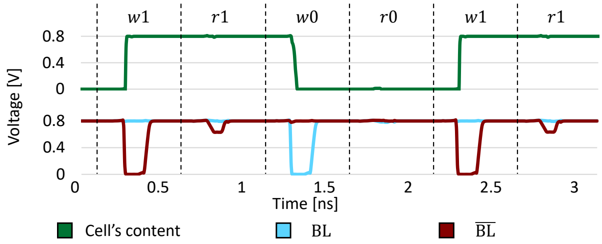


Figure 3.7: Electrical waveforms when simulating a cell affected by OC08 = 25 kΩ.

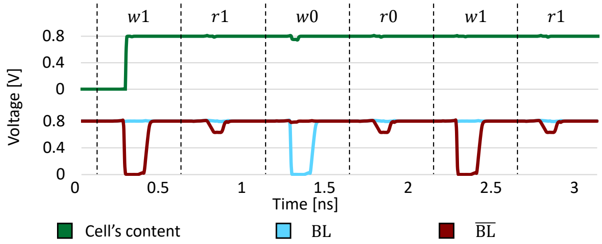


Figure 3.8: Electrical waveforms when simulating a cell affected by OC08 = 50 kΩ.

maximum of 7 consecutive operations to sensitize the same destructive behavior. This dynamic behavior is illustrated in Fig. 3.9. The top waveform shows the content of a cell affected by OC03 = 23 MΩ; it is necessary to apply 23 consecutive read operations to sensitize the destructive behavior. However, for a stronger defect, i.e., bigger resistance, the number of read operations necessary to trigger the destructive behavior is much smaller, as shown by the waveform at the bottom of the figure. The waveform concerns the content of a cell affected by OC03 = 50 MΩ; the destructive behavior is triggered after only seven read operations.

This chapter has discussed and classified faults based only on their sensitization. However, sensitization is only the first step in memory testing; *fault detection* is also a major critical step that determines the test's feasibility and efficiency. In the next chapter, we will explore how these faults discussed previously can be further split based on their detection requirements and what they imply for high-quality test solutions.

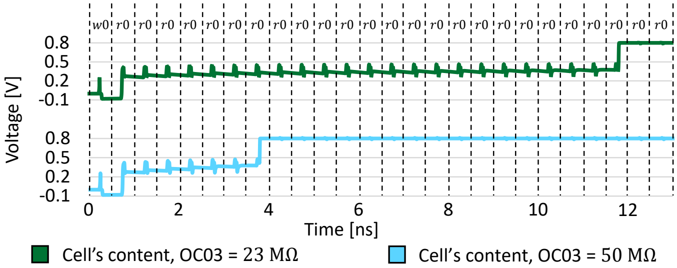


Figure 3.9: Electrical waveforms when simulating a cell affected by OC03 = 23 MΩ and OC03 = 50 MΩ.

Table 3.7: Single-cell static fault modeling results of all injected defects.

FP	Defect																							
	OC01	OC02	OC03	OC04	OC05	OC06	OC07	OC08	OC09	OC10	OC11	OC12	SC01	SC02	SB01	SB02	SW01	SW02	BC01	BC02	BC03	BC04	BC05	BC06
S0F1													X	X						X	X			
S1F0																						X		
W0DF1																X					X			
W0TF1					X	X	X	X	X	X					X	X		X			X			X
W0TF1$_r$		X			X																			
W1TF0						X												X						
iR0NF0																X					X			
iR0DF1														X		X								
rR0DF1													X	X										
S0FU																			X					
S1FU																			X					
W0TFU	X	X			X						X													
W1TFU											X													
rR0NF0			X	X				X	X	X		X	X		X	X		X			X		X	
rR1NF1												X						X					X	X
rR0DFU																			X					
rR1DFU																			X					
dR0DFU																			X					
BLS			X	X				X	X	X		X	X	X	X	X		X	X		X	X	X	X
SNM	X	X	X	X									X	X			X		X	X	X	X		
RNM			X	X									X	X					X	X	X	X		
PCH													X	X	X	X	X		X	X	X	X		X
PCR													X	X		X	X	X		X		X		X
PCTW													X		X	X	X	X					X	X
PCNW													X		X	X	X	X		X	X	X	X	X

Table 3.8: Single-cell static fault modeling results of resistive defects.

Defect Model	Resistance (Ω)	Fault Class	Faults Observed
OC01	(0, 474 K]	-	Fault Free
	(474 K, 3.8 M]	1	SNM
	(3.8 M, +∞)	2	FC-1 + W0TFU
OC03	(0, 2 K]	-	Fault Free
	(2 K, 20.2 K]	1	BLS
	(20.2 K, +∞)	2	FC-1 + rR0NF0
OC05	(0, 3.8 M]	-	Fault Free
	(3.8 M, 6.6 M]	1	W0TF1
	(6.6 M, +∞)	2	W0TFU, W0TF1_r
OC08	(0, 2 K]	-	Fault Free
	(2 K, 20 K]	1	BLS
	(20 K, 29.3 K]	2	FC-1 + rR0NF0
	(29.3 K, +∞)	3	FC-2 + W0TF1
SW02	(0, 200.9]	6	FC-5 + rR0NF0, rR1NF1
	(200.9, 473.6]	5	FC-4 + W0TF1, W1TF0
	(473.6 , 1.1 K]	4	FC-3 + BLS
	(1.1 K, 10.2 K]	3	FC-2 + PCNW
	(10.2 K, 29.3 K]	2	FC-1 + PCTW
	(29.3 K, 38.4 K]	1	PCR
	(38.4 K, +∞)	-	Fault Free
BC01	(0, 11 K]	6	FC-3 + S0FU, S1FU
	(11 K, 20.2 K]	5	FC-4 + rR0DFU, rR1DFU
	(20.2 K, 29.3 K]	4	FC-3 + dR0DFU, dR1DFU
	(29.3 K, 202 K K]	3	FC-2 + RNM
	(202 K, 1.1 M]	2	FC-1 + SNM
	(1.1 M, 22.5 G]	1	PCH
	(22.5 G, +∞)	-	Fault Free

Table 3.9: Single-cell dynamic fault modeling results of resistive opens.

Defect Model	Resistance (Ω)	Fault Class	Faults Observed
OC01	(0, 112.5 K]	-	-
	(112.5 K, 150 K]	FC-1	2-PCH
	(150 K, +∞)	FC-2	FC-1 + 2-SNM, 2-PCH
OC03	(0, 22 M]	-	-
	(22 M, 23 M]	FC-1	23-rR0DF1
	(23 M, 24 M]	FC-2	18-rR0DF1
	(24 M, 25 M]	FC-3	16-rR0DF1
	(25 M, 26 M]	FC-4	14-rR0DF1
	(26 M, 27 M]	FC-5	13-rR0DF1
	(27 M, 28 M]	FC-6	12-rR0DF1
	(28 M, 30 M]	FC-7	11-rR0DF1
	(30 M, 32 M]	FC-8	10-rR0DF1
	(32 M , 36 M]	FC-9	9-rR0DF1
	(36 M , 43 M]	FC-10	8-rR0DF1
	(44 M, +∞)	FC-11	7-rR0DF1

4

Test Solutions for FinFET SRAMs

In Chapter 3, we have validated the memory fault space, which included sensitizing all faults in the space. However, the manufacturing test procedure during mass production must not only sensitize but also detect these faults. This increases text complexity as faults may have different detecting conditions, e.g., not all faults are detected by simply writing and reading the memory. Therefore, high-quality test solutions must include different and appropriate techniques to guarantee the detection of many faults.

This chapter presents the test solutions proposed throughout this thesis project; they focus on detecting faults with complex detecting conditions. First, we introduce a fault classification scheme based on their detection requirements. Second, we present a framework for test development. Then, we discuss some of the existing test solutions proposed to detect faults with complex detecting conditions. Following, we present the proposed test solutions; they are divided into test solutions for functional faults and parametric faults. Finally, we briefly discuss the outlook for test solutions.

Parts of this chapter have been published in MR'18, ETS'19, DATE'20, TVLSI'21, ETS'21, and DTIS'21 [60]–[65]

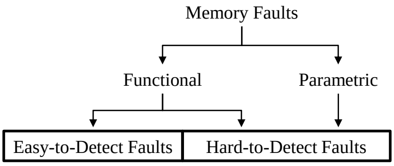


Figure 4.1: Two-level fault classification.

4.1. FAULT CLASSIFICATION BASED ON DETECTION

We have discussed in length the different characteristics faults can show regarding their impact and sensitization requirements. Some faults can impact the memory's functionality, while others cause parametric deviations. They may be associated with a single cell or a specific group of cells. When developing a test solution to sensitize as many faults as possible, it is essential to consider these characteristics. However, sensitization is not enough; test solutions must also detect these sensitized faults. On the one hand, detecting some faults is straightforward: writing and reading the memory is sufficient. On the other hand, other faults may exhibit more complex detecting conditions, e.g., random behaviors. Moreover, they may not be detectable by writing and reading the cell, e.g., parametric faults. Therefore, we propose to classify faults based on their detection requirements also. Fig. 4.1 illustrates this classification. While the first level considers the fault's impact, i.e., functional or parametric, the second level distinguishes ***easy-to-detect*** **(ETD)** faults from ***hard-to-detect*** **(HTD)** faults. The former denotes faults whose detection is *guaranteed* by writing and reading the memory; the latter denotes faults that *may* be detected by these operations – their detection is only guaranteed by using special testing solutions. We explore each of these groups next. Then, we discuss this classification's implications on the fault modeling results presented in Chapter 3.

4.1.1. EASY-TO-DETECT FAULTS

Functional ETD faults *always* impact the memory's functionality. Therefore, test approaches that rely on logic fault observation can easily detect them, i.e., their detection is *guaranteed* by simply writing and reading the memory. For example, $\mathrm{W1TF0} = \langle 0w1/0/-\rangle$ can be sensitized by writing '1' on a cell containing '0' and be detected by immediately reading this cell. Regarding memory array faults, an ETD fault is any fault whose F is '0' or '1', and whose R is either '0', '1', or '–'.

Functional ETD faults have been extensively analyzed in the literature due to their high impact on the memory's functionality. Works in memory testing focused on proposing algorithms, i.e., operations sequences, that can cover many ETD faults with different sensitizing conditions; such algorithms are known as *march tests* [29]. These tests consist of a finite sequence of march elements, i.e., a finite sequence of operations applied to every cell in the memory before proceeding to the next cell. The address order determines how the algorithm is applied through the memory. A traditional address order is linear addressing: *increasing* address order, denoted by ⇑, *decreasing* address order, denoted by ⇓, and *irrelevant* address order, denoted by ⇕. A more detailed explanation regarding the stressing conditions of a march test is given later in this thesis. A complete march test is delimited by the '{ ... }' bracket pair, while a '(...)' bracket pair delimits a march element. Semicolons separate the march elements, and commas separate the operations within a march element.

For example, MATS+ [149], one of the first march tests ever proposed, is defined as $\{\Updownarrow(w0);\Uparrow(r0,w1);\Downarrow(r1,w0)\}$. First, it initializes the memory with '0'; the order in which these write operations are applied to the memory is irrelevant. Then, in an increasing address manner, the operation sequence read '0' and write '1' is applied to each cell. Finally, the operation sequence read '1' and write '0' is applied to each cell, starting from the cell with the most significant address and then decreasing it. With technology miniaturization, many works focused on fault models with more complex sensitizing operations, e.g., coupling faults and dynamic faults [47], [48], [150]–[154]. Therefore, march tests had to be improved, thus increasing their complexity. Many well-known March algorithms that guarantee to detect ETD faults have been developed as results of these studies, such as March SS [155], March C- [156], March AB [157], and more recently, the FinFET-specific March FFDD [57].

4.1.2. Hard-to-Detect Faults

HTD faults are faults whose detection is not guaranteed by performing read and write operations on the cell; they can be divided into **parametric HTD** and **functional HTD** faults. Parametric HTD faults are severe parametric deviations that cause the memory cell to fail one or more of its specifications. They *do not* impact memory cells' functionality. Therefore, their detection is not guaranteed by performing read and write operations. From the logic functionality point of view, these faults are *undetectable*, as all operations pass correctly. Consequently, only special test solutions can guarantee the detection of parametric HTD faults.

Contrarily, functional HTD faults *may* impact the memory's functionality; they are related to random read outputs and undefined cell state. Considering that random effects such as PV impact the outcome of functional HTD faults, it is statistically expected that March tests will detect only part of these faults due to incorrect logic behavior. The remaining faults that do not cause incorrect logic behavior will result in test escapes and compromise the circuit's reliability. Therefore, only special testing solutions can guarantee the detection of functional HTD faults. Although HTD faults may not lead to functional errors at time zero, they may cause reliability issues. Undetected HTD fault may cause random faulty behaviors if the memory is used in harsh environments or conditions, leading to soft errors. Furthermore, HTD faults' impact may worsen once the memory ages, thus leading to a higher failure rate and a shorter lifetime [158]. Thus, detecting HTD faults and avoiding test escapes is of high importance. Next, we discuss each of these functional faulty HTD behaviors, i.e., random read and undefined state.

Random Read Faults

Defects may impact a cell's ability to discharge its BLs during a read operation, thus reducing the voltage difference between the BL pair when enabling the *sense amplifier* (SA). This voltage level is the *BL swing* (BLS), and is defined as BLS = $|BL-\overline{BL}|$; it is used as the SA's input. A ***random read fault*** **(RRF)** occurs when BLS voltage is too small for the SA to guarantee a correct output, resulting in a random output, i.e., either '0' or '1'

[47].

The FP notation describes RRFs as faults in which R is expressed as **?**. For example, $\langle 1r1/1/?\rangle$ denotes a read '1' operation that does not impact the cell's content but returns a random value. When the output matches the cell's content, the RRF is not detected. Therefore, performing a sequence of write and read operations *may* detect this fault; it is, therefore, defined as an HTD fault. Note that not all RRFs are considered HTD faults, e.g., an RRF is ETD if it also flips the cell's content to the inverted logic value, as a subsequent read operation in this same cell detects the deterministic faulty behavior. If not detected, RRFs become test escapes, a known cause for no-trouble-found components reliability issues [159], [160]. Therefore, detection of RRFs is critical to assure high-quality FinFET SRAMs.

It is statistically expected that some RRFs will lead to correct outputs and thus test escapes. The remaining will lead to a failure, i.e., the SA will not output the expected logic value, thereby enabling RRF detection. Two primary metrics determine the read operation outcome: the SA offset and BLS. The SA offset is the voltage shift that mismatches the SA's cross-coupled inverter pairs' strengths. In a PV-free SA, this offset is 0 V. However, PV will cause one pair to be stronger than the other, offsetting the SA towards the logic value '1' or '0'. The SA outputs the correct value only if its offset is smaller than BLS; we define this voltage difference as $\Delta\text{V} = \text{ BLS } - \text{ SA offset voltage}$. Consequently, if $\Delta\text{V} < 0$, the SA outputs the incorrect value. For example, an SA with a 3 mV offset will correctly output the cell's content if BLS is greater than 3 mV. We use the variation of these two parameters that form ΔV, i.e., SA offset and BLS, to express the failure rate.

We first consider SA's offset variation of a traditional 6T SA design, as previously shown in Chapter 2. We performed *Monte Carlo* (MC) analysis (20,000 simulations) on the SA, and obtained a (rounded) mean (μ) offset of 0 V and a standard deviation (σ) of 6 mV. The probability that the SA's offset variation is within or outside a given range can be calculated by the cumulative distribution function [161]; the probability that this variation is outside an $n\sigma$ range, i.e., the SA's offset is greater than $|n\sigma|$, where $n = 1$, 2, 3..., is given by $1 - erf(n\sigma/\sqrt{2})$, where *erf* is the error function

$$erf(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt.$$

We then relate $n\sigma$ to BLS, i.e., SA's input. The probability that the SA offset is greater than σ, 2σ, etc, is by default the probability that the SA's offset is greater than 6 mV, 12 mV, etc. This failure rate function is plotted in Fig. 4.2; it represents the probability of an incorrect output value, i.e., failure, for a given BLS input. Likewise, it also shows the likelihood of no failure, and thus test escapes.

A second failure rate can be estimated based on BLS variation. For example, consider a defect that bridges a column's BL pair; every read operation on this column will be impaired. If the impact is significant, this defect will cause RRFs. Nonetheless, the BLS among this column's cells will vary slightly due to PV; some cells will lead to correct outputs, while others will not. Considering that the SA's offset voltage is fixed after manufacturing, the probability p of incorrect output becomes a function based on BLS variation from one cell to another. In details, p is the probability that BLS variation will counter the voltage difference between mean BLS in the column and the SA's offset, i.e.,

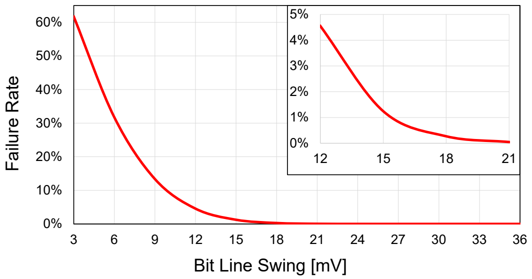


Figure 4.2: Expected failure rate for a given BLS.

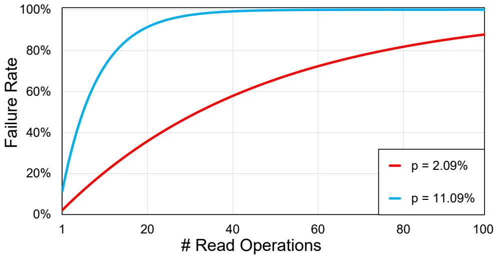


Figure 4.3: Expected failure rate after n read operations for two scenarios.

$p = Pr(\text{BLS variation} < -\Delta\text{V})$. Furthermore, we can use p to estimate the failure rate after a sequence of read operations in the same column, i.e., same BL pair, but different cells. As BLS will slightly vary for each cell, there is a possibility that after n read operations in the column, one of these variations will annul ΔV, leading to an incorrect output.

We illustrate this failure rate with the following example. Consider that a defect bridging both BLs reduces the column's μ BLS from its nominal value to 15 mV. Furthermore, consider that, after manufacturing, the SA's offset voltage due to PV is 5 mV. Accordingly, the ΔVin this column is ΔV= $15-5=10$ mV. Once ΔV is defined, we measure the standard deviation of BLS variation among cells. After MC analysis of 20,000 read operations, a BLS variation of $\sigma = 4.913$ mV was observed. We use this variation's cumulative distribution function to find the probability of BLS variation annulling ΔV. For ΔV= 10 mV, the probability of incorrect output is $p = Pr(\text{BLS variation} < -10 \text{ mV}) = 2.09\%$. Thus, there is a 2.09% chance that BLS will be smaller than the SA's offset, triggering an incorrect output. We can then use $1-(1-p)^n$ [161] to estimate the failure rate after performing n read operations based on the probability p of a single read operation; after 25 operations, the failure rate is around 40%, and 87% after 100 operations, as shown in Fig. 4.3. Additionally, a second failure rate for an exemplary case where ΔV = 6 mV is also plotted. As ΔV is smaller than the first case, the probability that BLS will annul ΔV, i.e., $Pr(\text{BLS} < -6 \text{ mV})$, is higher than before, namely, 11.09%). Accordingly, the number of read operations, and therefore effort, required to guarantee an incorrect output, i.e., 100% failure rate, and detect the RRF is much smaller.

Finally, the same model can also estimate the failure after n read operations on the same cell. In this case, variations in the BLS must originate from dynamic effects that will change the cell's BLS dynamically from one read operation to another, such as white, flicker, and temperature noise. Based on these analyses, we conclude the following regarding RRFs:

- The probability of an incorrect output increases as more read operations are performed, which justifies the industry's hammering techniques to test memories.
- A smaller BLS leads to higher failure rates. Hence, an efficient way to improve RRF detection is by reducing BLS, which can be achieved by applying specific stresses or using dedicated *Design-for-Testability* (DFT) circuits.
- Reducing BLS reduces test effort and time, as shown by the blue curve in Fig. 4.3. A smaller BLS results in a smaller ΔV and a bigger p, leading to fewer read operations necessary to achieve a higher failure rate.

UNDEFINED STATE FAULTS

Manufacturing defects may impact the cell's storage nodes, leading to voltage deviations on Q and $\overline{Q}$. An ***undefined state fault*** **(USF)** occurs when these voltages deviations severely impact the voltage difference between the storage nodes $\Delta V = |Q - \overline{Q}|$, which should be V_{DD} [47]. This undefined state may impact other memory parameters, such as the SNM or BLS.

The FP notation describes USFs as faults in which F is expressed as **U**. For example, $\langle 1w0/U/-\rangle$ denotes a write '0' operation in a cell currently storing '1'. However, instead of undergoing a transition, the cell's state becomes undefined, i.e., Q and $\overline{Q}$ are deviated from GND and V_{DD} respectively. The detection of this specific USF is not guaranteed by writing and reading the cell, as read operations do not output the analog value the cell is storing; it is only detected if a subsequent read operation returns an unexpected logic value. Therefore, it is considered an HTD fault. Note that not all USFs are HTD faults, e.g., the fault $\langle 0r0/U/1\rangle$ is ETD as the read operation returns a '1' instead of '0', even though the cell's content is destroyed after the read operation.

Likewise to RRFs, USFs also become test escapes if undetected. Furthermore, they may impact the memory's specs, such as SNM and BLS. A cell's *static noise margin* (SNM) is the exterior interference, e.g., temperature noise, α particles, cross-talk, it can stand before flipping its content. It is determined based on the cell's technology and cell configuration, i.e., sizing ratios between pull-ups, pull-downs, and access transistors. More specifically, a cell's SNM can be defined as [162]

$$\mathrm{SNM} = V_{TH} - \left(\frac{1}{k+1}\right)\left(\frac{V_{DD} - \frac{2r+1}{r+1}V_{TH}}{1 + \frac{r}{k(r+1)}} - \frac{V_{DD} - 2V_{TH}}{1 + k\frac{r}{q} + \sqrt{\frac{r}{q}\left(1 + 2k + \frac{r}{q}k^2\right)}}\right) \tag{4.1}$$

where r is the ratio between pull-down and access transistors, q is the ratio between pull-down and pull-up transistors, V_{th} is the threshold voltage, and

$$k = \left(\frac{r}{r+1}\right)\left(\sqrt{\frac{r+1}{r+1 - V_s^2/V_r^2}} - 1\right),$$

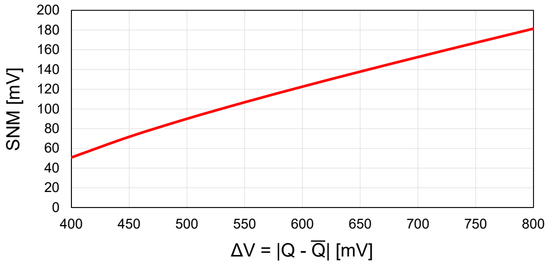


Figure 4.4: Impact of ΔV on the cell's SNM.

$$V_s = V_{DD} - V_{TH},$$

$$V_r = V_s - \left(\frac{r}{r+1}\right).$$

USF's impact on the cell's SNM can be estimated by replacing V_{DD} in Eq. 4.1 for ΔV, as shown in Fig. 4.4. For this analysis, it was assumed a cell ratio of 1:2:2 for pull-up, pass transistors, and pull-down, respectively, $V_{DD} = 0.8$ V, and $V_{th} = 0.11$ V [163]. It is clear that the smaller the ΔV, the smaller the cell's SNM. Accordingly, a cell with smaller SNM is more likely to suffer upsets from exterior noises such as noise and radiation, i.e., they are memory cells with reduced reliability. Thus, it is essential to identify and flag these compromised cells, as they could severely impact high-demanding applications such as automotive and aerospace.

USFs may also impact a cell's BLS. During a read operation, the BL discharge rate depends on many factors, such as the BL capacitance, the access and pull-down transistors' sizing, and the voltage on the storing nodes. Hence, deviations on Q and $\overline{Q}$ may affect a cell's ability to discharge its BLs, thus impacting the final BLS. In more severe cases, it could lead to RRFs as well. It is statistically expected that some RRFs will lead to a failure, i.e., the SA will not output the expected logic value, thereby enabling RRF detection and consequently USF detection. The remaining will lead to correct outputs and therefore test escapes.

We can investigate USF's impact on BLS by estimating how voltage deviations on the discharge node disrupt the cell's discharge capabilities; this is achieved by calculating the voltage on the BL capacitor during a read operation. The simplified version of the cell's discharging circuit is shown in 4.5. A voltage source is directly connected to the storing node Q to represent the USF's effect on the storage node; furthermore, we assume that BL is pre-charged to V_{DD} and $\overline{Q}$ is not affected (i.e., $\overline{Q} = V_{DD}$). The voltage on BL is calculated by estimating the charge in the capacitor over time [164]:

$$V_{BL}(t) = \frac{C_{BL}V_Q - Ke^{-t/C_{BL}R}}{C_{BL}},$$

where R is the resistance of N3, and

$$K = C_{BL}V_Q - Q_0,$$

$$Q_0 = V_{BL(t=0)}C_{BL}.$$

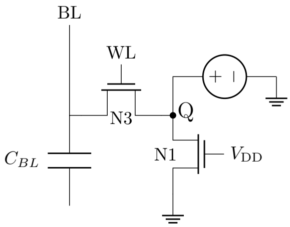


Figure 4.5: BL discharging circuit.

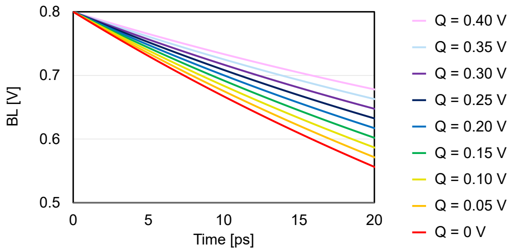


Figure 4.6: BL Discharge based on the voltage on Q.

Fig. 4.6 shows the BL discharge for a period of 20ps considering R = 3KΩ. Clearly, the introduced voltage on the storing node has a significant impact on the final BLS. This analysis assumed that only the discharging node is impacted; voltage deviations on $\overline{Q}$ could also have impacted $\overline{BL}$ i.e., $\overline{BL} < V_{DD}$, which would affect BLS even further.

Based on the analysis of USF's impact on both the SNM and BLS, we conclude the following:

- USFs impact the cell's parameters rather than its functionality. Nonetheless, this parametric impact may lead to functional faults.
- USFs can lead to both test and reliability issues: the first by hindering the cell's ability to develop a BLS, thus leading to random read outputs, and the second by reducing the cell's SNM, resulting in a cell that is more vulnerable to outside noise and more likely to suffer upsets.
- Without dedicated DFT circuits, USFs can only be detected if they (1) lead to an incorrect output generated due to an RRF or (2) suffer an upset caused by noise.

4.1.3. FAULT MODELING RESULTS REGARDING DETECTABILITY

When we discussed fault modeling in Chapter 3, we focused on fault sensitization. Now that a new classification has been presented, we can investigate how it relates to the previously-presented results. We can split the faults in the static single-cell fault space based on their detection requirements, as shown in Table 4.1. The space is roughly split in half; there are 16 ETD faults and 19 HTDs, of which 12 functional HTD and 7 parametric HTD.

Table 4.1: Static single-cell fault space divided into ETD and HTD faults.

ETD Faults	HTD Faults	
Functional ETD	Functional HTD	Parametric HTD
S0F1, S0F1	S0FU, S1FU	BLS
W0TF1, W1TF0	W0TFU, W1TFU	SNM, RNM
W0DF1, W1DF0	W0DFU, W1DFU	PCH, PCR, PCTW, PCNW
iR0NF0, iR1NF1	rR0NF0, rR1NF1	
dR0DF1, dR1DF0	dR0DFU, dR1DFU	
rR0DF1, rR1DF0	rR0DFU, rR1DFU	
iR0DF1, iR0DFU, iR1DF0, iR1DFU		

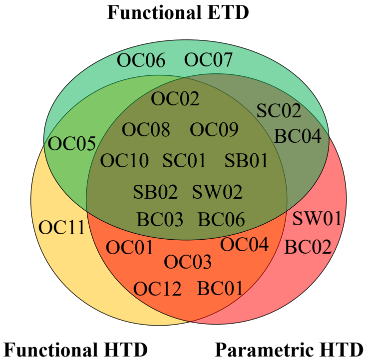


Figure 4.7: Distribution of resistive defects based on sensitized faults.

We can also relate the faults triggered by each defect during circuit simulation, as shown in Fig. 4.7. The green circle consists of defects that led to ETD faults; these defects can be detected by traditional march tests if they are sensitizing any of their ETD faults. However, March tests do not guarantee the detection of defects on the functional HTD or parametric HTD space (yellow and red circle, respectively) or their intersection. Furthermore, it is also possible that a weaker version of a defect that can cause a functional ETD fault is only sensitizing an HTD fault, therefore limiting the efficacy of March tests.

Based on the distribution shown in Fig. 4.7, as well as all the results presented previously in Section 3.5.3 of Chapter 3, we can conclude the following:

- Covering only functional ETD faults is not enough, as most defects will lead to some form of HTD faulty behavior. From the 24 resistive defects investigated, 22 led to HTD faults. Furthermore, 8 out of 24 defects led to HTD faults only, parametric or functional. Detecting these defects is not guaranteed without test solutions also targeting HTD faults. Thus, special test solutions are necessary to guarantee the detection of HTD faults.

- Some defects will lead to both functional and parametric HTD faults. In those

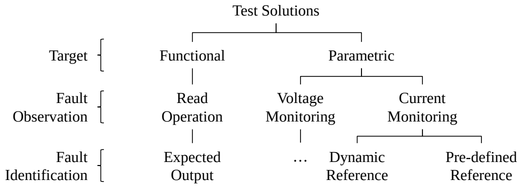


Figure 4.8: A 3-step framework to develop memory test solutions.

cases, test engineers have the flexibility to choose between test solutions that focus on functional faults or parametric faults.

- Few defects trigger only one type of fault; most defects will sensitize a combination of functional ETD, functional HTD, and parametric HTD faults. In general terms, weaker defects lead to parametric HTD faults, while stronger defects lead to functional HTD and ETD faults.
- The occurrence of static HTD faults does not necessarily mean the occurrence of dynamic ETD faults. Therefore, increasing the number of operations applied to a cell does not necessarily sensitize a fault that is easier to detect.

4.2. FRAMEWORK FOR TEST DEVELOPMENT

The first step when developing a test solution for a given memory is defining a fault space of all possible faults that could occur in this given memory. The second step consists of validating this faults space by verifying which faults can be sensitized by manufacturing defects; only these faults should be targeted when testing the given memory. The third step involves developing a method to (hopefully) detect the targeted faults. We propose a framework to develop memory test solutions. This framework consists of three sequential steps, as shown in Fig. 4.8:

1. The test target, i.e., the type of faults it targets;
2. The fault observation method, i.e., how it observes the faulty behavior;
3. The fault identification method, i.e., how the behavior is classified as faulty.

The first step is to define what will be tested: functionality or memory parameters. A functional test solution is a procedure that focuses on the logic functionality of the memory, i.e., if read and write operations are performed as expected. Differently, a parametric test solution focuses on identifying parametric deviations. More specifically, it aims at verifying whether parameters are within the tolerated performance range.

Once the testing target is defined, the next step in the framework is to define how to observe (faulty) behaviors. Since functional test solutions are testing the memory's functionality, the only way to observe faults is by performing read operations; this is the highest level of observability such test solutions may achieve. In contrast, parametric test solutions focus on the memory's parameters. Thus, they must monitor these parameters to observe such faults; which memory parameters will be monitored, e.g., voltage, current, depends on the targeted faults.

The third and final step consists of fault identification, i.e., confirming that the observed behavior is indeed a fault. Functional test solutions can only identify faults by verifying the memory's output, i.e., comparing the obtained read output of a given read operation with the expected output for this operation. Alternatively, parametric test solutions must compare the measured electrical parameter with a reference value. This reference could be dynamic, e.g., measurements from other components in the same circuit, or predefined, e.g., a threshold defined during the design phase.

Additionally, every test solution uses *stressing conditions* (SCs) to trigger or boost the sensitization of the targeted faults. SCs are parallel to the characteristics specified in the framework; any test solution may use a single SC or even a combination of different SCs, regardless of the faults it targets and how it observes and identifies faults. They may use a single SC or even a combination of different SCs. We split SCs between two types: algorithm-related stress and environment-related stress [143]. Algorithm-related stress specifies the algorithm that will be applied to the memory cells. It includes all types of SCs derived from using only write and read operations. Examples of algorithm-related SCs are as follows:

- **Base Test (BT)**: A BT is a sequence of operations (reads and writes) applied to a memory cell.
- **Address Order (AO)**: The AO indicates in which sequence the algorithm accesses addresses. It can be increasing ($\Uparrow$), decreasing ($\Downarrow$), or irrelevant ($\Updownarrow$).
- **Address Direction (AD)**: The AD indicates how the one-dimensional AO is applied in the two-dimensional cell array. It can be linear, or generated by addressing methods such as fast x [143] and H2/H3/HN1 [165].
- **Data Background (DB)**: DB is the pattern of ones and zeros, as seen in the memory array. The most known DBs are Solid (00...00/00..00), checkerboard (01...01/10...10), row Stripe (11...11/00..00), and column Stripe (11...11/00...00) [143].

In contrast, environment-related SCs use additional stress sources to create unrealistic SCs under nominal operating conditions. These include [143], but are not limited to:

- **Voltage Stress**: An additional source applies voltage stress to the circuit. It can apply additional stress in the entire circuit (e.g., changing the supply voltage) or specific components (e.g., *write driver* (WD), SA).
- **Timing Stress**: An additional source applies timing stress to the circuit. It can apply the additional stress in the entire circuit (e.g., changing the circuit's frequency) or specific components (e.g.,WD, SA) to change memory operations' timing.
- **Temperature Stress**: The temperature is either increased or reduced from its nominal value during testing.

4.3. Existing Solutions and Their Limitations

In this section, we discuss previously-proposed memory testing solutions; while we focus on FinFET SRAMs, we also mention strategies proposed for planar CMOS SRAMs and emerging memories. We classify them based on the characteristics defined by the proposed test development framework. Furthermore, we also identify their SCs, HTD fault coverage, and limitations. We do not discuss fault coverage for functional ETD faults since any test solution using appropriate algorithm-related stress can guarantee the detection of these faults.

We first discuss functional test solutions, which use read operation as the fault observation method, and compare the obtained output to an expected output value to identify faults. **Ad-hoc tests** are one of the oldest forms of structured memory test solutions [25]. Examples of ad-hoc tests include GalPat [25], SCAN [149], and Walking 1/0 [26]. **March tests** are the evolution of ad-hoc tests; they are based on FPs and have a linear time complexity. Examples of march tests include March SS [155], March C- [156], March AB [157]. To the best of our knowledge, the only march algorithm proposed specifically for FinFET SRAMs is March FFDD [57], which proposes to detect FinFET-specific dynamic faults.

Both ad-hoc and march tests use algorithm- and environment-related SCs [166]; They can only partially cover random RRFs due to the random nature of this kind of fault; they do not detect any of the other HTD faults. In contrast, they do not lead to yield loss, as they will not indicate fault-free cells as faulty. Furthermore, they do not require any hardware modification as they do not introduce additional hardware in the memory; consequently, they do not require post-silicon calibration.

The fault coverage of march tests can be improved by modifying the operations' timing through **self-timed circuits** [167]. This can be achieved by changing the WL signal's timing, effectively increasing or decreasing the time the cell can discharge BLS during a read operation, or that the WD can write the new value into the cell. Voltage stress is also applied by changing the circuit's V_{DD}. This test technique directly improves the coverage of functional HTD faults. Nevertheless, it cannot fully cover these faults as it only facilitates incorrect behavior instead of forcing it. It also leads to yield loss, as this solution creates unrealistic operating conditions that cause fault-free cells to fail; built-in post-silicon calibration can alleviate this drawback. This test solution has been validated through simulations of an industrial 28 nm planar CMOS memory.

Finally, additional hardware can be used to boost fault coverage even further. These circuits, i.e., DFTs, usually consist of hardware specially designed to improve the detection of a specific fault. A well-known DFT technique is to use an additional WD that is weaker than the standard WD, thus enabling a **weak-write test mode**. This technique has been proposed for older planar CMOS SRAMs [168] and emerging resistive memories [169]. In the SRAM technique, the voltage stress applied by the WD during write operations is significantly reduced, causing the failure of transition write operations in fault-free cells. However, cells in an undefined state are still flipped by the weaker WD. This technique does not lead to yield loss since only cells in an undefined state are flipped and identified. No post-silicon calibration strategies have been proposed for this test solution. The technique introduces additional hardware on the memory and consequently requires some hardware modification. Nevertheless, the modification is only marginal, as it only requires some extra gates and transistors on the peripherals.

While functional test solutions can be a bit rigid in that they all aim to observe incorrect read outputs, parametric solutions can be more versatile. For example, one might **monitor the cell's BLS**; such an approach has been used to test FinFET SRAMs [170]. The technique monitors the voltage on BLs during read operations and compares the obtained value to a predefined reference, i.e., a threshold. This test solution fully covers all HTD faults related to BLs, as any deviation that surpasses the predefined threshold is considered faulty. Furthermore, it does not require significant changes in the memory hardware, only a couple of logic gates for monitoring and detecting possible deviations. Nevertheless, the solution does lead to yield loss as it does not use post-silicon calibration strategies to adjust the detection threshold.

Finally, a parametric test solution that has been used for many years is ***quiescent current*** **(I_{DDQ})** [171], [172]. This technique uses measuring mechanisms to monitor the die's quiescent current and compare the obtained value to a predefined reference. These mechanisms can be internal, e.g., built-in sensors, or external, e.g., probing equipment. This test solution fully detects power consumption (PC) parametric HTD faults; it does not detect functional HTD faults as the memory is idle, i.e., no operations are performed. It does not lead to yield loss as it does not flag circuits in the expected performance range as faulty. Furthermore, the required hardware modification depends on the I_{DDQ} methodology used. If the test solution uses built-in sensors, the necessary hardware modification is marginal. However, if measurements are performed using external equipment only, then no modifications are required.

Based on this, one can conclude that there is no single optimal test solution covering all HTD faults; each test solution has a specific target. Ad-hoc tests, march tests, and self-timed circuits focus on functional ETD faults; they only partially detect functional HTD faults. BLS monitoring focuses on random read functional HTD faults and BLS parametric HTD faults. I_{DDQ} focuses on PC parametric HTD faults, but may also detect functional HTD faults. No test solution focuses on SNM/RNM parametric HTD faults. Based on the present limitations, it is safe to assume that a test solution that can efficiently detect HTD faults, i.e., with a high fault coverage and as few limitations as possible, is yet to be developed.

4.4. TEST SOLUTIONS FOR FUNCTIONAL HTD FAULTS

Functional HTD faults are faults that *may* cause incorrect functional behavior. Therefore, the only way to detect these faults is through an unexpected read output. Moreover, the only way to boost these faults' sensitization and detection is by forcing the incorrect, faulty behavior. In this section, we present two analyses carried out throughout this Ph.D. project and three test solutions developed due to these analyses. The analyses involve applying different algorithm- and environment-related SCs to boost RRF and USF detection; no additional hardware was used. We perform multiple experiments to investigate the SCs' efficiency in detecting HTD faults and discuss how they can be combined to maximize fault detection. The three test solutions comprise additional DFT circuits; two for RRFs, and one for USFs. We introduce their concept, describe their implementation, and present the validation experiments.

4.4.1. SCs TO IMPROVE RRF DETECTION

As discussed in Section 4.1.2, the detection of RRFs is directly related to BLS and SA's offset voltage. While altering the SA's offset requires additional hardware, e.g., DFT circuits, especial SCs can be applied to stress BLS and further improve detection rate. The different algorithm-related SCs that can be used were previously mentioned in Section 4.2. In summary, algorithm-related SCs consist of a base test (BT), address order (AO), address direction (AD), and data background (DB), while environment-related SCs consist of voltage stress, timing stress, and temperature stress. Once the most effective SCs are identified, they can be combined into a single functional test solution, i.e., a March test, that boosts HD fault coverage to the maximum possible without the additional stress from special testing hardware.

EXPERIMENTAL SETUP

We perform different experiments using various SCs to identify the most effective ones in detecting RRFs. The simulation setup is similar to the one presented in Section 3.5.1. A total of 14 defects that sensitize RRFs were injected: OC03, OC04, OC08, OC09, OC10, OC12, SC01, SB01, SB02, SW02, BC01, BC03, BC05, and BC06 (see Fig. 3.6). However, we focus on OC03, OC10, and BC05 as they have defect size ranges in which only RRFs are sensitized, i.e., a severely reduced BLS and no impact on the cell's content; thus, the detection of these defects is only achieved through RRF detection. Each defect was swept with increasing sizes (i.e., resistances) to identify the size range in which RRFs are triggered. Each simulation scenario (i.e., using stress conditions X and injecting defect Y of size Z) was simulated 100 times using MC simulations; from these, a mean BLS and a detection rate are calculated. For example, the scenario of using fast-row and checkerboard SC and injecting OC03 of size 20 kΩ was simulated 100 times applying PV effects. These effects are modeled using Pelgrom's model [173] and simulated using a voltage source on the transistor's gate contact of transistors. Measure commands are used to estimate BLS and assure the cell's content has not been destroyed.

DETECTION RESULTS

Fig. 4.9 depicts the detection rates of the three analyzed defects, i.e., OC03, OC10, and BC05. BC05's detection rates after 2 and 10 read operations (ops.) are also plotted; it is expected that its detection rate increases as more read operations are performed in different cells from the same column. Furthermore, the figure also shows the expected failure rate presented in Section 4.1.2. We can see that solid DB does not detect RRFs, i.e., the read output always matched what was expected; this is due to the output latch and its own DB. Because BLS is too small to influence the SA, the output latch influences the SA to remain in its current logic state. As the expected read output never changes, RRFs will not lead to incorrect read outputs, becoming test escapes. Therefore, to improve RRF detection, test algorithms must include checkerboard DB stress.

As expected, the detection rate of BC05 improved after consecutive read operations on the defective column. Furthermore, we can notice that the detection rate when using checkerboard DB matches the expected failure rate behavior, but with a shifted starting point depending on the defect. Thus, additional parameters, variations, and conditions must still be included in the expected failure rate to accurately estimate incorrect outputs due to RRF.

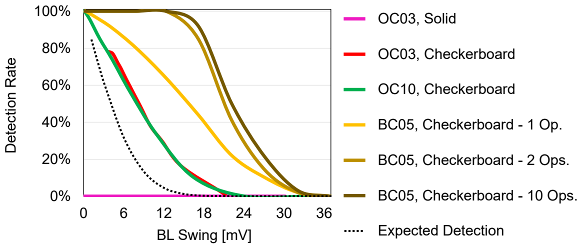


Figure 4.9: Detection rate of defects OC03, OC10, and BC05.

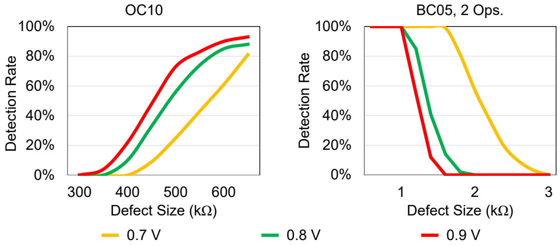


Figure 4.10: Detection rate of OC10 and BC05 for different supply voltages.

We then move to analyze environment-related SCs. Reducing frequency can boost RRF detection; a higher frequency leads to a smaller WL enable period, reducing the BL swing and increasing incorrect outputs. However, our memory uses self-timing circuits to limit WL enable period. Therefore, in our experiments, altering frequency did not affect RRF detection. Nevertheless, temperature and voltage improved RRF detection. While both scenarios had similar results, the detection gain was more significant for supply voltage than temperature; we focus on the former's results.

We have observed that supply voltage does not alter the relation between BLS and detection rate, i.e., a given BLS led to a similar detection rate. However, altering supply voltage changes the BLS for a given defect size. Thus, there were significant changes in the defect size to detection rate relation. Fig. 4.10 shows OC10's and BC05's (2 ops.) detection rates for varied defect sizes and supply voltages; defect OC03 is omitted as it experiences the same impact as BC05. For OC10, increasing the supply voltage reduces BLS. Since reducing BLS is one of the best approaches to improve RRF detection, increasing the supply voltage boosts detection of RRFs caused by OC10. On the other hand, a reduced supply voltage leads to a smaller BLS when considering defect BC05. Hence, the supply's voltage impact on the detection ultimately depends on whether changing V_{DD} will reduce or increase BLS.

DISCUSSION

Based on the experiments using different types of SCs, we can conclude the following:

- Different SCs will impact BLS and the SA's amplification differently. For example, the detection rate when using solid DB is significantly different from that when using checkerboard DB. Therefore, SCs must also be considered when estimating the RRF failure rate. Ignoring such operational aspects will lead to an inaccurate failure rate.

- We did not detect any RRF when using solid DB, only when using checkerboard DB. Therefore, when developing a test solution to detect RRFs, a checkerboard DB is a must.
- As expected, performing more read operations boosts RRF detection. Moreover, we have observed that the detection rate for two consecutive operations is closer to the detection rate for ten operations than that of only one operation. Therefore, the test engineer should consider how much effort, i.e., the number of operations, is sufficient to obtain the desired fault coverage.
- A defect's detection rate is directly connected to the operating conditions, e.g., voltage supply. However, each defect is impacted differently. Therefore, testing in different operating conditions is a must to obtain the highest RRF detection rate possible.

4.4.2. DFT FOR RRF BASED ON SA SENSING

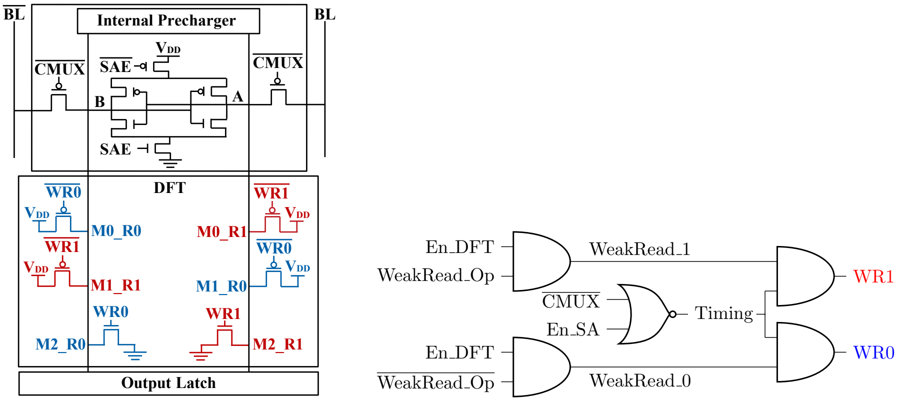
RRFs occur when the SA sensing, i.e., the SA discharge, is too slow due to a significantly reduced BLS. Thus, manipulating the SA discharge and the BLS generation could force an incorrect behavior due to RRF. We exploit this attribute to develop a functional test solution that uses DFT circuits to apply additional voltage stress into the SA during its sensing phase. We name these read operations in which the DFT is enabled to mismatch the SA as *weak read* operations.

CONCEPT

The DFT technique proposed in this work targets RRF faults by mismatching the SA during its sensing phase. Additional transistors create a mismatch in the SA and unbalance it. Fault-free read operations will easily overcome the mismatch during the sensing phase and successfully generate an adequate BLS. However, when an RRF occurs, the generated BLS will not be enough to overcome the DFT mismatch, leading to incorrect output values. Therefore, the proposed DFT can enable RRF detection using simple test algorithms and fault observation.

IMPLEMENTATION

The DFT is divided into the additional transistors in each SA and the DFT control unit, as shown in Fig. 4.11. A total of 6 transistors, divided into two symmetric groups, are connected to the SA's storing nodes. During a weak read '0' operation, transistors labeled with "_R0" are turned on, while "_R1" transistors are kept off; the opposite applies for weak read '1' operations. M0s and M1s are 6-fin and 3-fin PMOS, respectively, while M2s are 1-fin NMOS; they are sized to trigger incorrect read outputs only when the BLS is significantly reduced. During weak read operations, transistor M0 and M2 are simultaneously activated to discharge one node, while the counterpart transistor M1 is activated to prevent the discharge of the opposite node.



(a) The DFT coupled to the SA.

(b) DFT Control.

Figure 4.11: DFT Organization.

A control unit generates the signals to enable the DFT. Two input signals configure the DFT. The *En_DFT* indicates the DFT should be enabled as a weak read will be applied. This signal does not have any timing characteristics; it is enabled with the memory's clock signal. The *WeakRead_Op* signal indicates the type of read operation, i.e., weak read '0' or weak read '1'. The DFT's timing is dictated by the SA's pass transistor's gate signal, i.e., *CMUX*, and the enable SA signal, *En_SA*, which will ultimately lead to the *SAE* signal. Accordingly, the DFT is only active when the SA connects to the BLs and develops a BLS; hence, both DFT and SA have the same timing scheme. Finally, the timing signal is combined with the weak read signals by an OR gate.

DETECTION RESULTS

To validate the DFT, we inject the previously analyzed defects into the memory and estimate the percentage of incorrect outputs when the DFT is enabled. Fig. 4.12 shows the detection results when injecting defect OC03, i.e., an open defect between a storing node and the pull-down transistors, and OC11, i.e., an open defect affecting the gate of a pass-transistor. The figure also includes the expected failure rate described in Section 4.1.2, and the detection curve when using only algorithms. For algorithm-related SCs, it was applied fast row alongside checkerboard DB. All exterior environment-related SCs are kept nominal, i.e., frequency of 2 GHz, V_{DD} of 0.8 V, and operating temperature of 27 °C; note that the voltage applied by the DFT into the SA is considered voltage stress. We can see that while the DFT does improve RRF, its behavior is not yet perfect. The DFT may detect weak defects that would not lead to RRFs; this is shown by the detection rate when the BLS is somewhat still significant, e.g., > 30 mV. Therefore, the DFT leads to *some* yield loss. Furthermore, the max detection rate it achieves for OC03 is 85%; hence, it will also lead to *some* test escapes.

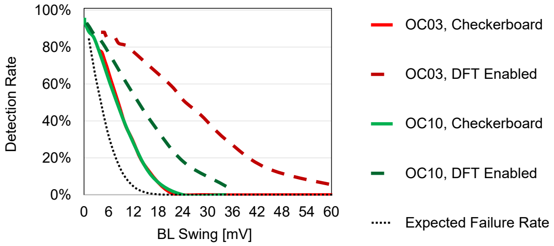


Figure 4.12: Detection rate of defects OC03 and OC10 when using the DFT.

DISCUSSION

- Manipulating the SA's discharge will directly impact incorrect read outputs due to RRFs.
- Even though the DFT presented benefits when aiming at RRFs caused by small defects, a DFT solution with less yield loss and test escapes would be preferred.
- Manipulating the SA's discharge requires a non-negligible area overhead in the SA as it is necessary to minimize the cell's effect in the SA, i.e., BLS development.
- Weak read approaches can be an excellent test solution as they provide the same detection capabilities as traditional algorithm-based solutions, plus the additional coverage of RRFs. However, appropriate mechanisms to make the read "weak" are a deciding factor for this test solution.

4.4.3. DFT FOR RRF BASED ON SA AMPLIFICATION

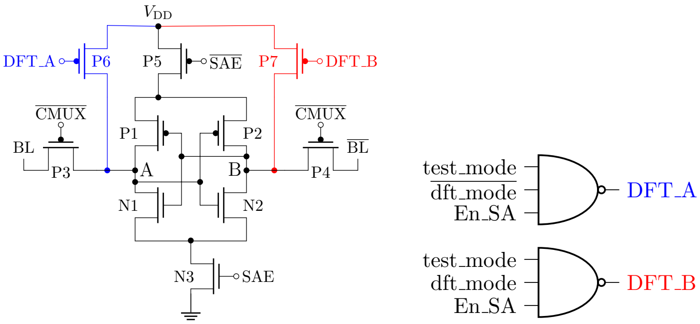
The previous DFT exploited the SA's sensing phase. The next phase following the sensing is amplification, which is also a crucial moment of a read operation. Once the BLs have sufficiently discharged the SA, the SA's pull-up and pull-down connections are enabled, thus forcing the SA to either '1' or '0'. A random read will occur if the BLs have barely discharged the SA; in this situation, the SA's amplification could go either way. Thus, manipulating the SA's amplification can force an incorrect behavior. We exploit this attribute to develop a functional test solution that uses DFT circuits to apply additional voltage stress into the SA during its amplification.

CONCEPT

The proposed DFT technique targets RRFs by creating a mismatch during the SA's amplification. Additional PMOS transistors are inserted into the SA and activated during *weak read* operations to skew the SA's amplification towards the opposite value that is being read. Fault-free read operations will easily overcome the DFT's mismatch, and the SA will output the correct cell content. On the other hand, when RRFs occur, the SA will fail to overcome the DFT's mismatch and output the opposite expected logic value.

IMPLEMENTATION

The DFT is implemented in two parts, as shown in Fig. 4.13. The first part consists of 1-fin PMOS transistors in the SA's pull-up network. Only one transistor is enabled at a time



(a) The DFT coupled to the SA.

(b) DFT Control.

Figure 4.13: DFT Organization.

to create a mismatch towards the opposite logic value expected from the read operation. For example, P6 is activated during a read '0' operation and charges node A since it is expected that BL will discharge this same node. During amplification, the SA becomes less likely to amplify based on the BL swing and more likely based on the DFT configuration, i.e., opposite of the expected value. The RRF is then detected if the SA amplifies to and outputs the opposite expected value. The second part is two 3-input NAND gates that generate the controlling signals based on the enable SA signal (*En_SA*) that generates the *SAE* signal; hence, both DFT and SA have the same timing scheme. Furthermore, two signals configure the DFT. The *test_mode* signal enables the DFT, while *dft_mode* indicates the current read operation, i.e. *dft_mode* = '0' during a read '0' operation and *dft_mode* = '1' during a read '1'.

DETECTION RESULTS

Fig. 4.14 shows the DFT's RRF detection for a given BLS under different supply voltages; algorithm-related SCs have been set to fast-row AD and checkerboard DB. Without the DFT, supply voltage does not change the relationship between the detection rate and BLS; hence, only the detection rate for 0.8 V is shown. However, a considerable increase in this relation is observed when using our DFT, leading to higher RRFs detection for a given BLS; the most significant gain is obtained with a supply voltage of 0.7 V. Nevertheless, we also analyze the DFT's detection based on defect size, as shown in Fig. 4.15; this analysis proves necessary as supply voltage's impact on BLS changes from defect to defect. While we see that BC05's RRF detection rate is indeed the highest when reducing supply voltage, the same is not valid for OC10. Although there is a significant detection gain when using the DFT at 0.7 V supply voltage, the highest coverage is still achieved at 0.9 V. Therefore, using the DFT with reduced and increased supply voltage is the best approach.

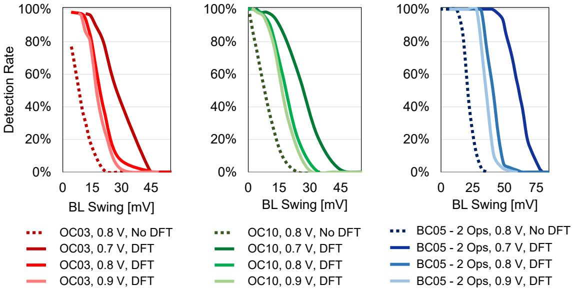


Figure 4.14: DFT Detection rate considering BL swing.

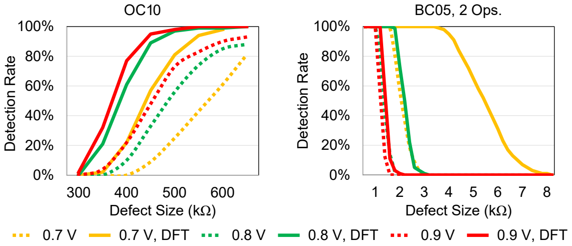


Figure 4.15: DFT Detection rate considering defect size.

POST-SILICON CALIBRATION

The DFT's detection rate is calculated based on the effects of PV. More specifically, two scenarios could occur: (1) the DFT may cause incorrect behavior in fault-free read operations, i.e., *yield loss*; or (2) the DFT does not trigger incorrect behavior during HTD faults, i.e., *test escapes*. Additional calibration techniques can counter PV effects and fine-tune the detection rate, e.g., post-silicon calibration.

We propose a simple yet effective calibration scheme that adjusts the DFT's timing by manipulating the DFT control unit's delay. The control unit has an intrinsic delay from its logic gates, resulting in a slight delay between enabling the SA with *En_SA* and enabling the DFT with either *DFT_A* or *DFT_B*. This delay can be minimized by appropriately designing the logic gates. Accordingly, it can also be increased by using weaker gates. Nevertheless, both solutions are not feasible once the chip is manufactured. Therefore, we propose to use dynamic delay paths to either reduce or increase the delay between enabling the SA and the DFT. The proposed circuit is shown in Fig. 4.16; in addition to the original delay path with a single delay unit, which represents the nominal delay, two supplementary delay paths are introduced. The delay paths are controlled by transmission gates, which are activated once the PV corner is identified. Note that this calibration is only necessary the first time the memory is tested. Thus, the mismatch introduced by the DFT during read operations can be calibrated by choosing an appropriate delay path, leading to a different detection rate. Note that the control unit's delay must be reduced so that the delay on the nominal path is similar to if the DFT did not have any post-silicon calibration circuit.

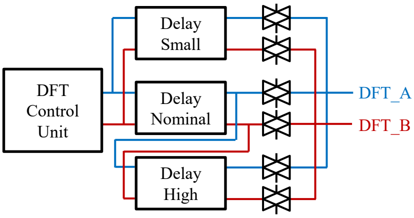


Figure 4.16: A post-silicon calibration technique for the proposed DFT.

DISCUSSION

Based on the DFT's detection capabilities, we conclude the following:

- A DFT that acts on the SA's amplification provides a detection rate more similar to the expected failure rate than the DFT that acts on the SA's sensing.
- It is much easier to design a DFT that acts on the SA's sensing than on the SA's amplification. This is due to the parts involved in each phase. While the foremost one consists only of the SA itself, the latter also includes the cell, the BL pairs, and the SA.
- The DFT introduces a negligible overhead. The control block consists of two 3-input NAND gates only. Moreover, the DFT also introduces two 1-fin PMOS transistors in each SA. Nevertheless, the additional transistors represent an area overhead of only 2.5% of the SA; this overhead is even more negligible considering the cell area covered by a single SA. Furthermore, the DFT does not introduce any test time overhead as it can be enabled throughout all read operations. Hence, combining the algorithm's fault coverage with the DFT's RRF coverage with the same effort and zero time overhead is possible.

4.4.4. SCs TO IMPROVE USF DETECTION

The previous test solutions focused on detecting RRFs; they applied as much stress as possible to obtain an incorrect read output. Nevertheless, RRF is not the only functional HTD behavior. A memory cell may also have its contents destroyed to an undefined value; it is not possible to define what the cell is storing. Appropriate SCs can also be applied to the memory to improve the sensitization of these faults. However, as they are faults related to the cell's content, their detection relies upon two steps: (1) manipulating the cell's content to trigger an undefined state and (2) reading the faulty cell's content. Since the cell is in an undefined state, the read operation may lead to a random output, i.e., an RRF. Therefore, any SCs that improve RRF detection should be used as well.

EXPERIMENTAL SETUP

We perform different experiments using various SCs to identify the most effective ones in sensitizing USFs. The simulation setup is similar to the one presented in Section 3.5.1. A total of four defects that sensitized USFs have been injected: OC01, OC02, OC05, OC10, and BC01 (see Fig. 3.6). Each defect was swept with increasing sizes (i.e., resistances)

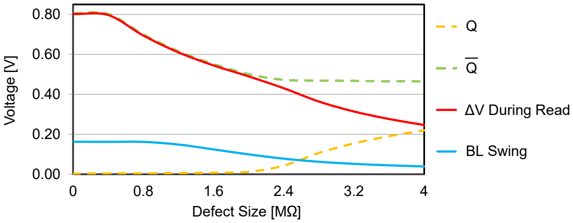


Figure 4.17: Impact on SNM due to defect OC01 after a transition write.

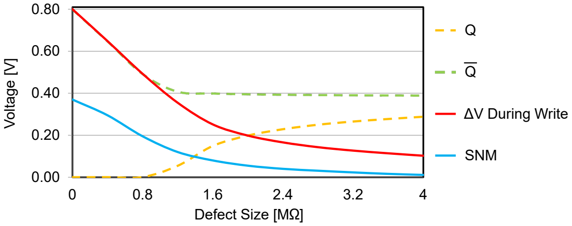


Figure 4.18: Impact on BLs due to defect OC01.

to identify the size range in which RRFs are triggered. Each simulation scenario (i.e., using stress conditions *X* and injecting defect *Y* of size *Z*) was simulated 100 times using MC simulations. The mean voltages on the cell's storing nodes are based on the values measured in each simulation; they are used to estimate electrical parameters, such as BLS and SNM.

SENSITIZATION AND DETECTION RESULTS

We focus on USF sensitization only; the SCs discussed below may impact other types of faults differently. Experiments have shown that USFs can be sensitized by either applying transition write operations or read operations. Furthermore, it was observed that after the USF was sensitized, the SNM and the BLS were significantly reduced. We exemplify this behavior by analyzing the impact of OC01, which sensitizes a USF after a transition write. Nevertheless, this defect will also significantly reduce SNM and BLS. The impact on the cell's SNM is illustrated in Fig. 4.17, which shows the ΔV and the cell's SNM right after the write transition. We can see a clear relationship between the cell's ΔV and SNM. Furthermore, the cell's SNM is significantly reduced from its original value, meaning that this cell is more likely to suffer upsets from exterior noise. Thus, this faulty cell must be flagged during testing.

The only way to detect this fault is by reading the cell. Nevertheless, as the cell is in an undefined state, it fails to properly discharge the cell, as illustrated by Fig. 4.18. The figure shows the cell's ΔV and BLS during a read operation that followed the transition write operation that triggered the USF. Again, we can clearly see the relation between ΔV and BLS. Furthermore, by the point of OC01 = 4MΩ, the read operation did not detect any fault. Thus, more complex SCs must be used to increase the detection of USFs.

Regarding environment-related SCs and their impact on USF sensitization and detection, supply voltage has the most significant impact as it directly changes the maximum ΔV, e.g., a reduced supply voltage leads to a reduced voltage difference between Q and $\overline{Q}$. Frequency and temperature will also impact the cell's SNM and BLS; reducing the frequency shortens the WL enable time, leading to (1) a reduced BLS due to less time to discharge BLs, and (2) write transition faults due to less time to flip the cell's content. Nevertheless, frequency and temperature will not directly impact the cell's ΔV and thus will not impact the sensitization of USFs, only its detection through other means, e.g., RRFs and transition faults.

DISCUSSION

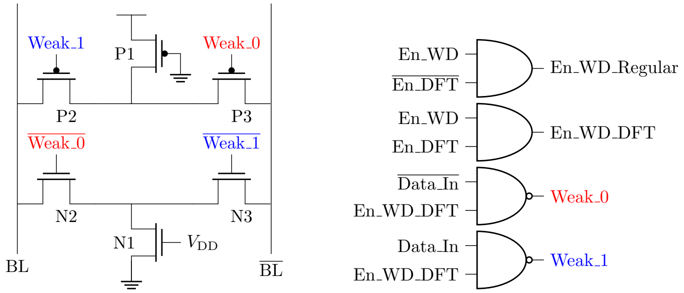
- The most efficient sensitizing BTs are 0w1, 1w0, 0r0, and 1r1. A read operation should follow to (try to) detect the undefined state, even though RRFs may occur.
- AO and AD did not lead to an increase in USF sensitization or detection. While DB did not lead to an increase in USF sensitization, it improves RRF detection and thus should be used when targeting USFs.
- Defects that led to USFs but no RRFs cannot be detected using only algorithm-related SCs. Defects that led to USFs and RRFs can be detected using algorithm-related SCs, but the detection rate will be minimal.
- Reducing supply voltage increases the sensitization of USFs as it diminishes the cell's ΔV hindering both the SNM and BLS. Accordingly, it also increases USF detection through RRFs.
- Changing temperature or frequency does not change the sensitization of USFs, but instead changes the detection of other types of faults, e.g., RRFs and failed transitions, which may lead to the detection of USFs.

4.4.5. DFT FOR USFS

While the sensitization of USFs is straightforward, the same cannot be said regarding their detection. They can only be detected without additional hardware circuits if subsequent RRFs output an incorrect read value. Therefore, some test solutions have been previously proposed to tackle this issue. A well-known solution is the *weak-write* mode [168]. More than 20 years ago, this solution was proposed for a technology node no longer used. Therefore, we propose a refreshed take on this test solution, now designed using the FinFET technology.

CONCEPT

The DFT technique proposed in this work uses the concept of weak-write test mode to target USFs. A second WD is introduced into the memory; it is weaker than the standard WD. This weaker WD is only activated during weak write operations, while the standard WD is enabled during regular operations. It has a unique characteristic in which it *cannot* flip defect-free cells; it is only able to flip cells that are currently in an undefined state. Thus, detection is enabled by performing a write operation to sensitize the USF, followed by a weak write on the same cell to flip the cell to its original logic value, and then read it to check its content. If the read output matches the expected value, it must



(a) Weak Write Driver. (b) DFT Control.

Figure 4.19: DFT Organization.

mean that the cell has undergone the last transition, thus indicating the occurrence of a USF. However, an unexpected read output value indicates the cell did not undergo the last transition operation, which characterizes the behavior of a defect-free cell.

IMPLEMENTATION

The DFT technique proposed in this work uses a similar organization as the circuit in [168] to weakly force a new value into the cell. The DFT's organization and control signals are illustrated in Fig. 4.19. Our proposed DFT differentiates from the one in [168] in the following aspects:

- The proposed DFT does not require additional write time, thus avoiding the overhead of changing the memory's time scheme.
- All pull-up transistors are PMOS devices instead of NMOS devices.
- N1's gate is V_{DD}, rather than its source.

Only one additional signal is necessary to activate the DFT. Furthermore, four additional logic gates are included to generate the appropriate control signals. Finally, the stress applied during the additional read operation is adjusted to improve the detection of RRFs, i.e., use a checkerboard DB stress to manipulate the output latch to influence the SA's amplification phase.

DETECTION RESULTS

Simulations have been performed in the same manner as previous experiments; each scenario (i.e., injecting defect Y of size Z) was simulated 100 times using MC simulations. Measure commands are used to check read outputs and define whether the USF was detected or not. Then, an average detection rate for each scenario is estimated.

Fig. 4.20 shows the DFT's OC01 detection; algorithm-related SCs have been appropriately adjusted to maximize the detection of USFs and RRFs. Without the DFT, detecting USFs is only possible if they trigger RRFs that lead to incorrect read outputs. We can see that detection is minimal, with only 6% at OC01 = 4MΩ and V_{DD} = 0.7V; coming back to Figs. 4.17 and 4.18, both the SNM and BLS are already very much impacted at this

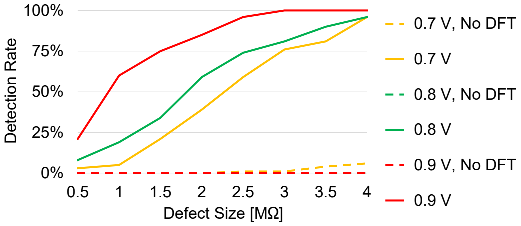


Figure 4.20: DFT Detection rate for detect OC01.

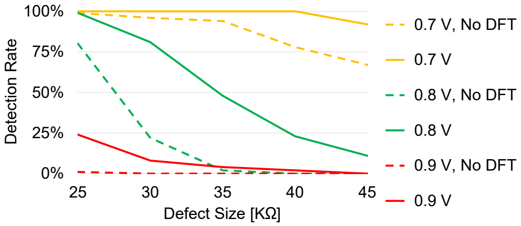


Figure 4.21: DFT Detection rate for defect BC01.

point. The detection rate is significantly increased when using our DFT, with the highest detection obtained with $V_{DD} = 0.9$ V.

Finally, Fig. 4.21 illustrates the DFT's BC01 detection rate. Again, SCs have been set to maximize the detection of USFs and RRFs. While detecting BC01 is easier than OC01, we can see that the DFT still provided significant detection gain by detecting weak defects that lead to BLS and SNM deviations. Overall, the DFT proved efficient in detecting defects that lead to USF and its related faulty effects, such as a reduced SNM and BLS. Furthermore, no yield loss was observed during the experiments, i.e., the DFT did not flag a fault-free cell as faulty, thus confirming that the proposed test solution is appropriate to detect USFs in FinFET SRAMs.

DISCUSSION

Based on the experiments with and without a DFT to detect USFs, we can conclude the following:

- Weak-write mode is still a viable technique to detect USFs in deep-scaled technologies.
- For some defects, it is virtually impossible to detect any USF without a dedicated DFT that specifically targets USFs.
- The area overhead introduced by the proposed DFT scheme is relatively small: only four 2-input gates and the additional write driver (six transistors) introduced into each column. In total, the overhead by column represents the same as one 6T cell.

- The DFT does require an additional sequence of operations to achieve USF detection; however, this additional sequence consists of only four operations maximum, i.e., sensitize USF from ‘1’, read cell, sensitize USF from ‘0’, read cell. Nevertheless, the gains obtained from using the DFT justify the additional test time.

4.4.6. REMARKS ON DFTS’ APPLICABILITY

The previous sections presented three different test solutions that can significantly boost the detection of HTD faults. Despite their benefits, they also have drawbacks and limitations that may jeopardize their applicability. Nevertheless, they all provide valuable benefits:

- In non-critical applications where some defects are allowed, it is cheaper to adapt SCs to increase RRF and USF detection rather than modifying sensitive areas such as the SA or including an additional WD. Notwithstanding, for critical applications such as automotive and aerospace, 0 defective parts per million becomes a must; in these scenarios, the costs of modifying circuits to include DFTs are acceptable. Therefore, employing these DFTs is a trade-off between fault coverage and test cost.

- With the scaling down of supply voltage and increased parasitics, even slight environmental noise will likely lead to random faulty behaviors or bit upsets. Thus, our DFTs are applicable to improve the detection of functional HTD faults in further-scaled FinFET memories, such as 7 and 5 nm. Furthermore, it could be employed in emerging memories if RRFs prove to be a concern for these technologies.

- The DFTs can be used during the memory’s prototyping, characterization, and validation to identify the occurrence of RRFs and USFS. Such knowledge can then be used to improve the memory’s design and yield before its mass production.

- All the proposed DFT’s detection can be calibrated in multiple ways. Run-time calibration can be achieved using different SCs. Post-silicon calibration is possible by changing the DFT timing to adjust the mismatch and hence detection; this can be achieved with the dynamic delay circuit previously presented.

4.5. TEST SOLUTION FOR PARAMETRIC HTD FAULTS

Parametrics faults only impact the performance range of memory specs; they do not cause any functional faults. They *cannot* be detected by write and read operation, i.e., they will not lead to incorrect read outputs. Therefore, they are classified as HTD faults. As previously mentioned, the only way to detect such faults is by using monitoring techniques that directly measure a specific parameter and identify concerning deviations. In this section, we present a parametric test solution to monitor a memory’s parameter and identify parametric discrepancies by comparing measurements. The solution has been developed considering parametric deviations in the BLs. Therefore, it monitors a column’s BLS and identifies deviations during read and write operations. However, the test solution could be used to monitor other parameters, e.g., leakage current, power consumption, and identify other parametric faults besides BLS.

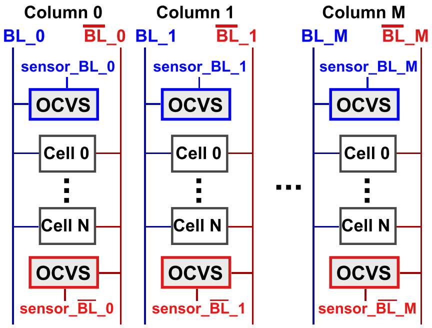


Figure 4.22: DFT organization.

CONCEPT

The BLs connect the cell to the rest of the memory. During a write operation, one of the BLs is discharged by the WD to flip the cell's node connected to the discharged BL. As for read operations, the cell itself discharges the BL connected to its storing node holding '0'; the voltage difference between both BLs during a read operation is what we have been referring to as *BL swing* (BLS). Once a memory operation is performed, both BLs are precharged back to V_{DD}. A manufacturing defect may impact how these BLs are charged or discharged. We have previously discussed how a significantly reduced BLS will lead to RRFs and possibly functional incorrect behavior. However, not-so-severe BLS deviations can still be outside the expected performance range. Furthermore, these deviations will not lead to any incorrect functional behavior. Therefore, a dedicated test solution is necessary to detect these BLS parametric faults.

The proposed DFT performs parametric analysis on the BLs of memory columns. It monitors the bitline swing using *on-chip voltage sensors* (OCVSs) and generates *pulse-width-modulated* (PWM) pulses based on the charging and discharging pace of BLs, i.e., the BLS. Discrepancies can be observed by comparing the PWM signals from neighboring columns; hence, the DFT proposed DFT uses dynamic references. Once a discrepancy in a neighborhood is pinpointed, the cell that generated the faulty BLS can be identified and singled out. Fig. 4.22 shows how the OCVSs are integrated into the memory columns. One sensor is required per BL; thus, each column contains two OCVSs: one for BL and another for $\overline{BL}$. Note that OCVSs monitor voltage levels on both BLs in parallel and independently.

IMPLEMENTATION

Fig. 4.23 depicts the architecture of OCVSs; in this specific figure, the OCVS is monitoring the BL of column 0. Each sensor comprises two blocks: a two-Stage *operational amplifier* (Op-Amp) and a PWM generator. First, the Op-Amp monitors the bitline swing and generates analog output pulses; we use a typical structure of a two-stage operational amplifier [174]. Subsequently, the PWM generator converts the analog pulses to a PWM digital signal that reflects the variations on the bitline swing.

The output of the sensors are collected by a *neighborhood comparison logic* (NCL) circuit [175] that processes them. They generate a flag signal when a fault is detected.

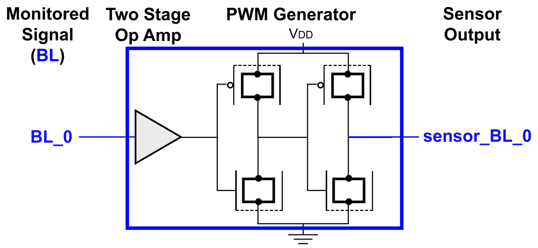


Figure 4.23: On-chip voltage sensor (OCVS).

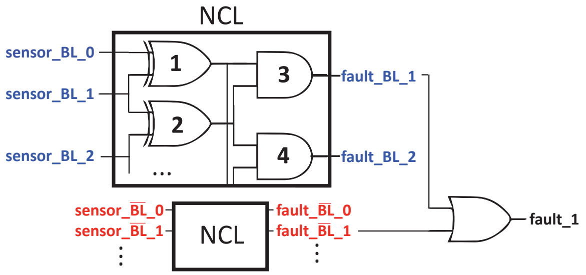


Figure 4.24: Neighborhood comparison logic (NCL).

Two NCL circuits collect sensor outputs from BL and $\overline{BL}$ separately; their results are combined with OR gates. The NCL circuit is shown in Fig. 4.24. It compares the signals from a group of neighboring cells and identifies any discrepancies. Each sensor's output is compared twice to that of other neighbors. For example, the figure shows that BL 1 is compared to BLs 0 and 2. The double comparison ensures the defect cell localization. For example, let us assume a defect is present in a cell connected to bitline 1. The outputs of the XOR gates connected to it, i.e., XOR 1 and 2, will be '1'. Consequently, the AND merging these two signals, i.e., AND 3, will raise a fault flag for BL 1 (fault_BL_1).

VALIDATION EXPERIMENTS

To demonstrate the feasibility of the proposed DFT, OCVSs and NCLs were introduced into the same memory used in previous experiments. A key aspect of the proposed DFT is that all cells in a row must be tested with the same stimuli. To do so, the March algorithm AB $\Updownarrow(w1)$; $\Updownarrow(w0,r0,w1,r1)$ is applied to the memory alongside the DFT. Therefore, the DFT monitors the BLs during all possible operations.

Fig. 4.26 shows the electrical waveforms when injecting defect OC03 = 5 kΩ. This defect does not lead to functional faults, only parametric BLS deviations. The first wave shows the BL of two columns, i.e., x and y, during the operation sequence $w0, r0$; note that the wave is zoomed in the 0.5 V to 0.8 V range to focus on the BLs behavior. The second wave shows each sensors' outputs, which is later forwarded to the NCL. Once the read operation is applied, we can see that the voltage on BL from column y does not discharge as much as column x's BL. OCVSs monitor the BLs and generate a PWM signal based on their behavior. While the signal generated based on column x has a clear

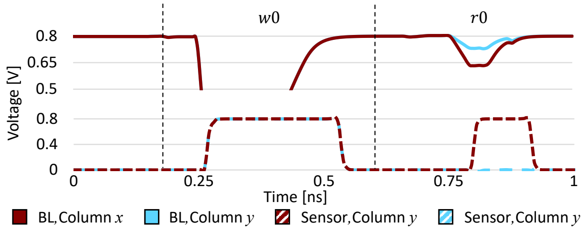


Figure 4.25: Simulation waveforms when injecting OC03 = 5 kΩ into a cell in column *y*.

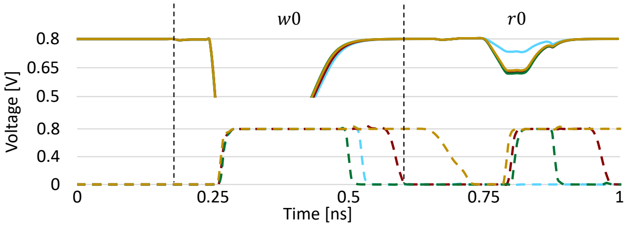


Figure 4.26: Simulation waveforms when considering PV effects; it is not possible to indicate a faulty column based only on sensor's output.

pulse form, the signal based on column *y* is stuck in '0'. Once these signals and the ones from the remaining fault-free columns (which have the same behavior as column *x*) are compared in the NCL, a flag indicating a fault in column *y* is raised.

DISCUSSION

Based on the experiments carried out to validate the proposed DFT, we can conclude the following:

- The proposed approach is an interesting approach to detect parametric deviations in BLs. OCVs can monitor signals such as the BLs, and using NCLs is a viable solution for dynamic references.

- However, this solution is *enormously* impacted by PV effects, leading to yield loss and test escapes. Therefore, as it stands, it is *not* a viable solution. It could become feasible if monitoring methodologies that do not rely on analog processing are used.

- The proposed technique has a high area overhead. First, it requires $2 \times C$ OCVSs, where C is the number of BL pairs. Processing the sensors' outputs requires $C/2$ NCLs with eight logic gates each. Finally, it requires C OR gates to combine the results from BL and $\overline{BL}$.

- One of the biggest markets for FinFET SRAMs is mobile. In these applications, power consumption is a hard constraint; a smartphone's battery should not last only a couple of hours. The proposed methodology could also be adapted to monitor the column's current flow and thus focus on power consumption parametric

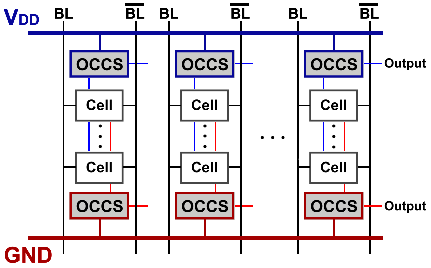


Figure 4.27: Adapter DFT organization if the proposed test solution were to monitor current.

faults. In this case, the DFT's organization must be adjusted to include *on-chip current sensors* (OCCSs), as shown in Fig. 4.27. Note that OCCSs do not have the same sizing as OCVSs and, therefore, must be properly designed to monitor current.

4.6. TEST SOLUTIONS OUTLOOK

The test solutions proposed during this Ph.D. project advance the state of the art by proposing test solutions that focus on sensitizing and detecting HTD faults. Table 4.2 summarizes the existing solutions, following the framework previously presented in Section 4.2. Furthermore, the table also identifies each test solution's stressing conditions, HTD fault coverage, and limitations. Based on the previously discussed and the proposed DFTs, one can conclude that there is no single optimal test solution covering all HTD faults; each test solution has a specific target. Fig. 4.28 illustrates this lack of appropriate test solutions. The figure categorizes the considered test solutions among their algorithm-related stress (BT, BT+AO+AD+AB), their environmental-related stress (No Environmental Stress, Voltage, Time, Temperate), and their faults coverage of Functional HTD (R Read, U State) and Parametric HTD (BLS, Power Consum., SNM/RNM) faults. A green background denotes full detection; yellow background denotes partial detection, while a gray background denotes that no test solutions use that combination of stresses to target a specific fault. Considering the limitations of existing test solutions and the number of possible combinations that are still gray, it is safe to assume that more high-quality test solutions targeting HTD faults (i.e., with a high fault coverage and as minimal limitations as possible) can still be developed. Next, we identify and discuss some of these approaches that memory test engineers can explore to improve HTD fault coverage during manufacturing testing.

- **New and PV-tolerant techniques for parametric test solutions:** parametric test solutions have the potential to provide the highest HTD fault coverage as they delve into the impact of manufacturing defects in these parameters. Hence, these test solutions can detect defects that do not sensitize functional faults. However, parametric tests have many drawbacks. PV immensely affects their detection, leading to yield loss and test escapes. Furthermore, they also have a high area overhead. Therefore, these test solutions may not be feasible. Nevertheless, they

Table 4.2: SRAM Test Solutions

Test Target	Fault Observation	Fault Identification	Solutions	Stressing Conditions							HTD Faults Coverage					Limitations		
				Algorithm				Environment			Func.		Param.					
				BT	AO	AD	DB	Volt.	Time	Temp.	R Read	U State	BLS	PC	SNM RNM	Yield Loss	Post-Silicon Calibration	HW Mod.
Func.	Read Op.	Expected Output	Ad-hoc Tests [166]	✓	✓	✓	✓	✓			P					No	N.A.	None
			March Tests [166]	✓	✓	✓	✓	✓			P					No	N.A.	None
			Self-timed Circuits [167]	✓	✓	✓	✓	✓	✓		P	P				Yes	Yes	Marginal
			Weak-Read [61], [62]	✓				✓			F					No	Yes	Marginal
			Weak-Write [63]	✓				✓			P	F				No	No	Marginal
Param.	Voltage Mon.	Dynamic Ref.	OCVSs [65]	✓							F		F			Yes	No	Severe
		Pre-Defined Ref.	BL Swing Monitoring [170]	✓							F		F			Yes	No	Marginal
	Current Mon.	Dynamic Ref.	OCCSs [64]	✓							P			F		Yes	No	Severe
		Pre-Defined Ref.	I_{DDQ} [171], [172]	✓							P			F		No	N.A.	Marginal

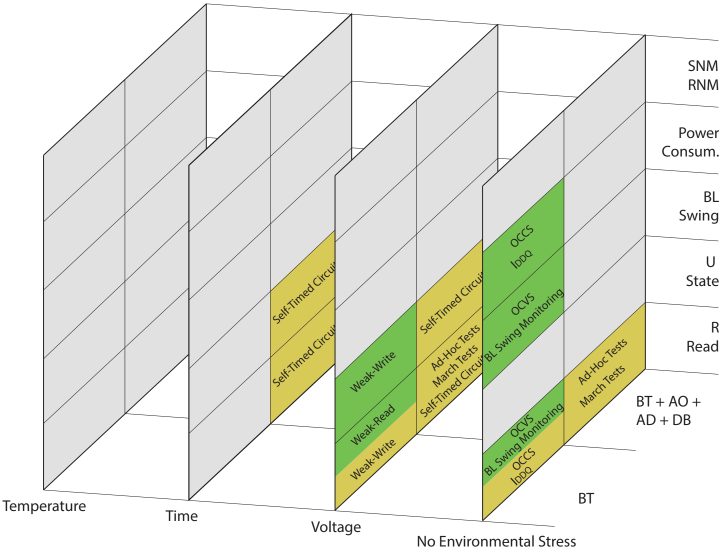


Figure 4.28: Space of existing test solutions for SRAMs.

could become a powerful test solution if new monitoring schemes resilient to PV effects are proposed.

- **Further exploration of SCs' impact**: the previously discussed SCs can be further explored to improve HTD faults coverage. It is known that the industry uses voltage, time, and temperature during manufacturing tests [166] and that these SCs significantly impact the detection of functional ETD faults [133], [176] as well as HTD faults [62], [63]. Unfortunately, no published works used experimental data to relate SCs to HTD faults. Thus, analyses with actual data are still needed to explore further SC's impact on HTD fault detection. New types of stressing conditions might also be identified as efficient ways for testing SRAMs throughout such experiments.

- **Public experimental data is still lacking**: the design and test of SRAMs is critical information for companies. Therefore, it is understandable that there is no publicly available data about the FinFET (SRAM) technology's most critical defects and the relation between SCs and fault coverage. If this type of data were publicly available, test engineers could develop realistic defect models and, consequently, accurate fault modeling and test solutions using appropriate SCs. Only then would it be possible to design FinFET-specific, high-quality test solutions for SRAMs.

- **Relation between functional and parametric HTD faults**: it is straightforward to connect some functional HTD faults to other parametric HTD faults. For example, defects that lead to a BLS parametric fault also lead to a functional random read fault, either ETD or HTD. Nevertheless, the link between some faulty behaviors may be less straightforward. A reduced SNM/RNM could indicate an undefined state functional HTD fault. Nevertheless, it is unknown the extent to which this statement is valid. This relation might exist for some defects, but not all. Thus, a more extensive analysis of the correlation between functional and parametric HTD faulty behavior is still necessary.

- **Test solutions for noise margin faults are missing**: no test solution guarantees the detection of the SNM/RNM parametric HTD faults. The lack of appropriate test solutions can negatively impact applications that require a high-reliability level, such as the aerospace and the automotive market [177]. SNM faults could be detected by test solutions that directly apply stress in every array cell; yet, such an approach's high costs make it unfeasible. Additionally, it might be possible to improve the detection of RNM faults by using test solutions similar to weak-write test mode, in which reduced stress is applied to the cell during write operations. A more detailed analysis of these faults' characteristics is still necessary to develop efficient test solutions that target SNM/RNM parametric HTD faults.

- **Combining test solutions to improve overall coverage**: some dedicated DFT circuits use the same SCs to detect different faults. For example, the DFTs for RRFs and USF proposed in this thesis use voltage stress. Combining somewhat similar test solutions would increase the overall HTD faults coverage. However, combining test solutions could also mean combining their limitations and overheads;

ideally, these combined test solutions would maximize HTD fault coverage while minimizing overheads.

5

HIERARCHICAL MEMORY DIAGNOSIS FOR FINFET SRAMS

Test solutions are not developed to provide fault information, e.g., fault model, fault location; this is usually provided by memory diagnosis. Diagnosis is a critical enabler for scaled memories as they accelerate yield learning and reduce time to market. They also help understand test escapes, customer returns, and no-trouble-found (NTF) devices.

This section presents an efficient Hierarchical Memory Diagnosis (HMD) approach that accurately diagnoses faults in the entire memory. First, we discuss the difference between detecting and diagnosing a fault. Second, we present a classification for diagnosing schemes. Third, we discuss why a new diagnosing approach is needed for scaled memories. Then, we present the proposed diagnosis scheme. We validate the approach by diagnosing both ETD and HTD faults. We conclude this section by discussing the proposed scheme and comparing it to existing diagnosis schemes.

Parts of this chapter have been accepted for publication at ETS'22 and submitted for publication at TVLSI

5.1. DETECTION VS. DIAGNOSIS

The assessment of a memory chip can be carried out mainly in two ways: memory testing and memory diagnosis. Memory testing applies appropriate and efficient stress conditions to detect a fault. This procedure is usually applied to *all* memory chips once they are fabricated; therefore, they must be fast and incur as little cost as possible. Furthermore, their outcome is summarized in a *pass* or *fail* result. Thus, they do not provide much information regarding the detected fault – or are developed to do so.

Contrarily, memory diagnosis focuses on providing insightful information regarding faulty behavior as much as possible. Memory diagnosis can occur throughout the memory development cycle, from prototyping to customer returns inspection. When prototyping a memory and manufacturing the first chips, it is desired to have satisfying quality and reliability issues alongside a fast yield learning curve. Thus, it is not sufficient to only screen memories for faults; precisely identifying the memory blocks affected by defects becomes a must. Such information is of high value to manufacturing companies, as it facilitates the correction of the manufacturing process, consequently enabling yield ramp up within minimum time.

Moreover, memory diagnosis can also be carried out during manufacturing tests. By diagnosing sample circuits now and then, one can control the quality of the manufacturing process and assure that mass production is not affecting the expected quality of the memory. Finally, diagnosis can also be performed on memory devices returned to the manufacturer due to faulty behavior. These devices are known as customer returns; they have passed the manufacturing test but failed on the field. They undergo a manufacturing test and, if they pass once again, they are classified as *no-trouble-found* (NTF) devices. NTFs are the perfect candidate for memory diagnosis as they represent precisely what the manufacturing test is not covering; they can provide valuable information that can be used to improve manufacturing tests and minimize test escapes.

The final outcome of the diagnosis, regardless of when it is applied in the memory development cycle, is the identification and understanding of failure root causes. After diagnosis, the design, manufacturing process, or test program will be tuned to improve yield, quality, and reliability. Therefore, a well-established memory diagnosis process is essential to reduce the ever-increasing test cost of newer ICs. A complete diagnosis process requires on-chip and off-chip analysis using several algorithmic sequences. It includes a set of diagnostic test algorithms and a method or tool to analyze the collected diagnostic data. This data is further processed to generate a detailed fault report for failure analysis.

5.2. DIAGNOSIS SCHEMES CLASSIFICATION

Memory fault diagnosis has not been given as much importance as the testing of memory chips. However, the growing need for a faster yield ramp-up and shorter time-to-market has heightened the importance of memory fault diagnosis. Due to the confidentiality in which companies surround their diagnose schemes, not much information about the existing memory diagnosis procedures in the industry is openly available, thus hindering drawing experience from existing approaches. Nevertheless, some approaches have been published. They rely on different underlying techniques and principles of operation that can be classified into three categories:

- **Probability-based diagnosis:** these are nondeterministic methods introduced during the early stages of memory diagnosis research. They are based on applying random experiments and generating statistical pass/fail information.
- **Signature-based diagnosis:** they are the most common approach to memory diagnosis. A signature is generated for each fault based on the output of specific diagnostic algorithms; faults are identified by comparing signatures with the obtained output. These techniques make use of fault models and march tests for the diagnosis.
- **Design-for-Diagnosis (DFD):** these techniques use additional hardware for memory diagnosis. They apply diagnostic march algorithms alongside the dedicated hardware to help diagnose the fault and identify the fault location.

We discuss each of the above classes in the remainder of this section. We present their methodologies, provide examples of diagnostic solutions, and highlight their limitations and drawbacks.

5.2.1. PROBABILITY-BASED DIAGNOSIS

Probability-based fault analysis methods for memory diagnosis were introduced in the nineties during the onset memory diagnosis research [26], [51], [178], [179]. They rely on applying a large number of random experiments to the memory. Fault locations are narrowed down by appropriately overlapping faulty addresses, and faults are distinguished by comparing the pass/fail data with statistically generated fault probabilities. The first step in a probability-based diagnosis is to apply random tests to different memory cell blocks, i.e., words.

Each experiment consists of initializing an area, testing the said area, and estimating the probabilities of each of the targeted faults. If results are uncertain, further experiments can be performed to obtain more information. Probability-based diagnosis can guarantee with a degree of confidence that a specific fault type exists in the device being diagnosed. Some diagnosing solutions are also able to determine the fault location for a limited number of faults [51]. Nevertheless, these random methodologies have high complexity, i.e., $O(n^2)$, which leads to high test application times. In summary, they do not have a deterministic nature, have a low fault coverage, and are rather time-consuming in terms of the test time. Furthermore, the fault type and fault location can be identified only for a small set of faults. Therefore, these methods did not find a wide application and were soon replaced by other diagnosis techniques.

5.2.2. SIGNATURE-BASED DIAGNOSIS

Signature-based diagnosis methods apply diagnostic March tests and use the results to identify the fault [53], [55], [56], [180]–[188]. They inherit the benefits related to March tests, i.e., linear test time and proven fault coverage. Diagnostic march tests are march tests that provide unique signatures for targeted faults. Traditionally, the number of bits in a signature equals the number of read operations in the march test; the address in

Table 5.1: Fault signature dictionary for March C-.

Fault Model	1st read	2nd read	3rd read	4th read	5th read	Signature
S0F1	1	0	1	0	1	10101
S1F0	0	1	0	1	0	01010
W1TF0	0	1	0	1	0	01010
W0TF1	0	0	1	0	1	00101

which read operations failed is also recorded. A failed read operation is denoted as a ‘1’ in the signature bit of that particular read operation for that particular memory cell. A ‘0’ in signature denotes that no fault was detected, meaning that either the cell is fault-free or that the fault cannot be detected through the applied test. The resulting signatures for different memory faults are grouped in a fault dictionary; all faults with a unique signature can be distinguished, while faults with the same signature require further processing or cannot be distinguished from each other.

We demonstrate the underlying principle of the signature-based diagnosis by generating the fault dictionary of the well-known March C- algorithm [30]. This algorithm is described as $\Updownarrow(w0)$; $\Uparrow(r0,w1)$; $\Uparrow(r1,w0)$; $\Downarrow(r0,w1)$; $\Downarrow(r1,w0)$; $\Updownarrow(r0)$. In this example, we focus on four FPs: S0F1, S1F0, W1TF0, and W0TF1; these faults were previously defined in Section 3.2.1. Note that fault signatures can be derived for all the fault models covered by March C-. The signatures for each of the analyzed faults are shown in Table 5.1; each signature is divided based on the read operation order. Furthermore, a ‘0’ denotes that the read operation returned the expected value, while ‘1’ denotes the opposite, i.e., an incorrect read output. For example, the S1F0 fault, i.e., the cell is permanently storing ‘0’, is detected by all $r1$ operations. Consequently, the march syndrome for SF0 is “01010”. Comparing these signatures with the obtained signatures during the test can identify the fault type causing the memory to fail. Nevertheless, note that not all algorithms lead to unique signatures; both S1F0 and W1TF0 have the same signature for March C-.

Furthermore, the scope of the existing approaches is also limited as most of the existing signature-based solutions target only the diagnosis of static faults. Moreover, these solutions are generally unable to distinguish between faults from the *entire* memory. Thus, it is impossible to determine which memory component is defective. For example, a signature-based diagnosis approach would indicate that a memory is affected by a transition fault. However, it does not provide any information on the faulty memory block in which the malfunction is caused; it could be caused by a defect in the memory cell in the WD. Such information is helpful for the yield ramp-up and repair schemes.

5.2.3. Design-for-Diagnosis Circuits

Additional circuits can also be used to improve diagnosis coverage. Such circuits, named *design-for-diagnosis* (DFD), are dedicated hardware with the sole purpose of facilitating diagnosis and providing additional information regarding the detected faults. They are usually applied in a specific memory block, e.g., the write driver [58], [59], decoders [57]. Despite their benefits, DFDs also present drawbacks. Adding hardware solutions to target all static and dynamic faults in each memory system component would surely lead to a considerable area overhead and a surge in power consumption. Furthermore, they do not provide any insight into the nature of the fault in the faulty block.

5.3. NEED FOR A NEW APPROACH

Fast yield learning is a crucial aspect of developing scaled memories. An essential tool to accelerate yield learning is a high-quality diagnosis, as it allows the characterization of the root causes of failures. It is also a powerful tool to investigate customer returns as it enables a fast understanding of failure roots in test escapes and NTFs. This information is later provided back to manufacturing companies to eliminate or alleviate these root causes by correcting their design, manufacturing process, and test procedures. Without an adequate memory diagnosis approach that can detect a broad scope of faults, identifying and removing the root causes of failures may significantly increase production costs.

However, new and more complex faults have become more relevant with technology miniaturization. For example, dynamic faults are more likely to occur in scaled memories, such as FinFET SRAMs [57]. Furthermore, HTD faults, whose detection is not guaranteed by simply writing and reading the cell, become more likely to occur with the supply voltage downscaling of scaled memories. Therefore, any high-quality memory diagnosis methodology for scaled memories must also cover these emerging faults. Ideally, a memory diagnosis approach suitable for scaled memories should:

- Target faults in all parts of a memory chip; it not only targets faults in the main memory cell array but also faults in all other circuits surrounding the array, such as decoders, *write drivers* (WDs), *sense amplifiers* (SAs), and pre-chargers (PCs).

- Cover the entire fault space, i.e., all possible faults. It should cover faults with different sensitizing conditions, e.g., static, dynamic, single-cell, and coupling faults, and faults with different detecting conditions, e.g., ETD and HTD faults.

- Be flexible, i.e., the approach can be adapted to the manufacturer's need, and easily extensible, i.e., new diagnosis capabilities are easily integrated with existing ones. New diagnosis capabilities must not impact the existing ones, and signatures must not be recompiled every time a new fault is included in the fault scope.

- Be platform-independent, i.e., it can be applied to any memory. Therefore, it must be suitable for traditional, e.g., planar CMOS, FinFET, and emerging memories, e.g., STT-MRAM, ReRAM. Ideally, it also does not require any specific implementation, such as additional hardware DFD circuits, such as column twisters [57].

Table 5.2 presents a summary of the most well-known diagnosis schemes; it is clear that the presented solutions fail to cover all of the requirements previously discussed. They will inaccurately diagnose faults, leading to the incorrect characterization of failures' root causes. Consequently, any feedback received by the manufacturing company will be unreliable, and modifications made because of this feedback may impact yield learn negatively. Thus, an appropriate diagnosis solution for scaled, embedded memories is still missing.

Table 5.2: Existing Diagnosis Approaches' Limitations

Property	[186]–[188]	[56]	[57]
Covers the Entire Memory	✗	✗	✗
Covers Dynamic Faults	✗	✗	✓
Easily Extensible	✗	✓	✗
Platform-Independent	✓	✓	✗

5.4. HIERARCHICAL MEMORY DIAGNOSIS

In the previous sections, we have discussed the drawbacks and shortcomings of traditional memory diagnosis approaches; there is undoubtedly a need for a new methodology that targets faults in all memory blocks and not only in the memory cell array. Therefore, we propose a new approach, namely *Hierarchical Memory Diagnosis* (HMD), to overcome current shortcomings. In this section, we describe the proposed approach. We first discuss its methodology. Then, we present the fault space definition. Subsequently, we describe its three diagnosis levels.

5.4.1. METHODOLOGY & FLOW

The concept behind HMD is to diagnose memory faults and memory blocks in a hierarchical order, as shown in Fig. 5.1. HMD aims at narrowing down the possible faulty component and applying diagnostic tests accordingly so that the effort put into fault diagnosis is in a well-directed way. In turn, this cuts down the time required for the diagnosis and ensures a fast yield ramp up.

The first step in the HMD methodology is to define the targeted faults, i.e., defining a fault space. This consists of identifying *all* the possible faults in *all* memory blocks. Furthermore, the fault space will be used as the foundation to generate all the diagnosing algorithms. Once the fault space is defined, a series of diagnosis routines, i.e., diagnosis levels, is performed hierarchically. Each level obtains details regarding the sensitized faults and accurately points the diagnosis procedure to the next level. First, given a faulty memory, Diagnosis Level 1 defines the fault location, i.e., the faulty block. Then, Diagnosis Level 2 defines the fault nature, i.e., static or dynamic. Finally, Diagnosis Level 3 third defines which fault from the fault space affects the memory, i.e., the fault model. More details regarding the fault space and diagnosis levels are given next.

5.4.2. FAULT SPACE DEFINITION

A fault space is the set of faults that could occur in a given circuit. It is used as the set of targeted faults when developing a testing algorithm, i.e., the test algorithm ideally covers – sensitizes and detects – all faults in the space. As HMD aims to diagnose faults in the entire memory chip, its fault space consists of faults in all memory blocks. Table 5.3 lists the HMD fault space; it includes static (i.e., faults sensitized by at most one operation) and dynamic (i.e., faults sensitized by more than one operation) faults from the entire memory. Note that row and column decoder faults are grouped as they represent the same faults affecting two distinct groups of cells. Moreover, faults related to a PC are "considered" to belong to the write path as it shares the same column as a WD. Finally, all memory array faults could be single-cell or coupling.

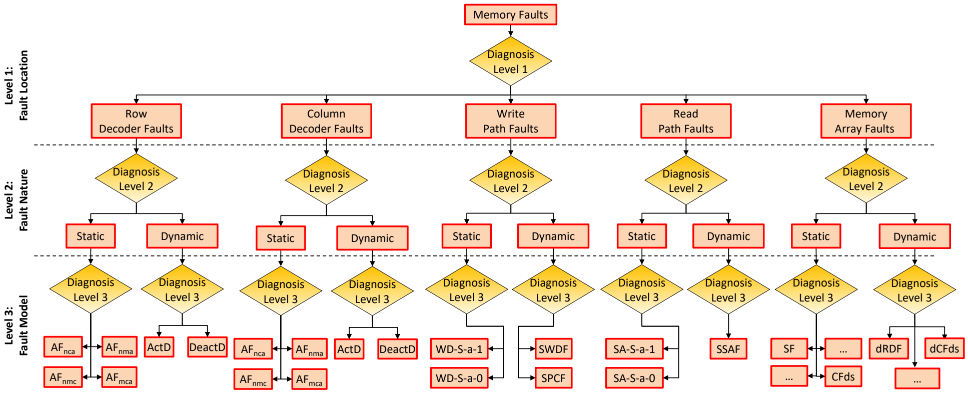


Figure 5.1: Hierarchical Memory Diagnosis Methodology.

The faults listed in Table 5.3 represent a mix of ETD and HTD faults. While ETD faults are faults whose detection is *guaranteed* by writing and reading the memory, HTD faults require additional test effort, e.g., particular stressing conditions or DFT circuits, for a guaranteed detection. Thus, when diagnosing HTD faults, the algorithms applied in each diagnosis level must be adjusted appropriately. To better introduce and explain each diagnosis level, we first only consider the occurrence of ETD faults. Later in this chapter, we provide a more in-depth discussion regarding the appropriate adjustments that must be applied to the HMD approach to enable the diagnosis of ETD and HTD altogether.

5.4.3. DIAGNOSIS LEVEL 1 – FAULT LOCATION

The first diagnosis level identifies the fault location; HMD assumes five faulty blocks: Row Decoder, Column Decoder, Write Path, Read Path, and Memory Array. To accurately identify the faulty block, diagnosis level 1 must sensitize and detect *all* faults in the fault space, static and dynamic. To do so, we developed Alg. 1, which makes use of march notation [26]: $\Uparrow$, $\Downarrow$, and $\Updownarrow$ denote increasing, decreasing, and irrelevant address access order, respectively. $w0$, $w1$, $r0$, $r1$ represent the operations write ‘0’, write ’1’, read ‘0’, and read ‘1’, respectively. The algorithm includes all conditions to detect all static array faults [190]; it also contains bursts of 8 read operations (i.e., $n = 8$) to guarantee the detection of realistic dynamic array faults [57]. Furthermore, it incorporates consecutive, opposite, i.e., *back-to-back*, memory operations required to detect write and read path dynamic faults [143]. The algorithm is executed twice to ensure the detection of both static and dynamic decoder faults: once with a *linear addressing method* to cover static faults, and a second time with *special addressing methods* based on hamming distance such as H=1 and fast-row [165] to cover dynamic faults.

Once the faults are sensitized, diagnosis level 1 moves on to identify the faulty block; this is achieved by creating a bitmap of each memory block, i.e., the pattern of faulty cells, and comparing them to the bitmap generated with the pass/fail information from the March HMD-LVL1 algorithm. Naturally, the bitmap depends on the memory organization; nevertheless, memories generally have a very standard organization, and generic

Table 5.3: HMD Fault Space

Location	**Nature**	**Fault Model**
Decoder (Row or Column)	Static [26]	No Cell, No Address (AF$_{nca}$) No Cell, Multiple Cell (AF$_{nmc}$) No Address, Multiple Address(AF$_{nma}$) Multiple Cell, Multiple Address (AF$_{mca}$)
	Dynamic [165]	Activation Delay (ActD) Deactivation Delay (DeactD) Activation + Deactivation Delay (ActD+DeactD)
Write Path	Static [24]	Stuck-at-1 (WD-S-a-1) Stuck-at-0 (WD-S-a-0)
	Dynamic [143]	Slow Write Driver Fault (SWDF) Slow Precharger Fault (SPCF)
Read Path	Static [24]	Stuck-at-1 (SA-S-a-1) Stuck-at-0 (SA-S-a-0)
	Dynamic [143]	Slow Sense Amplifier Fault (SSAF)
Memory Array	Static [189] (Single-Cell and Coupling)	State Fault (SF) Write Transition Fault (WTF) ... Read Destructive Coupling Fault (CFrd) Disturb Coupling Fault (CFds)
	Dynamic [57] (Single-Cell and Coupling)	dynamic Read Destructive Fault (dRDF) ... dynamic Disturb Coupling Fault (dCFds)

Algorithm 1 March HMD-LVL1

```
M1: $\{\Updownarrow (w1);$
M2: $\Uparrow (w0, r0, r0, r0, r0, r0, r0, r0, r0, w0, w1, w1);$
M3: $\Uparrow (r1, r1, r1, r1, r1, r1, r1, r1, w1, w0, w0);$
M4: $\Downarrow (r0, r0, r0, r0, r0, r0, r0, r0, w0, w1, w1);$
M5: $\Downarrow (r1, r1, r1, r1, r1, r1, r1, r1, w1, w0, w0);$
M6: $\Downarrow (w1, r1, r1, r1, r1, r1, r1, r1, r1, w0);$
M7: $\Uparrow (r0, r0, r0, r0, r0, r0, r0, r0)\}$
```

bitmaps can be derived. Fig. 5.2 shows some of the possible bitmaps; green cells have passed the algorithm, while striped red cells have failed. Each faulty block manifests itself on the memory bitmap in the following manner:

- Faulty **row decoder** on the bitmaps of Fig. 5.2a or b; it representes failures in bits of words coming from a single or multiple rows.

- Faulty **column decoder** on the bitmaps of Fig. 5.2c, d, e, or f; it represents failures in bits of words coming from a single or multiple columns. More specifically, Figs. 5.2c and 5.2e show a local failure, i.e., a single or multiple physical columns are faulty, while Figs. 5.2d and 5.2f show a global failure, i.e., a single or multiple logical column are faulty.

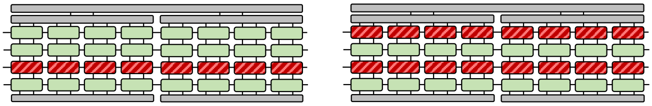
- Faulty **write path** on the bitmap of Fig. 5.2c; it represents faults in a single physical column.

- Faulty **read path** on the bitmap of Fig. 5.2g; the entire column group shares the same faulty SA, and therefore they all fail.

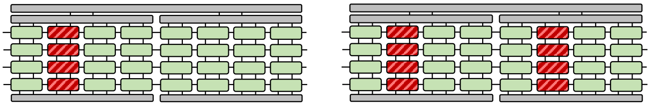
- If there are no matches, it is assumed that the faulty block is the **memory array**; an example of a possible bitmap is shown in 5.2h.

A bitmap can be either *complete*, i.e., all cells in the row/column fail, or *incomplete*, i.e., not all cell fails. An incomplete bitmap can have a *random* pattern due to inaccessible cells leading to random read outputs, or a *partial* pattern due to broken lines preventing full access to the row/column, resulting in a bitmap in which only part of the cells in the row or column has failed. These behaviors do *not* generate the clustered cell patterns of Fig. 5.2h, which are due to coupling faults from neighboring cells.

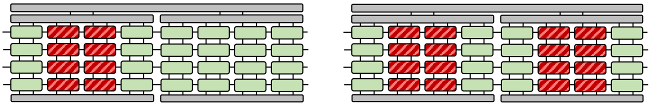
Note that the bitmap of Fig. 5.2c (single faulty column) can represent two different faulty blocks: faulty column decoder or faulty write path; a faulty local column mux will hinder access to a given physical column, while a faulty write path will either prevent the column from being pre-charged or write incorrect values in the cells. Thus, additional effort is necessary to distinguish between these two blocks. This is solved by applying the March sequence $\{\Updownarrow (w0); \Updownarrow (r0, w1); \Updownarrow (w1, r1)\}$ to the faulty column *only*; this leads to a *complete* single column bitmap shown in Fig. 5.2c if and only if the write path is faulty. If the column decoder is faulty, this algorithm leads to an *incomplete* random bitmap or simply no bitmap.



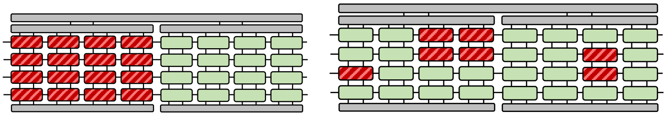
(a) Single faulty row (b) Multiple faulty rows



(c) Single faulty column, local (d) Single faulty column, global



(e) Multiple faulty columns, local (f) Multiple faulty columns, global



(g) Faulty columns sharing SA (h) Single and clustered faulty cells

Figure 5.2: Memory Bitmaps showing healthy (light) and faulty (dark) cells.

5.4.4. DIAGNOSIS LEVEL 2 – FAULT NATURE

Diagnosis level 2 identifies whether the fault is static or dynamic. This is achieved by applying custom diagnostic algorithms to sensitize **static faults** in the addresses that failed in level 1. If no static faults are sensitized, it is assumed the faulty block suffers from **dynamic faults**. As the faulty block is already known, it is no longer necessary to sensitize faults in the entire memory. Therefore, each memory block has its custom-designed diagnostic algorithm.

Table 5.4 lists the developed algorithms targeting only faults within one block. Except for the memory array's algorithm, all algorithms have been developed and tailored for the HMD approach. They are only applied to specific addresses based on the failing addresses information from level 1; this comprises only the failing rows and columns for decoders, write path, and read path. However, if the memory array was identified as the faulty block, then March SS [155] must be applied to the entire memory to ensure that coupling faults are also covered. Note that March SS may also sensitize and detect *some* dynamic faults. If that does happen, no faults will be identified during diagnosis level 3. The HMD then assumes an incorrect diagnosis occurred during level 2, and re-applies level 3 considering dynamic faults.

5.4.5. DIAGNOSIS LEVEL 3 – FAULT MODEL

While diagnosing the fault model might be redundant for small fault spaces, it does provide valuable information for large fault spaces. Thus, level 3 is essential for better un-

Table 5.4: Algorithms applied on Diagnosis Level 2.

Faulty Block	**Algorithm**
Row Decoder	$\{\Updownarrow (w0); \Uparrow (r0, w1, r1)\}$
Column Decoder	$\{\Updownarrow (w0); \Uparrow (r0, w1, r1)\}$
Write Path	$\{\Updownarrow (w0, r0); \Updownarrow (w1, r1)\}$
Read Path	$\{\Updownarrow (w0, r0); \Updownarrow (w1, r1)\}$
Memory Array	March SS [155]

Table 5.5: FC and TC Space for Dynamic Write Path Faults

FC	**Faults**	**TC**
FC1	SWDF	$\{\Updownarrow (w0); \Uparrow^{\mathrm{FR}} (w\mathrm{D}_1); \Downarrow (r\mathrm{D}_1)\}$
FC2	SPCF	$\{\Updownarrow (w0); \Uparrow^{\mathrm{FR}} (r\mathrm{D}_2, w\overline{\mathrm{D}}_2); \Downarrow^{\mathrm{FR}} (r\overline{\mathrm{D}}_2, w\mathrm{D}_2)\}$

derstanding the faults causing failures. It diagnoses fault models using *Fault Classes* (FC) and *Test Classes* (TC) [56]. An FC contains faults with the same sensitizing and detecting conditions that are *externally* indistinguishable, e.g., it is impossible to distinguish a destructive from a non-destructive incorrect read fault. A TC is an algorithm designed to detect a particular FC. Therefore, each FC has a specific TC that targets the FC's faults. Once each TC is executed, a unique signature is generated based on the pass/fail information of each TC. The FC (and consequently the fault model) can be inferred based on the unique signature. Some TCs may detect more than one FC; in these cases, a systematic analysis is required to assess whether all faults have a unique signature. Furthermore, optimizations can be performed by ignoring the TCs of FCs that have similar detection conditions to other FCs. For example, it is unnecessary to execute the TCs for some read-related FCs as these FCs will compulsorily be covered by other write-related FCs.

To demonstrate how FCs and TCs are generated, we present the case of diagnosing dynamic faults in the write path such as *Slow WD Fault* (SWDF) and *Slow PC Fault* (SPCF) [143]. Both are sensitized by performing consecutive (i.e., *back-to-back* [143]) operations to different addresses in the same column but with *inverse* data values $(\mathrm{D}, \overline{\mathrm{D}})$. SWDF requires two transition write operations, $w\mathrm{D}$ after $w\overline{\mathrm{D}}$, to different cells in the same column; in this case, $w\mathrm{D}$ will fail [143]. Similarly, SPCF is detected when performing $r\mathrm{D}$ after a $w\overline{\mathrm{D}}$ (back-to-back). They are externally distinguishable and thus divided into two different FCs with appropriate TCs.

Table 5.5 lists the FCs and TCs for dynamic write path faults; FC1={SWDF} is associated with TC1, FC2={SPCF} with TC2. The addressing method of the back-to-back sensitizing operations is set to fast-row (FR) [143], as required by these faults. TC1 uses a checkerboard data background [143] D_1 = "0101...", while TC2 uses a solid background D_2 = "0000..." [143]. Their coverage, i.e., the FC x TC dictionary, is shown in Table 5.6; '1' represents a detected fault, '0' denotes no fault. While TC1 only covers FC1, TC2 covers FC1 and 2. Nevertheless, even though TC2 covers both FCs, both FCs' signatures are still unique as TC1 only covers FC1. Hence, the fault model can be diagnosed accurately. At the end of diagnosis level 3, the HMD approach has identified the fault location, nature, and model.

Table 5.6: FC x TC Dictionary for Dynamic Write Path Faults

FC	TC1	TC2	Signature
FC1	1	0	10
FC2	0	1	01

5.5. EXPERIMENTAL SETUP

To validate the HMD approach, we have used a *Static Random Access Memory* (SRAM) netlist in SPICE. Unlike the previous memories described in 14 nm, we have used a 7 nm FinFET library [146]; the transistor model is still the same, i.e., BSIM-CMG [145]. We have also adjusted the memory's organization. The memory array comprises 64 rows and 32 physical columns divided into eight 8-bit logical words; hence, four interleaved neighboring physical columns share a single SA. Capacitive loads are applied to BLs and WLs to emulate a 1 kB memory. The memory also includes decoders, WDs, PCs, and timing circuits to generate the controlling signals; a self-timing signal enables and disables WLs. The memory operates on a nominal supply voltage of 0.7 V and a clock frequency of 2.5 GHz.

Resistive opens, shorts, and bridges have been injected into all parts of the memory, one at a time. Once the faults triggered by a defect are verified through electrical simulations, the HMD approach is applied to the faulty memory to assess whether it can accurately diagnose the fault location, fault nature, and fault model.

5.6. HMD FOR ETD FAULTS

ETD faults always lead to incorrect functionality. They are sensitized and detected by applying a write operation to a specific address or cell and later reading the same address or cell. Thus, they are easily detected by simply writing and reading the memory. If the value at the read output does not match the expected value, i.e., the logic value the cell is supposedly holding, a fault is flagged. As ETD faults do not require special sensitizing or detecting conditions, no DFT is required. Thus, the algorithms for diagnosis levels 1 and 2 do not have to be adjusted or modified.

In this section, we use HMD to diagnose ETD faults through SPICE circuit simulations. We discuss multiple study cases, demonstrating the efficacy and accuracy of the HMD approach.

5.6.1. DYNAMIC ROW DECODER FAULTS

Dynamic row decoder faults include Activation (ActD) and Deactivation Delay (DeactD). ActD hinders WLs' activation, i.e., the WL is delayed and not fully activated, while DeactD hinders the WLs' deactivation, leading to simultaneously activating two WLs. These faults are address-sequence dependent, i.e., they are sensitized only in *specific* address transitions.

Fig. 5.3 illustrates how WLs are affected by these faults due to a partial open defect in a pre-decoder. The faulty behavior comes from the pre-decoders failing to decode the address in due time; a timing signal deactivates WLs between operations. In the first operation, WL_x is correctly generated. However, when moving from x to y in the second

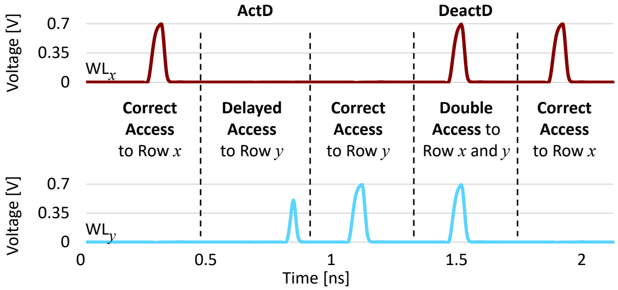


Figure 5.3: An ActD and DeactD Row Decoder Fault.

cycle, WL$_y$ suffers from ActD. Although the subsequent access of WL$_y$ in the third cycle passes correctly, two WLs are activated in the fourth cycle when moving from y back to x: WL$_x$, which is correctly accessed, and WL$_y$, which suffers from DeactD; hence, the simultaneous access of two addresses.

In the presence of the above defect in the pre-decoder under investigation (and hence the dynamic row decoder faults), we apply and validate the HMD methodology as follows:

1. **Level 1:** this level identifies the faulty block. March HMD-LVL1 uses a hamming-distance-based addressing method with fast-row to trigger and detect dynamic row decoder faults. ActD is sensitized and detected by M2 if $x < y$ or by M6 if $x > y$. DeactD is sensitized and detected by M6 and M7, respectively, if $x < y$, or by M2 and M3 if $x > y$. This algorithm generates the fault bitmap shown in Fig. 5.2a, indicating the correct identification of the faulty block. It is a complete bitmap as fast-row addressing order was used; it would have been partial if fast-column was used. Thus, the faulty block is correctly identified as the write path.

2. **Level 2:** this level identifies the fault nature. We apply Table 5.4's row decoder algorithm with *linear* addressing method to the faulty row *only* to sensitize static faults. Simulations showed no faults as there were no row transitions, indicating that the faulty block suffers from dynamic faults.

3. **Level 3:** this level identifies the fault model. Table 5.7 shows the FCs for dynamic row decoder faults and their TCs; FC3 does not require a TC as TC1 and TC2 cover it. The TCs are described by sequences of operations in rows x and y; operations that sensitize the targeted fault are underlined.

Fig. 5.4 shows the simulation of TC1, which targets ActD. The figure shows WL$_x$ and WL$_y$ (faulty row), and the contents of cells in rows x and y. ActD is sensitized by $w1_y$ (3rd operation), and the transition write operation fails. The following $r1_y$ operation detects the fault as it returns '0'.

The simulation of TC2, which targets DeactD, is shown in Fig. 5.5; it shows both WLs, as well as the contents of cells in row sy and the read output. DeactD is sensitized in $w1_x$ (3rd operation); WL$_y$'s is still enabled when accessing x. Note that DeactD is detected

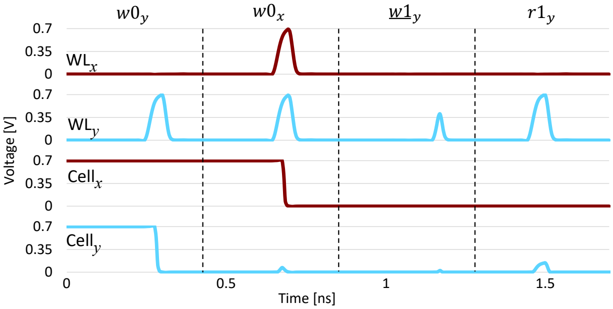


Figure 5.4: Applying TC1 on a memory suffering an ActD and DeactD fault.

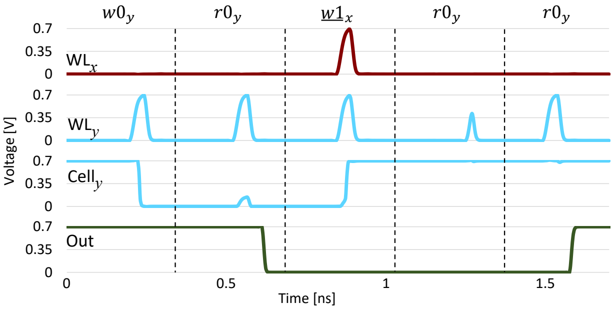


Figure 5.5: Applying TC2 on a memory suffering an ActD and DeactD fault.

in the subsequent read operations. Two read operations are necessary as the first one may be affected by an ActD; the read operation does not access the cell and returns the same value from the last read operation. With the pass/fail results of both TCs, the FC x TC signature dictionary for dynamic row decoder faults can be generated, as shown by Table 5.8. HMD successfully diagnosed the presence of both ActD and DeactD as both TCs failed.

5.6.2. STATIC COUPLING FAULTS

A (two-cell) coupling fault involves an aggressor and a victim cell. The *Fault Primitive* (FP) notation describes such faults as $\langle S_a; S_v/F/R\rangle$; S_a is the aggressor cell's state or sensitizing operation, and S_v is the state of or the operation applied to the victim cell; for a more in-depth discussion regarding these faults, please refer to Section 3.2.1. Static

Table 5.7: FC and TC Space for Dynamic Row Decoder Faults

FC	Faults	TC
FC1	Only ActD	$w0_y, w0_x, \underline{w1}_y, r1_y$
FC2	Only DeactD	$w0_y, r0_y, \underline{w1}_x, r0_y, r0_y$
FC3	Both ActD and DeactD	Detected by TC1 and TC2

Table 5.8: FC x TC Dictionary for Dynamic Row Decoder Faults.

FC	TC1	TC2	Signature
FC1	1	0	10
FC2	0	1	01
FC3	1	1	11

coupling faults in the memory array were triggered by injecting resistive bridge defects [60] connecting different nodes of two adjacent cells. We validate the HMD approach for such faults in the following manner:

1. **Level 1:** running March HMD-LVL1 on the memory under investigation results in the bitmap of Fig. 5.2h (only single cell); this indicates that the faulty block is the memory array.
2. **Level 2:** to identify the fault nature, we applied March SS [155], as shown in Table 5.4. The algorithm failed, indicating the most likely presence of static faults.
3. **Level 3**: to identify which fault model is causing the faulty memory array, we develop appropriate FCs and associated TCs for *all* static memory array faults. They consist of single-cell and coupling faults [189]; These were classified into 8 and 56 FCs, respectively. Table 5.9 lists all the 64 FCs related to static functional memory array faults, and their related TCs. “av” and “va” denote the address relation between aggressor and victim, i.e., $A_a < A_v$ and $A_a > A_v$, respectively; FCs 57 to 64 are single-cell faults, and therefore do not have this address relation. Moreover, FCs whose TC is “–” do not require a TC as multiple TCs of other FCs cover them. Once these TCs are applied to the memory, they generate unique signatures. Table 5.10 illustrates the FC x TC signature dictionary. 26 TCs are necessary to distinguish the 64 FCs, leading to a signature with the pass/fail result of these 26 TCs. The HMD methodology successfully generated a unique signature for each FC, enabling an accurate diagnosis of static coupling faults. If no faults are identified in level 3, the methodology assumes an incorrect diagnosis at level 2. It then moves to diagnose dynamic faults from the same faulty block.

Table 5.9: FC and TC space for Static Coupling Faults

FC	Fault Primitives	TC
FC1	$\langle 0;0/1/-\rangle$av, $\langle 0;r0/0/1\rangle$av, $\langle 0;r0/1/1\rangle$av	-
FC2	$\langle 0;1/0/-\rangle$av, $\langle 0;r1/1/0\rangle$av, $\langle 0;r1/0/0\rangle$av	-
FC3	$\langle 1;0/1/-\rangle$av, $\langle 1;r0/0/1\rangle$av, $\langle 1;r0/1/1\rangle$av	-
FC4	$\langle 1;1/0/-\rangle$av, $\langle 1;r1/1/0\rangle$av, $\langle 1;r1/0/0\rangle$av	-
FC5	$\langle 0;0w1/0/-\rangle$av	-
FC6	$\langle 0;1w0/1/-\rangle$av	-
FC7	$\langle 1;0w1/0/-\rangle$av	-
FC8	$\langle 1;1w0/1/-\rangle$av	-
FC9	$\langle 0;0w0/0/-\rangle$av	-
FC10	$\langle 0;1w1/1/-\rangle$av	$\{\Updownarrow(w1);\Uparrow(w1,r1,w0)\}$
FC11	$\langle 1;0w0/0/-\rangle$av	$\{\Updownarrow(w0);\Uparrow(w0,r0,w1)\}$
FC12	$\langle 1;1w1/1/-\rangle$av	-
FC13	$\langle 0;r0/1/0\rangle$av	$\{\Updownarrow(w0);\Uparrow(w0,r0,r0)\}$
FC14	$\langle 0;r1/0/1\rangle$av	$\{\Updownarrow(w0);\Downarrow(w1,r1,r1)\}$
FC15	$\langle 1;r0/1/0\rangle$av	$\{\Updownarrow(w1);\Downarrow(w0,r0,r0)\}$
FC16	$\langle 1;r1/0/1\rangle$av	-
FC17	$\langle 0;0/1/-\rangle$va, $\langle 0;r0/0/1\rangle$va, $\langle 0;r0/1/1\rangle$va	-
FC18	$\langle 0;1/0/-\rangle$va, $\langle 0;r1/1/0\rangle$va, $\langle 0;r1/0/0\rangle$va	-
FC19	$\langle 1;0/1/-\rangle$va, $\langle 1;r0/0/1\rangle$va, $\langle 1;r0/1/1\rangle$va	-
FC20	$\langle 1;1/0/-\rangle$va, $\langle 1;r1/1/0\rangle$va, $\langle 1;r1/0/0\rangle$va	-
FC21	$\langle 0;0w1/0/-\rangle$va	-
FC22	$\langle 0;1w0/1/-\rangle$va	-
FC23	$\langle 1;0w1/0/-\rangle$va	-
FC24	$\langle 1;1w0/1/-\rangle$va	-
FC25	$\langle 0;0w0/0/-\rangle$va	-
FC26	$\langle 0;1w1/1/-\rangle$va	$\{\Updownarrow(w1);\Downarrow(w1,r1,w0)\}$
FC27	$\langle 1;0w0/0/-\rangle$va	$\{\Updownarrow(w0);\Downarrow(w0,r0,w1)\}$
FC28	$\langle 1;1w1/1/-\rangle$va	-
FC29	$\langle 0;r0/1/0\rangle$va	-
FC30	$\langle 0;r1/0/1\rangle$va	$\{\Updownarrow(w0);\Uparrow(w1,r1,r1)\}$
FC31	$\langle 1;r0/1/0\rangle$va	$\{\Updownarrow(w1);\Uparrow(w0,r0,r0)\}$
FC32	$\langle 1;r1/0/1\rangle$va	$\{\Updownarrow(w1);\Uparrow(w1,r1,r1)\}$
FC33	$\langle r0;0/1/-\rangle$av	-
FC34	$\langle r0;1/0/-\rangle$av	$\{\Updownarrow(w1);\Uparrow(r1,w0,r0)\}$
FC35	$\langle r1;0/1/-\rangle$av	$\{\Updownarrow(w0);\Uparrow(r0,w1,r1)\}$
FC36	$\langle r1;1/0/-\rangle$av	-
FC37	$\langle 0w0;0/1/-\rangle$av	$\{\Updownarrow(w0);\Uparrow(r0,w0)\}$
FC38	$\langle 1w0;0/1/-\rangle$av	$\{\Updownarrow(w0);\Uparrow(r0,w1,w0)\}$
FC39	$\langle 0w1;0/1/-\rangle$av	-
FC40	$\langle 1w1;0/1/-\rangle$av	$\{\Updownarrow(w0);\Uparrow(r0,w1,w1)\}$
FC41	$\langle 0w0;1/0/-\rangle$av	$\{\Updownarrow(w1);\Uparrow(r1,w0,w0)\}$
FC42	$\langle 1w0;1/0/-\rangle$av	-
FC43	$\langle 0w1;1/0/-\rangle$av	$\{\Updownarrow(w1);\Uparrow(r1,w0,w1)\}$
FC44	$\langle 1w1;1/0/-\rangle$av	$\{\Updownarrow(w1);\Uparrow(r1,w1)\}$
FC45	$\langle r0;0/1/-\rangle$va	-
FC46	$\langle r0;1/0/-\rangle$va	$\{\Updownarrow(w1);\Downarrow(r1,w0,r0)\}$
FC47	$\langle r1;0/1/-\rangle$va	$\{\Updownarrow(w0);\Downarrow(r0,w1,r1)\}$
FC48	$\langle r1;1/0/-\rangle$va	-
FC49	$\langle 0w0;0/1/-\rangle$va	$\{\Updownarrow(w0);\Downarrow(r0,w0)\}$
FC50	$\langle 1w0;0/1/-\rangle$va	$\{\Updownarrow(w0);\Downarrow(r0,w1,w0)\}$
FC51	$\langle 0w1;0/1/-\rangle$va	-
FC52	$\langle 1w1;0/1/-\rangle$va	$\{\Updownarrow(w0);\Downarrow(r0,w1,w1)\}$
FC53	$\langle 0w0;1/0/-\rangle$va	$\{\Updownarrow(w1);\Downarrow(r1,w0,w0)\}$
FC54	$\langle 1w0;1/0/-\rangle$va	-
FC55	$\langle 0w1;1/0/-\rangle$va	$\{\Updownarrow(w1);\Downarrow(r1,w0,w1)\}$
FC56	$\langle 1w1;1/0/-\rangle$va	$\{\Updownarrow(w1);\Downarrow(r1,w1)\}$
FC57	$\langle 0/1/-\rangle,\langle 0r0/0/1\rangle,\langle 0r0/1/1\rangle$	-
FC58	$\langle 1/0/-\rangle,\langle 1r1/1/0\rangle,\langle 1r1/0/0\rangle$	-
FC59	$\langle 0w1/0/-\rangle$	-
FC60	$\langle 1w0/1/-\rangle$	-
FC61	$\langle 0w0/1/-\rangle$	-
FC62	$\langle 1w1/0/-\rangle$	-
FC63	$\langle 0r0/1/0\rangle$	-
FC64	$\langle 1r1/0/1\rangle$	-

Table 5.10: FC x TC Dictionary for Static Coupling Faults

FC	**TC10**	**TC11**	**TC13**	**TC14**	**TC15**	**TC26**	**TC27**	**TC30**	**TC31**	**TC32**	**TC34**	**TC35**	**TC37**	**TC38**	**TC40**	**TC41**	**TC43**	**TC44**	**TC46**	**TC47**	**TC49**	**TC50**	**TC52**	**TC53**	**TC55**	**TC56**	**Signature**
FC1	0	0	1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	0	0	1	1	1	1	1	0	0	00100010101011010001111100
FC2	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	10010000001000010001001000
FC3	0	1	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	01001000000100100010000000
FC4	0	0	0	0	0	1	0	1	0	1	0	1	0	0	1	0	1	1	1	0	0	0	0	1	1	1	00000101010100101110000111
FC5	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1	0	0	0	00010000000001000001011000
FC6	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	00000000101001010000010000
FC7	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	00000001000100101000000010
FC8	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	00001000000000001010000110
FC9	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	00100010000010010000100000
FC10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	10000000000000000000001000
FC11	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	01000000000000000000000100
FC12	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	00000100010000100100000001
FC13	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00100000101000000000000000
FC14	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	00010000000000000001000000
FC15	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	00001000000000000010000000
FC16	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000001010100000000000000
FC17	0	1	1	0	1	0	0	0	0	0	0	1	1	1	1	0	0	0	1	0	1	1	0	1	0	0	01101000000111100010110100
FC18	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	00000101000100100010000100
FC19	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	00000010100100010001001000
FC20	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0	1	1	1	0	1	0	0	1	0	1	1	10010000011000011101001011
FC21	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	00000001000101100000010000
FC22	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1	0	0	00001000000001000010010100
FC23	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	00010000000000001001001010
FC24	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	00000000101000011000000010
FC25	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	01100000000010000000100100
FC26	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	00000100000000100000000000
FC27	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	00000010000000010000000000
FC28	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	10000000010000000100001001
FC29	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	00101000000000000010000000
FC30	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000001000100000000000000
FC31	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000000101000000000000000
FC32	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	00010000010000000001000000
FC33	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	00000000000111100000000000
FC34	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000000001000000000000000
FC35	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000000000100000000000000
FC36	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	00000000001000011100000000
FC37	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	00000000000010000000000000
FC38	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	00000000000010000000000000
FC39	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	00000000000101100000000000
FC40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	00000000000000100000000000
FC41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	00000000000000010000000000
FC42	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	00000000001000011000000000
FC43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	00000000000000001000000000
FC44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	00000000000000000100000000
FC45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	00000000000000000001111000
FC46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	00000000000000000010000000
FC47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	00000000000000000001000000
FC48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	00000000000000000010000111
FC49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	00000000000000000000100000
FC50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	00000000000000000000010000
FC51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	00000000000000000001011000
FC52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	00000000000000000000001000
FC53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	00000000000000000000000100
FC54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	00000000000000000010000110
FC55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	00000000000000000000000010
FC56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	00000000000000000000000001
FC57	0	1	1	0	1	0	1	0	1	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	01101010101111110011111100
FC58	1	0	0	1	0	1	0	1	0	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	10010101011100111111001111
FC59	0	0	0	1	0	0	0	1	0	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0	00010001000101101001011010
FC60	0	0	0	0	1	0	0	0	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	00001000101001011010010110
FC61	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	01100010000010010000100100
FC62	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	10000100010000100100001001
FC63	0	0	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	00101000101000000010000000
FC64	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	00010001010100000001000000

Table 5.11: FC & TC Space for Static Read Path Faults

FC	Faults	TC
FC1	SA-SA1	$\{\Updownarrow(w0); \Updownarrow(r0)\}$
FC2	SA-SA0	$\{\Updownarrow(w1); \Updownarrow(r1)\}$

Table 5.12: FC x TC Signature Dictionary for Static Read Path Faults

FC	TC1	TC2	Signature
FC1	1	0	10
FC2	0	1	01

5.6.3. STATIC READ PATH FAULTS

Dynamic row decoder faults include Activation (ActD) and Deactivation Delay (DeactD). ActD hinders WLs' activation, i.e., the WL is delayed and not fully activated, while DeactD hinders the WLs' deactivation, leading to simultaneously activating two WLs. These faults are address-sequence dependent, i.e., they are sensitized only in *specific* address transitions.

Static faults in the read path are related to the value outputted by a logic column's SA. It includes the traditional faults stuck-at-'1' (SA-SA1) and stuck-at-'0' (SA-SA0); they cause the SA or the output latch to output a fixed logic value, regardless of the cell's content or the amplified analogic value. Due to their simple sensitizing and detection requirements, static faults in the read path can be easily diagnosed with HMD's three diagnosis levels.

1. **Level 1:** when running March HMD-LVL1, SA-SA1 is sensitized and detected in the second March element, i.e., M2: $\Uparrow(w0,\ r0,\ r0,\ ...)$. On the other hand, SA-SA0 is sensitized and detected in the first read operation of the third March element, i.e., M3: $\Uparrow(r1, r1, ...)$. Because the faulty read path affects an entire logical column (and thus multiple physical columns), the fault bitmap generated by running March HMD-LVL1 will be the one shown in Fig. 5.2g.

2. **Level 2:** the March algorithm for write path faults described in Table 5.4 includes the conditions to detect SA-SA1, i.e., $\Updownarrow(w0, r0)$, and SA-SA0, i.e., $\Updownarrow(w1, r1)$. Therefore, static faults are detected. Note that the algorithm does not contain includes the sensitizing sequences to detect dynamic read path faults, i.e., $(w1,r0)$ and $(w0,r1)$, and thus cannot detect dynamic faults.

3. **Level 3:** this level discerns the two possible faults, i.e., SA-SA0 and SA-SA1. As these two faults are externally distinguishable, they are divided into two FCs. Table 5.11 shows the FC space for static read path faults, alongside the appropriate TC for each FC. Since each TC only covers the faults within its own FC, the FC x TC signature dictionary generation is straightforward, as shown in Table 5.12. Only one TC will lead to faults; thus, the fault model can be easily diagnosed by applying both TCs and verifying which detected faults.

5.6.4. DYNAMIC SINGLE-CELL FAULTS

With the aggressive down-scaling of embedded memories, dynamic single-cell faults have become even more relevant [133]. Sensitizing a memory cell dynamic fault is no trivial task, as only specific operation sequences performed in the same cell trigger these faults. Dynamic faults will not be sensitized if the applied algorithm is not appropriately designed, leading to test escapes and an inaccurate diagnosis. Therefore, companies must diagnose dynamic faults and understand their root causes.

Dynamic single-cell faults can be described using the FP notation. The particularity of dynamic faults comes from S, which takes the form of $S = x_0 O_1 x_1 ... O_n x_n$, where $x_i \in \{0,1\}$, $i \in \{0,1,...,n\}$, $O \in \{r, w\}$, and $n \geq 2$. According to recent studies [57], it has been stated that a sequence of 8 consecutive read operations (i.e., $n = 8$) can cover the majority of dynamic faults in scaled, embedded memories such as FinFET SRAMs. Thus, we assume a burst of read operations with $n = 8$. For a burst of write operations, we consider $n = 2$. We validate the HMD approach for such faults in the following manner:

1. **Level 1:** March HMD-LVL1 can detect single-cell dynamic faults due to its long operation sequences. While it sensitizes write-related dynamic faults in elements M2, M3, M4, and M5, it only detects in M3, M4, and M5. It sensitizes read-related dynamic faults in all elements but M1.

2. **Level 2:** March SS [155] is used to sensitize single-cell static faults. This algorithm is tailored for static faults, so it is unlikely to trigger dynamic faults. If no faults are detected, it will move to level 3 to diagnose dynamic faults. Nevertheless, it may sensitize read-related dynamic faults detected with two operations. If that does indeed happen, HMD will move on to diagnose static faults instead of dynamics in level 3. However, since no faults will be sensitized, it will assume an incorrect diagnosis in level 2. Therefore, it will reapply level 3, but now considering dynamic faults.

3. **Level 3:** there are six possible sensitizing sequences for dynamic read-related faults: $0r0^8$, $1r1^8$, $0w0r0^8$, $0w1r1^8$, $1w0r0^8$, $1w1r1^8$, where $r0^8$ and $r1^8$ denote a sequence of eight consecutive read operations. Furthermore, there are eight sensitizing sequences for write-related faults: $0w0w0$, $0w0w1$, $0w1w0$, $0w1w1$, $1w0w0$, $1w0w1$, $1w1w0$, and $1w1w1$. Each of the sequences mentioned above constitutes an externally indistinguishable FC, as shown in 5.13. Some TCs will compulsorily cover other FCs besides their own; thus, not all FCs have TCs of their own. The FC x TC signature dictionary is shown in 5.14; the final signature is composed of the results of the ten TCs, i.e., a 10-bit signature. We can see that each FC has a unique signature distinguishable from other FCs' signatures. Therefore, by applying the ten TCs, it is possible to distinguish the fourteen FCs accurately.

5.7. HMD FOR HTD FAULTS

This section shows how to use DFT circuits and the HMD approach to diagnose HTD faults. We discuss two study cases, namely the diagnosis of HTD faults in the column decoder and the memory array.

Table 5.13: FC & TC space for Dynamic Single-Cell Faults

FC	Fault Primitives	TC
FC1	$\langle r0^8/0/1\rangle$, $\langle r0^8/1/1\rangle$, $\langle r0^7/1/0\rangle$, $\langle r0^8/0/1\rangle$, $\langle r0^8/1/1\rangle$	-
FC2	$\langle r1^8/1/0\rangle$, $\langle r1^8/0/0\rangle$, $\langle r1^7/0/1\rangle$, $\langle r1^8/1/0\rangle$, $\langle r1^8/0/0\rangle$	-
FC3	$\langle 0w0r0^8/0/1\rangle$, $\langle 0w0r0^8/1/1\rangle$, $\langle 0w0r0^7/1/0\rangle$, $\langle 0w0r0^8/0/1\rangle$, $\langle 0w0r0^8/1/1\rangle$	$\{\Updownarrow(w0);\Uparrow(w0,r0^8)\}$
FC4	$\langle 1w0r0^8/0/1\rangle$, $\langle 1w0r0^7/1/1\rangle$, $\langle 1w0r0^8/1/0\rangle$, $\langle 1w0r0^8/0/1\rangle$, $\langle 1w0r0^8/1/1\rangle$	-
FC5	$\langle 0w1r1^7/1/0\rangle$, $\langle 0w1r1^8/0/0\rangle$, $\langle 0w1r1^8/0/1\rangle$, $\langle 0w1r1^8/1/0\rangle$, $\langle 0w1r1^7/0/0\rangle$	-
FC6	$\langle 1w1r1^8/1/0\rangle$, $\langle 1w1r1^8/0/0\rangle$, $\langle 1w1r1^8/0/1\rangle$, $\langle 1w1r1^7/1/0\rangle$, $\langle 1w1r1^8/0/0\rangle$	$\{\Updownarrow(w1);\Uparrow(w1,r1^8)\}$
FC7	$\langle 0w0w0/1/-\rangle$	$\{\Updownarrow(w0);\Uparrow(w0,w0,r0)\}$
FC8	$\langle 0w0w1/0/-\rangle$	$\{\Updownarrow(w0);\Uparrow(w0,w1,r1^8)\}$
FC9	$\langle 0w1w0/1/-\rangle$	$\{\Updownarrow(w0);\Uparrow(w1,w0,r0^8)\}$
FC10	$\langle 0w1w1/0/-\rangle$	$\{\Updownarrow(w0);\Downarrow(w1,w1,r1^8)\}$
FC11	$\langle 1w0w0/1/-\rangle$	$\{\Updownarrow(w1);\Downarrow(w0,w0,r0^8)\}$
FC12	$\langle 1w0w1/0/-\rangle$	$\{\Updownarrow(w1);\Uparrow(w0,w1,r1^8)\}$
FC13	$\langle 1w1w0/1/-\rangle$	$\{\Updownarrow(w1);\Uparrow(w1,w0,r0^8)\}$
FC14	$\langle 1w1w1/0/-\rangle$	$\{\Updownarrow(w1);\Uparrow(w1,w1,r1)\}$

Table 5.14: FC x TC Signature Dictionary for Dynamic Single-Cell Faults

FC	TC03	TC06	TC07	TC08	TC09	TC10	TC11	TC12	TC13	TC14	Signature
FC1	1	0	0	0	1	0	1	0	1	0	1000101010
FC2	0	1	0	1	0	1	0	1	0	0	0101010100
FC3	1	0	0	0	0	0	1	0	0	0	1000001000
FC4	0	0	0	0	1	0	0	0	1	0	0000100010
FC5	0	0	0	1	0	0	0	1	0	0	0001000100
FC6	0	1	0	0	0	1	0	0	0	0	0100010000
FC7	0	0	1	0	0	0	0	0	0	0	0010000000
FC8	0	0	0	1	0	0	0	0	0	0	0001000000
FC9	0	0	0	0	1	0	0	0	0	0	0000100000
FC10	0	0	0	0	0	1	0	0	0	0	0000010000
FC11	0	0	0	0	0	0	1	0	0	0	0000001000
FC12	0	0	0	0	0	0	0	1	0	0	0000000100
FC13	0	0	0	0	0	0	0	0	1	0	0000000010
FC14	0	0	0	0	0	0	0	0	0	1	0000000001

Unlike ETD faults, HTD faults do require additional stressing conditions, e.g., DFT circuits, to guarantee – or at least improve – their detection and enable an accurate distinction between HTD and ETD fault. To validate HMD's capabilities regarding HTD faults, we focus on diagnosing random read faults. As previously explained, this HTD behavior occurs when the SA's input during a read operation is significantly reduced so that it is not possible to predict what the SA's output will be; it could be either correct or incorrect.

To accurately diagnose the random read HTD faulty behavior using the HMD approach, we employ the DFT circuit discussed in Section 4.4.3, which biases the SA towards a specific logic value. This DFT effectively detects random read outputs by improving the SA's likelihood of outputting an unexpected logic value, e.g., if the SA is biased towards '1' when performing a $r0$ with a small input, the SA will likely output '1' instead of the expected '0'. In contrast, the DFT can also improve the SA's likelihood of outputting the expected logic value by biasing the SA accordingly, thus disguising the random read output. Both modes of operation are used when applying the HMD approach to either improve or mask the occurrence of HTD faults. Therefore, the DFT is adjusted to bias towards '1' or '0' during a read operation, depending on the cell's content and purpose, i.e., detecting or masking the random read.

Finally, we do not deal with undefined state faults as the existing DFTs to deal with this HTD behavior cannot be easily integrated into the HMD approach. DFTs for undefined states detect fault by identifying cells that have not failed, unlike DFTs for random reads, which detect faults by pointing out which cells have failed. This distinction causes a direct impact on the generation of failure bitmaps and signatures. Therefore, we do not focus on diagnosing undefined state faults for simplification.

5.7.1. STATIC HTD COLUMN DECODER FAULTS

Static Column Decoder faults occur whenever a given address is accessed, regardless of the previously accessed addresses; they have been previously described in Section 3.2.2 and summarized by Fig. 3.3. Such faults are easily sensitized by simply accessing the cell in the corresponding faulty address. Generally speaking, they are also easily detectable; a simple read-write-read sequence is, most of the time, sufficient to detect any static column decoder fault. However, such a sequence becomes inefficient if a read operation returns a random output because two cells with different logic values are accessed simultaneously. This faulty behavior occurs when the column decoder is affected by an AF_{mca}, and cells C_x and C_y store different logic values. When accessing address A_y, both C_x and C_y will be accessed simultaneously. Since they store opposite logic values, each cell will discharge a different BL: one will discharge BL, the other will discharge $\overline{BL}$. Even though these cells are connected to different BL pairs, they will ultimately be connected to the same SA. Their BLs will eventually annul the SA's input, leading to a random read output. If the output does not match the content's of C_y, the fault will be detected and consequently be inaccurately diagnosed as an ETD fault. On the other hand, if the output does match C_y's content, the fault will not be detected, thus leading to a test escape. A random read output means that both these scenarios could happen, which would lead to a nondeterministic and inaccurate diagnostic. Therefore, adjustments should be applied to diagnose these HTD faulty behaviors.

Table 5.15: FC & TC Space for Static Column Decoder Faults

FC	**Faults**	**TC**
FC1	AF_{nca} $\text{AF}_{nmc}, \text{A}_x < \text{A}_y$ $\text{AF}_{nmc}, \text{A}_x > \text{A}_y$	$\{\Updownarrow (w0); \Uparrow (w0, r0, w1, r1)\}$ DFT is not enabled
FC2	$\text{AF}_{nma}, \text{A}_x < \text{A}_y$	Covered by TC4
FC3	$\text{AF}_{nma}, \text{A}_x > \text{A}_y$	Covered by TC5
FC4	$\text{AF}_{mca}, \text{A}_x < \text{A}_y$	$\{\Updownarrow (w1); \Downarrow (r0, w1)\}$ DFT set to bias the SA towards ‘0’
FC5	$\text{AF}_{mca}, \text{A}_x > \text{A}_y$	$\{\Updownarrow (w1); \Uparrow (r0, w1)\}$ DFT set to bias the SA towards ‘0’

1. **Level 1:** static faults occur whenever the faulty column is being accessed. More specifically, the HTD faulty behavior from AF_{mca} is sensitized by writing ‘1’ to C_x then expecting to read ‘0’ from C_y, or the same sequence with opposite logic values. This scenario can occur at M3, M4, and M5. To improve the detection of AF_{mca} faults that may lead to random reads, the DFT is enabled throughout the algorithm execution and configured to detect RRFs rather than mask them.

2. **Level 2:** the March algorithm for static column decoder faults described in Table 5.4 includes the conditions to detect static column decoder faults. Once again, the DFT is enabled throughout the March execution to improve RRF detection.

3. **Level 3:** to diagnose the fault model, we use the DFT configured to mask the occurrence of RRFs. The FC & TC space is shown in Table 5.15. The DFT is not enabled when applying TC1 as this TC does not diagnose any HTD faulty behavior. On the other hand, the DFT is enabled when applying TC4 and TC5 to detect the HTD faulty behavior caused by faults in their respective FCs. By biasing the SA towards ‘0’ to mask the HTD fault, we create a clear distinction between FCs related to AF_{nma} and AF_{mca}, leading to unique signatures. The FC x TC signature dictionary for static column decoder faults is shown in Table 5.16. Unlike other faulty blocks, static faults in decoders may be detected in two different addresses, i.e., A_x and A_y. Therefore, each TC will have a signature composed of two parts: the first half from A_x, and the second from A_y. We can see that because the DFT is set to mask the behavior of the HTD faulty behavior, i.e., the random read output, TC4 and TC5 do not detect any failure at address A_y. If the DFT were set to improve the detection of the HTD faulty behavior, TC4 would detect FC5 at A_y, while TC5 would detect FC4 at A_y. Consequently, FC2 and FC4 would have the same signature, therefore externally indistinguishable; the same applies to FC3 and FC5. Furthermore, if there were no DFT, it would be impossible to determine the signatures of FC4 and FC5 deterministically; they could either have a unique signature or the same as FC2 and FC3, leading to an inaccurate and nondeterministic diagnosis. Therefore, including the DFT into the HMD approach enabled the accurate and complete diagnosis of static faults in the column decoder.

Table 5.16: FC x TC Signature Dictionary for Static Column Decoder Faults

FC	**TC, A$_x$**			**TC, A$_y$**			**Signature**
	TC1	**TC4**	**TC5**	**TC1**	**TC4**	**TC5**	
FC1	1	0	0	0	0	0	100/000
FC2	0	1	0	0	0	1	010/001
FC3	0	0	1	0	1	0	001/010
FC4	0	1	0	0	0	0	010/000
FC5	0	0	1	0	0	0	001/000

5.7.2. STATIC HTD SINGLE-CELL FAULTS

Single-cell static faults occur whenever a given cell is accessed; their single-cell nature indicates that they are not sensitized based on the logic state of other neighboring cells, while their static nature indicates that they are sensitized by at most one operation. They can also be HTD if their detection is not guaranteed by only writing and reading memory cells. Therefore, using a DFT to bias the SA (as used previously to diagnose HTD faults in the column decoder) is also beneficial in diagnosing HTD static faults in the memory array. The HMD approach diagnoses static single-cell faults, including the random read HTD faulty behavior, in the following manner:

1. **Level 1:** to sensitize HTD faults, the DFT is enabled throughout the execution of HMD-LVL1; it is configured to improve RRF detection rather than masking it. Faults can be detected at any read operation.

2. **Level 2:** March SS [155] covers all static faults. Its fault coverage is increased by enabling the DFT to improve RRF detection. Therefore, any RRFs are also detected.

3. **Level 3:** the FC and TC space for Single-Cell HTD static faults is shown in Table 5.17. The faults are described using the FP notation; random read HTD faults are represented by a '?' for the *R*element, i.e., the read output. To distinct HTD from ETD faults, the DFT is adjusted to either improve or mask the detection of HTDs. When applying TC2 and TC4, the DFT is set to bias the SA towards the opposite expected logic value, thus improving the likelihood of detecting an HTD behavior. When applying the other TCs, the DFT is set to bias the SA towards the expected logic value. Therefore, only ETD faults are detected, enabling a deterministic distinction between them and other HTD faults. Table 5.18 lists the FC x TC signature dictionary for Single-Cell HTD static faults. Because ETD and HTD faults are divided in a deterministic manner, FC1 and FC2 have unique signatures; the same applies to FCs3 and FC4. Without the DFT, it would be impossible to deterministically predict the signatures of FC2 and FC4, as other TCs might also detect them. Thus, including the DFT circuit allowed the accurate diagnosis of static random read faults in the memory array.

5.8. DISCUSSION & COMPARISONS

The HMD approach provides many benefits to memory diagnosis. Nevertheless, it is necessary to evaluate and compare it to existing approaches. This section discusses the

Table 5.17: FC & TC space for Static HTD Single-Cell Faults

FC	Fault Primitives	TC
TC1	$\langle 0/1/-\rangle$, $\langle 0r0/0/1\rangle$, $\langle 0r0/1/1\rangle$	Covered by TC2, TC4, TC7, & TC9
TC2	$\langle 0r0/0/?\rangle$, $\langle 0r0/1/?\rangle$	$\{\Updownarrow(w1);\Updownarrow(r1,w0,r0)\}$ DFT is set to bias the SA towards ‘1’
TC3	$\langle 1/0/-\rangle$, $\langle 1r1/1/0\rangle$, $\langle 1r1/0/0\rangle$	Covered by TC2, TC4, TC8, & TC10
TC4	$\langle 1r1/1/?\rangle$, $\langle 1r1/0/?\rangle$	$\{\Updownarrow(w0);\Updownarrow(r0,w1,r1)\}$ DFT is set to bias the SA towards ‘0’
TC5	$\langle 0w1/0/-\rangle$	Covered by TC4 & TC10
TC6	$\langle 1w0/1/-\rangle$	Covered by TC2 & TC9
TC7	$\langle 0w0/1/-\rangle$	$\{\Updownarrow(w0);\Updownarrow(w0,r0)\}$ DFT is set to bias the SA towards ‘0’
TC8	$\langle 1w1/0/-\rangle$	$\{\Updownarrow(w1);\Updownarrow(w1,r1)\}$ DFT is set to bias the SA towards ‘1’
TC9	$\langle 0r0/1/0\rangle$	$\{\Updownarrow(w1);\Updownarrow(w0,r0,r0)\}$ DFT is set to bias the SA towards ‘0’
TC10	$\langle 1r1/0/1\rangle$	$\{\Updownarrow(w0);\Updownarrow(w1,r1,r1)\}$ DFT is set to bias the SA towards ‘1’

Table 5.18: FC x TC Signature Dictionary for Static HTD Single-Cell Faults

FC	TC2	TC4	TC7	TC8	TC9	TC10	Signature
FC1	1	1	1	0	1	0	111010
FC2	1	0	0	0	0	0	100000
FC3	1	1	0	1	0	1	110101
FC4	0	1	0	0	0	0	010000
FC5	0	1	0	0	0	1	010001
FC6	1	0	0	0	1	0	100010
FC7	0	0	1	0	0	0	001000
FC8	0	0	0	1	0	0	000100
FC9	0	0	0	0	1	0	000010
FC10	0	0	0	0	0	1	000001

Table 5.19: Comparison of Diagnosis Methodologies

Property	**[182], [186]–[188]**	**[56]**	**[57]**	**HMD**
Scheme Coverage	Array	Array	Array	Entire Memory
Dynamic Faults Coverage	No	No	Array	Entire Memory
Easily Extensible	No	Yes	No	Yes
Platform-Independent	Yes	Yes	No	Yes

implications of using the HMD approach. We discuss the approach's main benefits, advantages over other approaches, and limitations.

ADDED VALUE

The HMD approach provides significant gains. It can be applied during memory characterization to improve the design and manufacturing process, thus boosting manufacturing yield and speeding yield learning and time to market. It can also be applied during customer returns inspection if address manipulation is allowed by the embedded instruction set; this helps understand test escapes, customer returns, and NTF devices.

KEY DIFFERENTIATORS

Table 5.19 shows a comparison of different diagnose methodologies. Compared to existing diagnosis methodologies, the HMD approach improves memory coverage by shifting the diagnosis focus from the memory cells to the *entire* memory chip, including peripherals and decoders. It also includes dynamic faults, thus enabling the diagnosis of dynamic faults coming from all memory components. The approach is also easily extensible; new diagnosis capabilities can be easily integrated into existing ones without recompiling signatures to guarantee their uniqueness. Furthermore, HMD is platform-independent. While we have validated the approach using 7 nm FinFET technology, HMD can be applied to all sorts of memories organizations and technologies, traditional, e.g., planar CMOS, deep-scaled FinFET, or emerging, e.g., RRAM, STT-MRAM. Moreover, HMD does not require any additional diagnosis circuits, i.e., DFD circuits; yet, existing DFTs circuits can be used to extend HMD's fault coverage.

Finally, the HMD approach also alleviates the pass/fail requirements compared to other approaches. Typically, they require each read operation's status [186], [187], leading to excessive data processing. Furthermore, this information is not available on every memory test platform [56]. The HMD approach only requires each algorithm's pass/fail information rather than each read operation's, significantly alleviating the data processing needed. Moreover, this pass/fail requirement also dismisses the need for applying the *Stop On Nth Error* [57] loop methodology, reducing the time spent applying the algorithms.

LIMITATIONS

To generate the bitmaps of diagnosis level 1, HMD requires memory scrambling information, i.e., the peripheral circuitry distribution. Nevertheless, scrambling is required in all high-quality memory testing and diagnosis methodologies – fault coverage is substantially reduced when scrambling is not considered [191].

A second limitation of the approach is that it cannot indicate the aggressor cell's location, only the address relation between aggressor and victim; galloping-pattern algorithms [25] are necessary to obtain the precise location of aggressor cells. A more in-depth analysis is required to evaluate how they can be aggregated into the HMD methodology.

6

Conclusion

This dissertation addressed fault modeling, test solutions, and a memory diagnosis strategy for FinFET SRAMs. First, the fault space of the entire memory has been determined and compiled into functional fault models. This fault space has been divided based on fault sensitization characteristics, e.g., functional vs. parametric, static vs. dynamic. A fault space validation methodology was proposed; fault space for memory array faults has been verified using the proposed methodology. Based on the simulation results, realistic fault models have been introduced. These have been used to derive efficient, high-quality test solutions. The proposed test solutions focused on hard-to-detect faults, i.e., faults whose detection is not guaranteed by writing and reading the memory. Moreover, they were divided into functional and parametric test solutions. A diagnosis methodology was proposed with the knowledge obtained during the fault modeling and test solution development. This methodology aims at diagnosing all faults in the entire memory hierarchically; it uses adequate stressing conditions to ensure fault detection.

This chapter summarizes the comprehensive investigations and achievements of this dissertation. First, it summarizes the main conclusions presented in each chapter. Then, it proposes future research directions.

6.1. SUMMARY

Chapter 1, "Introduction", establishes the importance of testing and explains some fundamental concepts in this field. It introduces the different testing phases and describes two key aspects related to testing, i.e., test escapes and yield loss. Further, it introduces the state of the art regarding memory testing and memory diagnosis and highlights that technology downscaling brought significant challenges to memory testing and diagnosis – some of which are still unsolved. Adequate fault models, which have to be verified through electrical simulation, need to be established. High-quality test solutions with low test time and high fault coverage still have to be developed. DFT techniques have to be designed to improve fault coverage even further. Finally, diagnosis solutions that cover a vast fault space and include faults from all parts of the memory fault are still missing.

Chapter 2, "FinFET SRAM Technology & Models", presents the background related to the FinFET technology. It introduces the FinFET transistor and its critical physical parameters, e.g., its fin's height (H_{FIN}) and width (T_{FIN}), its gate (and channel) length (L_g), and the number of fins (N_{FIN}). These parameters define the FinFET device's structure and are the manufacturers' guidelines during manufacturing. The manufacturing process can be divided into four key steps: preparing the substrate, etching the fin, forming the gate, and raising the source/drain contacts. Each step can lead to different structural defects, which will impact the transistor's performance. This chapter also surveys SRAMs and how FinFETs can significantly improve the performance of SRAMs. Furthermore, the versatility of the FinFET technology also enables the design of different SRAM cells. It is expected that the FinFET technology will still maintain miniaturization until 2025. Beyond that point, new multi-gate structures are expected to take over the semiconductor industry. The most prominent structure is the *gate-all-around* (GAA) transistor; such a transistor consists of a channel surrounded by the gate on *all* its sides. While this new structure will enable downscaling even further, it is expected to lead to many new issues, both in the near term and long term.

Chapter 3, "Fault Modeling for FinFET SRAMs", establishes the complete framework for memory faults. Different memory classifications based on faults' sensitization and impact are proposed. In summary, three distinctions have been defined: based on the fault's impact, i.e., functional vs. parametric; on the fault's nature, i.e., static vs. dynamic; and on the number of cells involved in the fault's sensitization, i.e., single-cell vs. coupling. The *fault primitive* (FP) concept, which is a compact notation for precisely describing a functional fault, has been introduced. This concept is used to analytically establish a complete framework with which the space of functional fault models can be analyzed. On the other hand, parametric faults are defined using fault models. The fault space for SRAMs is presented using the proposed classifications, divided into static fault space and dynamic fault space. Each fault space contains faults coming from the memory array, decoders, and write and read path. Moreover, the memory array fault space is expanded into single-cell, coupling, and parametric faults. A fault space validation methodology is proposed to assess the fault spaces. This methodology consists of performing electrical simulations with defect injection, measuring the memory's parameters, and classifying the behavior into different fault models. The validation methodology is applied to the single-cell fault space; it results in a list of all functional, parametric, static, and dynamic single-cell faults sensitized in the analyzed defective memory. This type of experiment provides test engineers with precious information regarding the accuracy of the fault space, i.e., which faults are likely to occur in real SRAMs.

Chapter 4, "Test Solutions for FinFET SRAMs", presents different test solutions to detect memory faults. Two distinct types of faults can be identified based on their detection requirements. The first one, *easy-to-detect* (ETD) faults, are faults whose detection is *guaranteed* by simply writing and reading the memory. They have been extensively studied in the past due to their direct impact on a test solution's outcome. The second one, *hard-to-defect* (HTD) faults, are faults whose detection is *not guaranteed* by writing and reading the memory. Parametric HTD faults are severe parametric deviations that cause the memory cell to fail one or more of its specifications; thus, they *do not* impact memory cells' functionality. From the logic functionality point of view, these faults are *undetectable*, as all operations pass correctly. Functional HTD faults, on the other hand, *may* impact the memory's functionality; they are related to random read outputs and undefined cell state. Considering that random effects such as process variation (PV) impact the outcome of functional HTD faults, it is statistically expected that March tests will detect only part of these faults due to incorrect logic behavior. The remaining faults that do not cause incorrect logic behavior will result in test escapes and compromise the circuit's reliability. Only special testing solutions can guarantee the detection of parametric and functional HTD faults.

Nevertheless, existing test solutions are not developed targeting HTD faults. Therefore, their HTD fault coverage is limited. With that in mind, we propose different test solutions for HTD faults. We focus mainly on functional HTD faults; in total, five test solutions are proposed. The first relies on applying appropriate algorithms to sensitize and detect *random read faults* RRFs. However, applying algorithms is not enough. Therefore, two *two design-for-testability* (DFT) circuits are proposed. Both focus on the memory's *sense amplifier* (SA); while one focuses on the SA's sensing phase, the other focuses on the SA's amplification phase. While both improve RRF detection, the DFT focusing on the SA's amplification requires less additional hardware; therefore, it is the recommended test solution for RRFs. The remaining two functional test solutions focus on *undefined state faults* (USFs). Again, it was observed that using algorithms only does not lead to a satisfactory detection rate. In fact, without any additional DFT, USFs can only be detected if the defect generating the USF also leads to RRFs outputting incorrect values. Therefore, we propose a DFT to manipulate the cell's content and facilitate the identification of cells in undefined states.

Finally, a brief discussion over a proposed parametric test solution is presented. The solution introduces on-chip sensors to monitor parameters during memory operations. While the proposed test solution seems feasible and promising at first, an evaluation of PV impact on the solution's detection capabilities showed that on-chip sensors are *not* appropriate to monitor the parameters of scaled memories. Nevertheless, the methodology could be employed if other types of monitoring structures were to be used. A test solutions outlook is discussed, considering the previously existing test approaches and the solutions proposed in this project. While the proposed solutions significantly aggregate to the space of memory test solutions, it is safe to assume that more high-quality test solutions targeting HTD faults can still be developed. Therefore, some of the approaches memory test engineers can explore to improve HTD fault coverage during manufacturing testing are identified and discussed.

Chapter 5, "Hierarchical Memory Diagnosis for FinFET SRAMs", introduces a new memory diagnosis methodology. First, the different types of memory diagnosis are introduced: probability-based, signature-based, and *design-for-diagnosis* circuits. Probability-based diagnoses were proposed at the onset of memory diagnosis research; they did not find a wide application and were soon replaced by other diagnosis techniques. Signature-based techniques are the most common diagnosis methodology; they generate unique signatures for each fault for a given diagnostic algorithm. Finally, DFD circuits improve diagnosing capabilities and reduce diagnosis efforts. Nevertheless, they all have limitations; thus, a new diagnosis approach is necessary. The *hierarchical memory diagnosis* approach aims at narrowing down the possible faulty component and applying diagnostic tests accordingly so that the effort put into fault diagnosis is in a well-directed way. In turn, this cuts down the time required for the diagnosis and ensures a fast yield ramp up. The HMD approach is validated for a total of six case studies – four for ETD faults and two for HTD faults. The proposed methodology successfully diagnosed faults in all parts of the memory, enabling the precise identification of failure roots responsible for yield loss. The approach can be used at characterization to speed up yield learning and time to market and during customer returns' inspection to investigate test escapes and NTFs.

6.2. FUTURE RESEARCH DIRECTIONS

In this dissertation, fault modeling, efficient test solutions, and a comprehensive memory diagnosis have been proposed. Nevertheless, many topics on SRAM testing still need to be explored. These topics include but are not limited to the following.

1) Test of GAA SRAMs: GAA transistors are expected to be the structure semiconductor manufacturers use to replace FinFETs. Such a structure consists of a silicon sheet or wire covered by the gate on all sides. While such a transistor brings many benefits to the performance, its delicate structure will also be prone to new, complex manufacturing defects. These defects may lead to unexpected faulty behaviors and thus faults that have yet to be modeled. Accordingly, these faults may also require new and improved test solutions that can guarantee the detection of these emerging faults. In summary, all the investigations carried out for the FinFET technology and FinFET SRAMs specifically must also be carried out for GAAs and GAA SRAMs.

2) Device-aware testing for CMOS: manufacturing defects have been traditionally modeled as linear resistors; this has been the defect injection methodology industry and academia have adopted for decades. When performing defect injection in SRAM cells, these resistors are placed between cell nodes rather than only the memory's inputs and

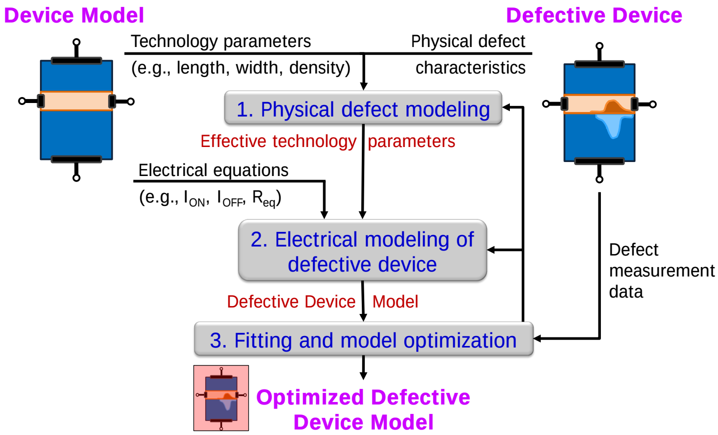


Figure 6.1: *Device-aware testing*'s (DAT) defect modeling methodology [192].

outputs. Such a technique is better known as *cell-aware testing* (CAT); it assumes there are defects *within* a standard cell structure. However, with the development of emerging memories, e.g., STT-MRAM, ReRAM, with new, more complex materials and mechanisms, the injection of linear resistors to model the effects of manufacturing defects became questionable. Mainly, it has been argued that resistors *cannot* model the physical behaviors of these defects. Therefore, using linear resistors to model defects in these technologies will lead to inaccurate fault modeling and low-quality test solutions.

Device-aware testing (DAT) proposes to go one step deeper than CAT; it proposes to develop defect models based on the physical equations of the device. Therefore, it aims to improve the fault modeling and thus obtain realistic fault models that can be used to develop high-quality test solutions. The DAT methodology consists of three steps: (1) defect modeling, (2) fault modeling, and (3) test development. While steps 2 and 3 are similar to the fault modeling and test development performed in this dissertation, step 1 represents a significant change for FinFET SRAMs. The defect modeling methodology is shown in Fig 6.1 [192]; it consists of developing defect models based on technology parameters and electrical equations while fitting the model to physical defect characteristics and defect measurement of real, defective devices. Naturally, companies do not want to disclose the electrical behaviors of their defective transistor devices. Therefore, not much information is publically available regarding the electrical behavior of defective FinFETs. Thus, DAT has not yet been applied to FinFET technology. Nevertheless, DAT has a great potential to improve the quality of FinFET (and GAA) SRAM testing, assuming that one has the necessary characterization and measurement data.

3) Fault space validation for coupling faults: in this dissertation, we validated the fault space of single-cell faults. Due to time constraints, we could not validate the fault space for coupling faults. Therefore, this investigation is still missing. While the fault space for coupling faults can be validated with the proposed validation methodology, some modifications are necessary. Mainly, two aspects must be reviewed: (1) the defect set and

(2) stimulus generation. The first relates to the defects that will be injected to sensitize faulty behaviors. For single-cell faults, defect injection is straightforward; defects should be injected between *all* cell nodes. Therefore, only single-cell faults would be sensitized. However, defects must be injected across multiple cells to sensitize coupling faults, i.e., a bridge defect connecting nodes of two different cells. Accordingly, the defect set must be adapted to include all possible defect scenarios.

The second aspect relates to stimulus generation. The sensitizing sequence must cover at least two cells for coupling faults, i.e., the aggressor and the victim. While this implies a small space of possible Scombinations when considering static faults, this space becomes much more extensive and complex for dynamic faults. All possible combinations of operations, sequences, and states should be analyzed for a complete investigation. However, as the number of operations increases, this becomes unfeasible. Therefore, it is necessary to establish a proper methodology to generate the sensitizing sequence.

4) Industrial analysis with environment-related stress: when testing a circuit, one can apply multiple algorithm- and environment-related *stressing conditions* (SCs). While the detection capabilities of many testing algorithms have been evaluated in real chips, not much has been disclosed about environment-related SCs. For example, changing the operating temperature, frequency, and supply voltage will significantly impact fault sensitization and detection. However, each SC may have a different impact. Therefore, experiments evaluating the impact of SC could be performed in real chips. Efficient test solutions could be derived based on the measurement data.

5) PV-tolerant on-chip sensors: detecting parametric faults is not as trivial as detecting functional faults. While the latter can be detected by performing write and read operations, the former *cannot* be detected by applying operations. Therefore, special testing solutions are required to cover the detection of parametric faults. In this dissertation, we have proposed using on-chip sensors to monitor voltage and current. These sensors can generate unique pulses that enable identifying cells with deviated parameters in a PV-free scenario. Nevertheless, once one considers PV effects, each sensor will output different signals, thus hindering the identification of parametric deviations. While the methodology is effective, the implementation needs to be improved. This drawback could be alleviated by either designing an on-chip sensor tolerant to PV or proposing a new methodology to monitor the memory's parameter. Both solutions require an in-depth PV analysis and extensive simulations.

REFERENCES

[1] R. Schaller, “Moore’s law: Past, present and future”, *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, 1997. DOI: 10.1109/6.591665.

[2] G. E. Moore, “Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff.”, *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 33–35, 2006. DOI: 10.1109/N-SSC.2006.4785860.

[3] S. Borkar, “Microarchitecture and design challenges for gigascale integration”, in *37th International Symposium on Microarchitecture (MICRO-37’04)*, 2004, pp. 3–3. DOI: 10.1109/MICRO.2004.24.

[4] E. Ungersboeck, V. Sverdlov, H. Kosina and S. Selberherr, “Strain engineering for cmos devices”, in *2006 8th International Conference on Solid-State and Integrated Circuit Technology Proceedings*, 2006, pp. 124–127. DOI: 10.1109/ICSICT.2006.306094.

[5] K. Mistry, C. Allen, C. Auth *et al.*, “A 45nm Logic Technology with High-k+Metal Gate Transistors, Strained Silicon, 9 Cu Interconnect Layers, 193 nm Dry Patterning, and 100% Pb-free Packaging”, in *2007, IEEE International Electron Devices Meeting*, Dec. 2007, pp. 247–250. DOI: 10.1109/IEDM.2007.4418914.

[6] S. Natarajan, M. Armstrong, M. Bost *et al.*, “A 32nm logic technology featuring 2nd-generation high-k + metal-gate transistors, enhanced channel strain and 0.171μm^2 SRAM cell size in a 291Mb array”, in *2008 IEEE International Electron Devices Meeting*, Dec. 2008, pp. 1–3. DOI: 10.1109/IEDM.2008.4796777.

[7] J. Colinge, *Silicon-on-Insulator Technology: Materials to VLSI*, ser. Kluwer international series in engineering and computer science: VLSI, computer architecture, and digital signal processing. Springer US, 1991, ISBN: 9780792391500. [Online]. Available: https://books.google.nl/books?id=KkzP0etDp64C.

[8] B. Doyle, S. Datta, M. Doczy *et al.*, “High performance fully-depleted tri-gate cmos transistors”, *IEEE Electron Device Letters*, vol. 24, no. 4, pp. 263–265, 2003. DOI: 10.1109/LED.2003.810888.

[9] W. (Xiong, “Multigate mosfet technology”, in *FinFETs and Other Multi-Gate Transistors*, J.-P. Colinge, Ed. Boston, MA: Springer US, 2008, pp. 49–111, ISBN: 978-0-387-71752-4. DOI: 10.1007/978-0-387-71752-4_2.

[10] F. Arnaud, A. Thean, M. Eller *et al.*, “Competitive and cost effective high-k based 28nm cmos technology for low power applications”, in *2009 IEEE International Electron Devices Meeting (IEDM)*, 2009, pp. 1–4. DOI: 10.1109/IEDM.2009.5424255.

[11] S.-Y. Wu, J. Liaw, C. Lin *et al.*, "A highly manufacturable 28nm cmos low power platform technology with fully functional 64mb sram using dual/tripe gate oxide process", in *2009 Symposium on VLSI Technology*, 2009, pp. 210–211.

[12] D. Bhattacharya and N. K. Jha, "FinFETs: From Devices to Architectures", *Advances in Electronics*, vol. 2014, pp. 1–21, Sep. 2014. DOI: 10.1155/2014/365689.

[13] E. Karl, Y. Wang, Y.-G. Ng, Z. Guo, F. Hamzaoglu, U. Bhattacharya, K. Zhang, K. Mistry and M. Bohr, "A 4.6GHz 162Mb SRAM design in 22nm tri-gate CMOS technology with integrated active VMIN-enhancing assist circuitry", in *2012 IEEE International Solid-State Circuits Conference*, IEEE, Feb. 2012, pp. 230–232, ISBN: 978-1-4673-0377-4. DOI: 10.1109/ISSCC.2012.6176988.

[14] *Ivy Bridge (Microarchitecture)*, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Ivy_Bridge_(microarchitecture)&oldid=1050530898 (visited on 02/11/2021).

[15] M. J. Van Dal, G. Vellianitis, G. Doornbos *et al.*, "Demonstration of scaled Ge p-channel FinFETs integrated on Si", *Technical Digest - International Electron Devices Meeting, IEDM*, pp. 521–524, 2012, ISSN: 01631918. DOI: 10.1109/IEDM.2012.6479089.

[16] V. P.-H. Hu, P.-C. Chiu, A. B. Sachid and C. Hu, "Negative capacitance enables FinFET and FDSOI scaling to 2 nm node", in *2017 IEEE International Electron Devices Meeting (IEDM)*, IEEE, Dec. 2017, pp. 23.1.1–23.1.4, ISBN: 978-1-5386-3559-9. DOI: 10.1109/IEDM.2017.8268443.

[17] N. Loubet, T. Hook, P. Montanini *et al.*, "Stacked nanosheet gate-all-around transistor to enable scaling beyond FinFET", in *2017 Symposium on VLSI Technology*, IEEE, Jun. 2017, T230–T231, ISBN: 978-4-86348-605-8. DOI: 10.23919/VLSIT.2017.7998183.

[18] D. James, *TSMC's 7nm, 5nm, and 3nm "are just numbers... it doesn't matter what the number is"*, Sep. 2019. [Online]. Available: https://www.pcgamesn.com/amd/tsmc-7nm-5nm-and-3nm-are-just-numbers (visited on 02/11/2021).

[19] S. K. Moore, "The node is nonsense", *IEEE Spectrum*, vol. 57, no. 8, pp. 24–30, 2020. DOI: 10.1109/MSPEC.2020.9150552.

[20] *Marvell and TSMC Collaborate to Deliver Data Infrastructure Portfolio on 5nm Technology*, Aug. 2020. [Online]. Available: https://www.hpcwire.com/off-the-wire/marvell-and-tsmc-collaborate-to-deliver-data-infrastructure-portfolio-on-5nm-technology/ (visited on 02/11/2021).

[21] I. Cutress, *'Better Yield on 5nm than 7nm': TSMC Update on Defect Rates for N5*, Aug. 2020. [Online]. Available: https://www.anandtech.com/show/16028/better-yield-on-5nm-than-7nm-tsmc-update-on-defect-rates-for-n5 (visited on 02/11/2021).

[22] A. Frumusanu, *Apple Announces 5nm A14 SoC - Meagre Upgrades, Or Just Less Power Hungry?*, Sep. 2020. [Online]. Available: https://www.anandtech.com/show/16088/apple-announces-5nm-a14-soc-meagre-upgrades-or-less-power-hungry (visited on 03/11/2021).

[23] D. Patel, *Apple's A14 Packs 134 Million Transistors/mm^2, but Falls Short of TSMC's Density Claims*, Oct. 2020. [Online]. Available: https://semianalysis.com/apples-a14-packs-134-million-transistors-mm2-but-falls-far-short-of-tsmcs-density-claims/ (visited on 03/11/2021).

[24] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer US, 2004, ISBN: 9780792379911.

[25] M. Breuer and A. Friedman, *Diagnosis and Reliable Design of Digital Systems*, ser. Digital system design series. Computer Science Press, 1976, ISBN: 9780914894575. [Online]. Available: https://books.google.nl/books?id=Ic4iAAAAMAAJ.

[26] A. van de Goor, *Testing Semiconductor Memories: Theory and Practice*, ser. From Chips to Boards. J. Wiley & Sons, 1991, ISBN: 9780471925873. [Online]. Available: https://books.google.nl/books?id=P4pTAAAAMAAJ.

[27] A. J. van de Goor and C. A. Verruijt, "An Overview of Deterministic Functional RAM Chip Testing", *ACM Comput. Surv.*, vol. 22, no. 1, pp. 5–33, Mar. 1990, ISSN: 0360-0300. DOI: 10.1145/78949.78950.

[28] R. Nair, S. M. Thatte and J. A. Abraham, "Efficient algorithms for testing semiconductor random-access memories", *IEEE Trans. Comput.*, vol. 27, no. 6, pp. 572–576, Jun. 1978, ISSN: 0018-9340. DOI: 10.1109/TC.1978.1675150.

[29] Suk and Reddy, "A march test for functional faults in semiconductor random access memories", *IEEE Transactions on Computers*, vol. C-30, no. 12, pp. 982–985, 1981. DOI: 10.1109/TC.1981.1675739.

[30] M.-C. V. Marinescu, "Simple and efficient algorithms for functional ram testing", in *Proceedings of International Test Conference*, 1982.

[31] J. R. Brown, "Pattern sensitivity in mos memories", in *Digest Symposium on Testing to Integrate Semiconductor Memories into Computer Mainframes*, 1972.

[32] C. A. Papachristou and N. B. Sahgal, "An improved method for detecting functional faults in semiconductor random access memories", *IEEE Trans. Comput.*, vol. 34, no. 2, pp. 110–116, Feb. 1985, ISSN: 0018-9340. DOI: 10.1109/TC.1985.1676547.

[33] R. Dekker, F. Beenker and L. Thijssen, "A realistic fault model and test algorithms for static random access memories", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 6, pp. 567–572, 1990. DOI: 10.1109/43.55188.

[34] P. Mazumder and K. Chakraborty, *Testing and Testable Design of High-Density Random-Access Memories*. USA: Kluwer Academic Publishers, 1996, ISBN: 0792397827.

[35] J. P. Shen, W. Maly and F. J. Ferguson, "Inductive fault analysis of mos integrated circuits", *IEEE Design Test of Computers*, vol. 2, no. 6, pp. 13–26, 1985. DOI: 10.1109/MDT.1985.294793.

[36] W. Maly, "Modeling of lithography related yield losses for cad of vlsi circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 4, no. 3, pp. 166–177, 1985. DOI: 10.1109/TCAD.1985.1270112.

[37] K. Kinoshita and K. Saluja, “Built-in testing of memory using an on-chip compact testing scheme”, *IEEE Transactions on Computers*, vol. 35, no. 10, pp. 862–870, Aug. 1986, ISSN: 1557-9956. DOI: 10.1109/TC.1986.1676677.

[38] P. Mazumder and J. H. Patel, “An efficient design of embedded memories and their testability analysis using Markov chains”, *Journal of Electronic Testing 1992 3:3*, vol. 3, no. 3, pp. 235–250, Aug. 1992, ISSN: 1573-0727. DOI: 10.1007/BF00134733.

[39] K. K. Saluja, S. H. Sng and K. Kinoshita, “Built-in self-testing ram: A practical alternative”, *IEEE Design Test of Computers*, vol. 4, no. 1, pp. 42–51, 1987. DOI: 10.1109/MDT.1987.295113.

[40] H. Koike, T. Takeshima and M. Takada, “A bist scheme using microprogram rom for large capacity memories”, in *Proceedings. International Test Conference 1990*, 1990, pp. 815–822. DOI: 10.1109/TEST.1990.114099.

[41] I. M. Ratiu and H. B. Bakoglu, “Pseudorandom built-in self-test methodology and implementation for the ibm risc system/6000 processor”, *IBM Journal of Research and Development*, vol. 34, no. 1, pp. 78–84, 1990. DOI: 10.1147/rd.341.0078.

[42] A. Meixner and J. Banik, “Weak write test mode: An sram cell stability design for test technique”, in *Proceedings International Test Conference 1996. Test and Design Validity*, 1996, pp. 309–318. DOI: 10.1109/TEST.1996.556976.

[43] C. F. Hawkins and J. M. Soden, “Electrical characteristics and testing considerations for gate oxide shorts in CMOS ICs”, in *Proceedings of the IEEE International Test Conference*, 1985, pp. 544–555.

[44] J. M. Soden, C. F. Hawkins, R. K. Gulati and W. Mao, “Iddqtesting: A review”, in *IDDQ Testing of VLSI Circuits*, R. K. Gulati and C. F. Hawkins, Eds. Boston, MA: Springer US, 1993, pp. 5–17, ISBN: 978-1-4615-3146-3. DOI: 10.1007/978-1-4615-3146-3_1.

[45] I. Schanstra and A. Van De Goor, “Industrial evaluation of stress combinations for march tests applied to srams”, in *International Test Conference*, 1999, pp. 983–992. DOI: 10.1109/TEST.1999.805831.

[46] A. van de Goor and J. de Neef, “Industrial evaluation of dram tests”, in *Design, Automation and Test in Europe Conference and Exhibition, 1999. Proceedings (Cat. No. PR00078)*, 1999, pp. 623–630. DOI: 10.1109/DATE.1999.761194.

[47] S. Hamdioui and A. Van De Goor, “An experimental analysis of spot defects in SRAMs: realistic fault models and tests”, in *Asian Test Symp.*, IEEE, 2000, pp. 131–138, ISBN: 0-7695-0887-1. DOI: 10.1109/ATS.2000.893615.

[48] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri and M. Hage-Hassan, “Resistive-open defects in embedded-SRAM core cells: Analysis and march test solution”, in *13th Asian Test Symp. (ATS)*, Nov. 2004, pp. 266–271. DOI: 10.1109/ATS.2004.75.

[49] A. N. Bhoj, M. O. Simsir and N. K. Jha, “Fault models for logic circuits in the multigate era”, *IEEE Transactions on Nanotechnology*, vol. 11, no. 1, pp. 182–193, 2012. DOI: 10.1109/TNANO.2011.2169807.

[50] M.-F. Chang, W. Fuchs and J. Patel, “Diagnosis and repair of memory with coupling faults”, *IEEE Transactions on Computers*, vol. 38, no. 4, pp. 493–500, 1989. DOI: 10.1109/12.21142.

[51] R. David and A. Fuentes, “Fault diagnosis of rams from random testing experiments”, *IEEE Transactions on Computers*, vol. 39, no. 2, pp. 220–229, 1990. DOI: 10.1109/12.45207.

[52] W. Z. W. Hasan, M. Othman and B. S. Suparjo, “A realistic march-12n test and diagnosis algorithm for sram memories”, in *2006 IEEE International Conference on Semiconductor Electronics*, 2006, pp. 919–923. DOI: 10.1109/SMELEC.2006.380773.

[53] G. Harutunyan, V. Vardanian and Y. Zorian, “Minimal march-based fault location algorithm with partial diagnosis for all static faults in random access memories”, in *2006 IEEE Design and Diagnostics of Electronic Circuits and systems*, 2006, pp. 260–265. DOI: 10.1109/DDECS.2006.1649632.

[54] S. M. Al-Harbi, F. Noor and F. M. Al-Turjman, “March dss: A new diagnostic march test for all memory simple static faults”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 9, pp. 1713–1720, Sep. 2007, ISSN: 1937-4151. DOI: 10.1109/TCAD.2007.895609.

[55] A. Ney, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch and A. Virazel, “A signature-based approach for diagnosis of dynamic faults in srams”, in *2008 3rd International Conference on Design and Technology of Integrated Systems in Nanoscale Era*, 2008, pp. 1–6. DOI: 10.1109/DTIS.2008.4540243.

[56] Z. Al-Ars and S. Hamdioui, “Fault diagnosis using test primitives in random access memories”, in *2009 Asian Test Symposium*, 2009, pp. 403–408. DOI: 10.1109/ATS.2009.79.

[57] G. Harutyunyan, S. Martirosyan, S. Shoukourian and Y. Zorian, “Memory Physical Aware Multi-Level Fault Diagnosis Flow”, *IEEE Tran. on Emerging Topics in Computing*, pp. 1–1, 2018. DOI: 10.1109/TETC.2018.2789818.

[58] A. Ney, P. Girard, S. Pravossoudovitch, A. Virazel, M. Bastian and V. Gouin, “A design-for-diagnosis technique for sram write drivers”, in *Design, Autom. & Test in Europe (DATE)*, 2008, pp. 1480–1485. DOI: 10.1109/DATE.2008.4484883.

[59] ——, “An sram design-for-diagnosis solution based on write driver voltage sensing”, in *26th IEEE VLSI Test Symposium (vts 2008)*, 2008, pp. 89–94. DOI: 10.1109/VTS.2008.17.

[60] G. C. Medeiros, M. Fieback, L. Wu, M. Taouil, L. M. B. Poehls and S. Hamdioui, “Hard-to-Detect Fault Analysis in FinFET SRAMs”, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, pp. 1–14, 2021. DOI: 10.1109/TVLSI.2021.3071940.

[61] G. C. Medeiros, C. Cem Gürsoy, L. Wu, M. Fieback, M. Jenihhin, M. Taouil and S. Hamdioui, “A DFT scheme to improve coverage of hard-to-detect faults in FinFET SRAMs”, in *Design, Autom. & Test in Europe (DATE)*, 2020, pp. 792–797. DOI: 10.23919/DATE48585.2020.9116278.

[62] G. C. Medeiros, M. Fieback, M. Taouil, L. B. Poehls and S. Hamdioui, “Detecting random read faults to reduce test escapes in FinFET SRAMs”, in *2021 IEEE Eur. Test Symp. (ETS)*, May 2021.

[63] G. C. Medeiros, M. Fieback, T. S. Copetti, A. Gebregiorgis, M. Taouil, L. B. Poehls and S. Hamdioui, “Improving the detection of undefined state faults in finfet srams”, in *2021 16th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2021, pp. 1–6. DOI: 10.1109/DTIS53253.2021.9505130.

[64] G. Medeiros, L. Bolzani Poehls, M. Taouil, F. Luis Vargas and S. Hamdioui, “A defect-oriented test approach using on-chip current sensors for resistive defects in FinFET SRAMs”, *Microelectronics Reliability*, vol. 88-90, pp. 355–359, Sep. 2018. DOI: 10.1016/j.microrel.2018.07.092.

[65] G. C. Medeiros, M. Taouil, M. Fieback, L. B. Poehls and S. Hamdioui, “DFT scheme for hard-to-detect faults in FinFET SRAMs”, in *European Test Symposium (ETS)*, IEEE, May 2019. DOI: 10.1109/ETS.2019.8791517.

[66] G. Cardoso Medeiros, M. Fieback, A. Gebregiorgis, M. Taouil, L. B. Poehls and S. Hamdioui, “Hierarchical Memory Diagnosis”, in *2022 IEEE Eur. Test Symp. (ETS)*, May 2022, in press.

[67] G. Cardoso Medeiros, M. Fieback, A. Gebregiorgis, M. Taouil, L. M. B. Poehls and S. Hamdioui, “Hierarchical Memory Diagnosis: Dealing With Static and Dynamic Faults in All Parts of a Memory”, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, pp. 1–14, 2022, submitted for publication.

[68] D. Hisamoto, T. Kaga, Y. Kawamoto and E. Takeda, “A fully depleted lean-channel transistor (DELTA)-a novel vertical ultra thin SOI MOSFET”, in *International Technical Digest on Electron Devices Meeting*, IEEE, 1989, pp. 833–836. DOI: 10.1109/IEDM.1989.74182.

[69] J. P. Colinge, *FinFETs and other multi-gate transistors*, J.-P. Colinge, Ed. Boston, MA: Springer US, 2008, pp. 1–340, ISBN: 9780387717517. DOI: 10.1007/978-0-387-71752-4. arXiv: arXiv:1011.1669v3.

[70] Zheng Guo, S. Balasubramanian, R. Zlatanovici, Tsu-Jae King and B. Nikolic, “FinFET-based SRAM design”, in *ISLPED ’05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005.*, IEEE, 2005, pp. 2–7, ISBN: 1-59593-137-6. DOI: 10.1109/LPE.2005.195476.

[71] M. O. Simsir, A. Bhoj and N. K. Jha, “Fault Modeling for FinFET circuits”, in *2010 IEEE/ACM International Symposium on Nanoscale Architectures*, IEEE, Jun. 2010, pp. 41–46, ISBN: 978-1-4244-8020-3. DOI: 10.1109/NANOARCH.2010.5510927.

[72] P. Mishra, A. Muttreja and N. K. Jha, “FinFET Circuit Design”, in *Nanoelectronic Circuit Design*, N. K. Jha and D. Chen, Eds., 2011, ch. 2, pp. 23–54, ISBN: 978-1-4419-7444-0. DOI: 10.1007/978-1-4419-7609-3.

[73] P. Mishra and N. K. Jha, “Low-power FinFET circuit synthesis using surface orientation optimization”, *Design Automation amp Test in Europe Conference amp Exhibition DATE 2010*, pp. 6–9, 2010. DOI: 10.1109/DATE.2010.5457187.

[74] *7 nm lithography process*. [Online]. Available: https://en.wikichip.org/wiki/7_nm_lithography_process.

[75] R. Rettmann, T. McCormack, O. D. Patterson, H. Lin, K. Nummy, D. Poindexter and P. Parries, “BOX breakdown: A novel defect mode in a 14nm SOI FinFET technology”, in *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference, ASMC 2018*, IEEE, Apr. 2018, pp. 70–73, ISBN: 9781538637487. DOI: 10.1109/ASMC.2018.8373220.

[76] S. Sze and K. K. Ng, *Physics of Semiconductor Devices*. Hoboken, NJ, USA: John Wiley & Sons, Inc., Oct. 2006, ISBN: 9780470068328. DOI: 10.1002/0470068329.

[77] S. Adachi, *Properties of Group-IV, III-V and II-VI Semiconductors*. Chichester, UK: John Wiley & Sons, Ltd, Feb. 2005, ISBN: 9780470090343. DOI: 10.1002/0470090340.

[78] S. Oktyabrsky and P. Ye, Eds., *Fundamentals of III-V Semiconductor MOSFETs*. Boston, MA: Springer US, 2010, ISBN: 978-1-4419-1546-7. DOI: 10.1007/978-1-4419-1547-4.

[79] J.-H. Hur and S. Jeon, “Dislocation effects in FinFETs for different III–V compound semiconductors”, *Journal of Physics D: Applied Physics*, vol. 49, no. 15, p. 155 101, Apr. 2016, ISSN: 0022-3727. DOI: 10.1088/0022-3727/49/15/155101.

[80] J. A. del Alamo, “Nanometre-scale electronics with III–V compound semiconductors”, *Nature*, vol. 479, no. 7373, pp. 317–323, Nov. 2011. DOI: 10.1038/nature10677.

[81] P. Majhi, P. Kalra, J. Oh, R. Harris, H. Tseng and R. Jammy, “CMOS Scaling Beyond High-k and Metal Gates”, *Meeting Abstracts*, vol. MA2007-02, no. 25, pp. 1312–1312, Sep. 2007.

[82] Y. Liu, T. Matsukawa, K. Endo *et al.*, “Advanced FinFET CMOS Technology: TiN-Gate, Fin-Height Control and Asymmetric Gate Insulator Thickness 4T-FinFETs”, in *2006 International Electron Devices Meeting*, IEEE, 2006, pp. 1–4, ISBN: 1-4244-0438-X. DOI: 10.1109/IEDM.2006.346953.

[83] A. Milenin, L. Witters, K. Barla and A. Thean, “Self-aligned double patterning process for subtractive Ge fin fabrication at 45-nm pitch”, *Thin Solid Films*, vol. 602, pp. 64–67, 2016, The 9th International Conference on Silicon Epitaxy and Heterostructures, ISSN: 0040-6090. DOI: https://doi.org/10.1016/j.tsf.2015.08.032.

[84] A. B. Sachid and C. Hu, “Denser and More Stable SRAM Using FinFETs With Multiple Fin Heights”, *IEEE Transactions on Electron Devices*, vol. 59, no. 8, pp. 2037–2041, Aug. 2012, ISSN: 0018-9383. DOI: 10.1109/TED.2012.2199759.

[85] M. Fulde, *Variation Aware Analog and Mixed-Signal Circuit Design in Emerging Multi-Gate CMOS Technologies*, ser. Springer Series in Advanced Microelectronics. Dordrecht: Springer Netherlands, 2010, vol. 28, ISBN: 978-90-481-3279-9. DOI: 10.1007/978-90-481-3280-5.

[86] X. Wang, A. R. Brown, Binjie Cheng and A. Asenov, “Statistical variability and reliability in nanoscale FinFETs”, in *2011 International Electron Devices Meeting*, IEEE, Dec. 2011, pp. 5.4.1–5.4.4, ISBN: 978-1-4577-0505-2. DOI: 10.1109/IEDM.2011.6131494.

[87] P. Mishra, A. N. Bhoj and N. K. Jha, “Die-level leakage power analysis of FinFET circuits considering process variations”, in *2010 11th International Symposium on Quality Electronic Design (ISQED)*, IEEE, Mar. 2010, pp. 347–355, ISBN: 978-1-4244-6454-8. DOI: 10.1109/ISQED.2010.5450554.

[88] T. Matsukawa, S. O. Uchi, K. Endo, Y. Ishikawa, H. Yamauchi, Y. X. Liu, J. Tsukada, K. Sakamoto and M. Masahara, “Comprehensive analysis of variability sources of FinFET characteristics”, *Symp. on VLSI Technol.*, pp. 118–119, Jun. 2009, ISSN: 07431562.

[89] K. Eriguchi, “Modeling of defect generation during plasma etching and its impact on electronic device performance—plasma-induced damage”, *Journal of Physics D: Applied Physics*, vol. 50, no. 33, p. 333 001, Aug. 2017. DOI: 10.1088/1361-6463/aa7523.

[90] K. Eriguchi, A. Matsuda, Y. Takao and K. Ono, “Effects of straggling of incident ions on plasma-induced damage creation in “fin”-type field-effect transistors”, *Japanese Journal of Applied Physics*, vol. 53, no. 3S2, 03DE02, Jan. 2014. DOI: 10.7567/JJAP.53.03DE02.

[91] A. H. Gencer, D. Tsamados and V. Moroz, “Fin bending due to stress and its simulation”, in *2013 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, IEEE, Sep. 2013, pp. 109–112, ISBN: 978-1-4673-5736-4. DOI: 10.1109/SISPAD.2013.6650586.

[92] Y. Liu and Q. Xu, “On modeling faults in FinFET logic circuits”, in *Int. Test Conf.*, IEEE, Nov. 2012, pp. 1–9, ISBN: 978-1-4673-1595-1. DOI: 10.1109/TEST.2012.6401565.

[93] D. Chen and N. K. Jha, “Introduction to Nanotechnology”, in *Nanoelectronic Circuit Design*, N. K. Jha and D. Chen, Eds., 2011, ch. 1, pp. 1–22, ISBN: 978-1-4419-7444-0. DOI: 10.1007/978-1-4419-7609-3.

[94] M. Masahara, R. Surdeanu, L. Witters *et al.*, “Demonstration of Asymmetric Gate Oxide Thickness 4-Terminal FinFETs”, in *2006 IEEE international SOI Conferencee Proceedings*, IEEE, Oct. 2006, pp. 165–166, ISBN: 1-4244-0289-1. DOI: 10.1109/SOI.2006.284489.

[95] C. Auth, C. Allen, A. Blattner *et al.*, “A 22nm high performance and low-power cmos technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density mim capacitors”, in *2012 Symposium on VLSI Technology (VLSIT)*, Jun. 2012, pp. 131–132. DOI: 10.1109/VLSIT.2012.6242496.

[96] C.-H. Lin, J. Chang, M. Guillorn, A. Bryant, P. Oldiges and W. Haensch, “Non-planar device architecture for 15nm node: FinFET or trigate?”, in *2010 IEEE International SOI Conference (SOI)*, IEEE, Oct. 2010, pp. 1–2, ISBN: 978-1-4244-9130-8. DOI: 10.1109/SOI.2010.5641060.

[97] A. Goel, S. K. Gupta and K. Roy, “Asymmetric Drain Spacer Extension (ADSE) FinFETs for Low-Power and Robust SRAMs”, *IEEE Transactions on Electron Devices*, vol. 58, no. 2, pp. 296–308, Feb. 2011. DOI: 10.1109/TED.2010.2090421.

[98] P. K. Pal, B. K. Kaushik and S. Dasgupta, “High-performance and robust SRAM cell based on asymmetric dual-k spacer Finfets”, *IEEE Transactions on Electron Devices*, vol. 60, no. 10, pp. 3371–3377, Oct. 2013, ISSN: 00189383. DOI: 10.1109/TED.2013.2278201.

[99] L. Mathew, M. Sadd, B. E. White, A. Vandooren, J. Cobb, T. Stephens and R. Mora, “Finfet with isolated n+ and p+ gate regions strapped with metal and polysilicon”, in *2003 IEEE International Conference on SOI*, IEEE, 2003, pp. 109–110, ISBN: 0-7803-7815-6. DOI: 10.1109/SOI.2003.1242918.

[100] J. Kedzierski, D. Fried, E. Nowak *et al.*, “High-performance symmetric-gate and CMOS-compatible V/sub t/ asymmetric-gate FinFET devices”, in *International Electron Devices Meeting. Technical Digest*, IEEE, 2001, pp. 19.5.1–19.5.4, ISBN: 0-7803-7050-3. DOI: 10.1109/IEDM.2001.979530.

[101] C. W. Lin, M. C.-T. Chao and C. C. Hsu, “Investigation of gate oxide short in FinFETs and the test methods for FinFET SRAMs”, in *Proceedings of the IEEE VLSI Test Symposium*, IEEE, Apr. 2013, pp. 1–6, ISBN: 9781467355438. DOI: 10.1109/VTS.2013.6548929.

[102] R. Dibaj, D. Al-Khalili and M. Shams, “Comprehensive Investigation of Gate Oxide Short in FinFETs”, in *VLSI Test Symp.*, IEEE, Apr. 2017, pp. 1–6. DOI: 10.1109/VTS.2017.7928960.

[103] F. Moradi, S. K. Gupta, G. Panagopoulos, D. T. Wisland, H. Mahmoodi and K. Roy, “Asymmetrically Doped FinFETs for Low-Power Robust SRAMs”, *IEEE Transactions on Electron Devices*, vol. 58, no. 12, pp. 4241–4249, Dec. 2011. DOI: 10.1109/TED.2011.2169678.

[104] J. Kedzierski, M. Ieong, E. Nowak, T. Kanarsky, Y. Zhang, R. Roy, D. Boyd, D. Fried and H.-S. Wong, “Extension and source/drain design for high-performance finfet devices”, *IEEE Transactions on Electron Devices*, vol. 50, no. 4, pp. 952–958, 2003. DOI: 10.1109/TED.2003.811412.

[105] J. Li, J. Prasad, B.-G. Min and Z. Sun, “Effective wet clean method to eliminate unwanted growth SiGe defect in FinFET”, in *2015 26th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, IEEE, May 2015, pp. 152–154, ISBN: 978-1-4799-9930-9. DOI: 10.1109/ASMC.2015.7164458.

[106] M. Wolf, *Modern VLSI design : a systems approach*. PTR Prentice Hall, 1994, p. 468, ISBN: 0135883776.

[107] J. Rabaey, *Digital Integrated Circuits: A Design Perspective*, ser. Prentice Hall electronics and VLSI series. Prentice Hall, 1996, ISBN: 9780131786097. [Online]. Available: https://books.google.nl/books?id=VSpVPgAACAAJ.

[108] T. Song, W. Rim, S. Park *et al.*, “A 10 nm finfet 128 mb sram with assist adjustment system for power, performance, and area optimization”, *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 240–249, 2017. DOI: 10.1109/JSSC.2016.2609386.

[109] V. S. Basker, T. Standaert, H. Kawasaki *et al.*, “A 0.063 µm² finfet sram cell demonstration with conventional lithography using a novel integration scheme with aggressively scaled fin and gate pitch”, in *2010 Symposium on VLSI Technology*, 2010, pp. 19–20. DOI: 10.1109/VLSIT.2010.5556135.

[110] T. Copetti, G. Cardoso Medeiros, M. Taouil, S. Hamdioui, L. Bolzani Poehls and T. Balen, “Evaluation of single event upset susceptibility of FinFET-Based SRAMs with weak resistive defects”, *Journal of Electronic Testing*, vol. 37, no. 3, pp. 383–394, 2021. DOI: 10.1007/s10836-021-05949-x.

[111] H. Farkhani, A. Peiravi, J. M. Kargaard and F. Moradi, “Comparative study of FinFETs versus 22nm bulk CMOS technologies: SRAM design perspective”, in *2014 27th IEEE International System-on-Chip Conference (SOCC)*, IEEE, Sep. 2014, pp. 449–454, ISBN: 978-1-4799-3378-5. DOI: 10.1109/SOCC.2014.6948971.

[112] H. Kawasaki, V. S. Basker, T. Yamashita *et al.*, “Challenges and solutions of FinFET integration in an SRAM cell and a logic circuit for 22 nm node and beyond”, in *2009 IEEE International Electron Devices Meeting (IEDM)*, IEEE, Dec. 2009, pp. 1–4, ISBN: 978-1-4244-5639-0. DOI: 10.1109/IEDM.2009.5424366.

[113] M. Guillorn, J. Chang, A. Pyzyna *et al.*, “A 0.021 μm^2 trigate SRAM cell with aggressively scaled gate and contact pitch”, *2011 Symposium on VLSI Technology - Digest of Technical Papers*, vol. 5, no. 2010, pp. 64–65, 2011, ISSN: 0743-1562.

[114] K. Endo, S. I. O’uchi, Y. Ishikawa *et al.*, “Independent-gate four-terminal FinFET SRAM for drastic leakage current reduction”, in *Proceedings - 2008 IEEE International Conference on Integrated Circuit Design and Technology, ICICDT*, IEEE, Jun. 2008, pp. 63–66, ISBN: 9781424418114. DOI: 10.1109/ICICDT.2008.4567247.

[115] S. A. Tawfik and V. Kursun, “Portfolio of FinFET memories: Innovative techniques for an emerging technology”, in *2008 International SoC Design Conference*, IEEE, Nov. 2008, pp. I–101–I–104, ISBN: 978-1-4244-2598-3. DOI: 10.1109/SOCDC.2008.4815583.

[116] A. N. Bhoj and N. K. Jha, “Parasitics-Aware Design of Symmetric and Asymmetric Gate-Workfunction FinFET SRAMs”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 548–561, Mar. 2014. DOI: 10.1109/TVLSI.2013.2252031.

[117] M. Chen, C.-y. Lin and Y. Hou, “A 10 nm Si-based bulk FinFETs 6T SRAM with multiple fin heights technology for 25% better static noise margin”, *Symposium on VLSI Technology*, vol. 296, no. 2011, p. 2013, 2013, ISSN: 07431562.

[118] D. Burnett, S. Parihar, H. Ramamurthy and S. Balasubramanian, “Finfet sram design challenges”, in *2014 IEEE International Conference on IC Design Technology*, 2014, pp. 1–4. DOI: 10.1109/ICICDT.2014.6838606.

[119] C.-H. Jan, U. Bhattacharya, R. Brain *et al.*, “A 22nm soc platform technology featuring 3-d tri-gate and high-k/metal gate, optimized for ultra low power, high performance and high density soc applications”, in *2012 International Electron Devices Meeting*, 2012, pp. 3.1.1–3.1.4. DOI: 10.1109/IEDM.2012.6478969.

[120] V. P.-H. Hu, M.-L. Fan, C.-Y. Hsieh, P. Su and C.-T. Chuang, “FinFET SRAM Cell Optimization Considering Temporal Variability Due to NBTI/PBTI, Surface Orientation and Various Gate Dielectrics”, *IEEE Transactions on Electron Devices*, vol. 58, no. 3, pp. 805–811, Mar. 2011. DOI: 10.1109/TED.2010.2099661.

[121] S. Gangwal, S. Mukhopadhyay and K. Roy, “Optimization of Surface Orientation for High-Performance, Low-Power and Robust FinFET SRAM”, in *IEEE Custom Integrated Circuits Conference 2006*, IEEE, Sep. 2006, pp. 433–436, ISBN: 1-4244-0076-7. DOI: 10.1109/CICC.2006.321009.

[122] H. Dadgour, Kazuhiko Endo, V. De and K. Banerjee, “Modeling and analysis of grain-orientation effects in emerging metal-gate devices and implications for SRAM reliability”, in *2008 IEEE International Electron Devices Meeting*, IEEE, Dec. 2008, pp. 1–4, ISBN: 978-1-4244-2377-4. DOI: 10.1109/IEDM.2008.4796792.

[123] M. Kang, S. C. Song, S. H. Woo *et al.*, “FinFET SRAM Optimization With Fin Thickness and Surface Orientation”, *IEEE Transactions on Electron Devices*, vol. 57, no. 11, pp. 2785–2793, Nov. 2010. DOI: 10.1109/TED.2010.2065170.

[124] M.-L. Fan, V. P.-H. Hu, C.-Y. Hsieh, P. Su and C.-T. Chuang, “Subthreshold FinFET SRAM cell optimization considering surface-orientation dependent variability”, in *2010 Proceedings of the European Solid State Device Research Conference*, IEEE, Sep. 2010, pp. 198–201, ISBN: 978-1-4244-6658-0. DOI: 10.1109/ESSDERC.2010.5618391.

[125] Semiconductor Industry Association, “International Technology Roadmap for Semiconductors, 2005 Edition, Executive Summary”, Tech. Rep., 2005. [Online]. Available: https://www.semiconductors.org/wp-content/uploads/2018/08/20051_Executive-Summary.pdf.

[126] IRDS, “International roadmap for devices and systems (IRDS), 2021 Edition”, Tech. Rep., Aug. 2021. [Online]. Available: https://irds.ieee.org/editions/2021.

[127] C. Auth, A. Aliyarukunju, M. Asoro *et al.*, “A 10nm high performance and low-power cmos technology featuring 3rd generation finfet transistors, self-aligned quad patterning, contact over active gate and cobalt local interconnects”, in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 29.1.1–29.1.4. DOI: 10.1109/IEDM.2017.8268472.

[128] G. Yeap, S. S. Lin, Y. M. Chen *et al.*, “5nm cmos production technology platform featuring full-fledged euv, and high mobility channel finfets with densest 0.021µm² sram cells for mobile soc and high performance computing applications”, in *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019, pp. 36.7.1–36.7.4. DOI: 10.1109/IEDM19573.2019.8993577.

[129] IRDS, “International roadmap for devices and systems (IRDS), 2021 Edition, Executive Summary”, Tech. Rep., Aug. 2021. [Online]. Available: https://irds.ieee.org/images/files/pdf/2021/2021IRDS_ES.pdf.

6

[130] A. Veloso, B. Parvais, P. Matagne *et al.*, "Junctionless gate-all-around lateral and vertical nanowire FETs with simplified processing for advanced logic and analog/RF applications and scaled SRAM cells", in *2016 IEEE Symposium on VLSI Technology*, IEEE, Jun. 2016, pp. 1–2, ISBN: 978-1-5090-0638-0. DOI: 10.1109/VLSIT.2016.7573409.

[131] IRDS, "International roadmap for devices and systems (IRDS), 2021 Edition, More Moore", Tech. Rep., Aug. 2021. [Online]. Available: https://irds.ieee.org/images/files/pdf/2021/2021IRDS_MM.pdf.

[132] H. Kaeslin, *Digital Integrated Circuit Design From VLSI Architectures to CMOS Fabrication*. 2008, pp. 1–845, ISBN: 9780521882675. DOI: 10.1109/JSSC.2008.917523.

[133] G. Harutyunyan, G. Tshagharyan and Y. Zorian, "Test and repair methodology for finfet-based memories", *IEEE Transactions on Device and Materials Reliability*, vol. 15, no. 1, pp. 3–9, 2015. DOI: 10.1109/TDMR.2015.2397032.

[134] A. van de Goor and Z. Al-Ars, "Functional memory faults: A formal notation and a taxonomy", in *Proceedings 18th IEEE VLSI Test Symposium*, 2000, pp. 281–289. DOI: 10.1109/VTEST.2000.843856.

[135] S. Hamdioui, *Testing Static Random Access Memories: Defects, Fault Models and Test Patterns*, ser. Frontiers in Electronic Testing. Springer US, 2004, vol. 26, ISBN: 9781402077524. DOI: 10.1007/978-1-4757-6706-3. [Online]. Available: https://books.google.nl/books?id=I2IfUN371IwC.

[136] L. Wu, S. Rao, M. Taouil, E. J. Marinissen, G. S. Kar and S. Hamdioui, "Characterization, Modeling and Test of Synthetic Anti-Ferromagnet Flip Defect in STT-MRAMs", in *2020 IEEE International Test Conference (ITC)*, 2020, pp. 1–10. DOI: 10.1109/ITC44778.2020.9325258.

[137] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel and M. B. Hage-Hassan, "Data retention fault in SRAM memories: analysis and detection procedures", in *23rd IEEE VLSI Test Symposium (VTS'05)*, 2005, pp. 183–188. DOI: 10.1109/VTS.2005.37.

[138] S. Sunter, "Analog fault simulation - a hot topic!", in *2020 IEEE Eur. Test Symp. (ETS)*, May 2020. DOI: 10.1109/ETS48528.2020.9131581.

[139] S. Hamdioui and A. Van De Goer, "Address decoder faults and their tests for two-port memories", in *Proceedings. International Workshop on Memory Technology, Design and Testing (Cat. No.98TB100236)*, 1998, pp. 97–103. DOI: 10.1109/MTDT.1998.705954.

[140] S. Hamdioui, Z. Al-Ars and A. van de Goor, "Opens and Delay Faults in CMOS RAM Address Decoders", *IEEE Transactions on Computers*, vol. 55, no. 12, pp. 1630–1639, 2006. DOI: 10.1109/TC.2006.203.

[141] Z. Al-Ars and A. van de Goor, "Approximating infinite dynamic behavior for dram cell defects", in *Proceedings 20th IEEE VLSI Test Symposium (VTS 2002)*, 2002, pp. 401–406. DOI: 10.1109/VTS.2002.1011171.

[142] A. van de Goor, S. Hamdioui and Z. Al-Ars, “Tests for address decoder delay faults in rams due to inter-gate opens”, in *Proceedings. Ninth IEEE European Test Symposium, 2004. ETS 2004.*, 2004, pp. 146–151. DOI: 10.1109/ETSYM.2004.1347646.

[143] A. J. van de Goor, S. Hamdioui and R. Wadsworth, “Detecting faults in the peripheral circuits and an evaluation of SRAM tests”, in *Int. Conf. on Test*, IEEE, Oct. 2004, pp. 114–123. DOI: 10.1109/TEST.2004.1386943.

[144] D. Kraak, M. Taouil, S. Hamdioui, P. Weckx, S. Cosemans and F. Catthoor, “eS-RAM Reliability: Why is it still not optimally solved?”, in *2020 15th Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2020, pp. 1–6. DOI: 10.1109/DTIS48698.2020.9081145.

[145] Y. Chauhan, D. Lu, S. Venugopalan, S. Khandelwal, J. Duarte, N. Paydavosi, A. Niknejad and C. Hu, *FinFET Modeling for IC Simulation and Design: Using the BSIM-CMG Standard*. Elsevier Science, 2015, ISBN: 9780124200852.

[146] Nanoscale Integration and Modeling (NIMO), *Predictive Technology Model (PTM)*, 2012. [Online]. Available: http://ptm.asu.edu/.

[147] I. Agbo, M. Taouil, D. Kraak, S. Hamdioui, H. Kükner, P. Weckx, P. Raghavan and F. Catthoor, “Integral Impact of BTI, PVT Variation, and Workload on SRAM Sense Amplifier”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 4, pp. 1444–1454, 2017. DOI: 10.1109/TVLSI.2016.2643618.

[148] S. S. Sapatnekar, “Overcoming variations in nanometer-scale technologies”, *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 1, pp. 5–18, 2011. DOI: 10.1109/JETCAS.2011.2138250.

[149] M. S. Abadir and H. K. Reghbati, “Functional testing of semiconductor random access memories”, *ACM Comput. Surv.*, vol. 15, no. 3, pp. 175–198, Sep. 1983, ISSN: 0360-0300. DOI: 10.1145/356914.356916.

[150] S. Borri, M. Hage-Hassan, L. Dilillo, P. Girard, S. Pravossoudovitch and A. Virazel, “Analysis of dynamic faults in embedded-SRAMs: Implications for memory test”, *Journal of Electron. Testing*, vol. 21, no. 2, pp. 169–179, Apr. 2005. DOI: 10.1007/s10836-005-6146-1.

[151] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel and M. Hage-Hassan, “Data retention fault in sram memories: Analysis and detection procedures”, in *23rd IEEE VLSI Test Symp. (VTS’05)*, 2005, pp. 183–188. DOI: 10.1109/VTS.2005.37.

[152] R. A. Fonseca, L. Dilillo, A. Bosio, P. Girard, S. Pravossoudovitch, A. Virazel and N. Badereddine, “Analysis of resistive-bridging defects in SRAM core-cells: A comparative study from 90nm down to 40nm technology nodes”, in *2010 15th IEEE Eur. Test Symp. (ETS)*, May 2010, pp. 132–137. DOI: 10.1109/ETSYM.2010.5512768.

[153] E. I. Vatajelu, A. Bosio, L. Dilillo, P. Girard, A. Todri, A. Virazel and N. Badereddine, “Analyzing resistive-open defects in SRAM core-cell under the effect of process variability”, in *2013 18th IEEE Eur. Test Symp. (ETS)*, May 2013. DOI: 10.1109/ETS.2013.6569373.

[154] G. C. Medeiros, L. B. Poehls and F. F. Vargas, “Analyzing the impact of SEUs on SRAMs with resistive-bridge defects”, in *2016 29th Int. Conf. on VLSI Design (VLSID)*, Jan. 2016, pp. 487–492. DOI: 10.1109/VLSID.2016.146.

[155] S. Hamdioui, A. J. van de Goor and M. Rodgers, “March SS: A test for all static simple RAM faults”, in *Int. Workshop on Memory Technol., Des. and Test*, IEEE, Jul. 2002, pp. 95–100. DOI: 10.1109/MTDT.2002.1029769.

[156] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri and M. Hage-Hassan, “Dynamic read destructive fault in embedded-SRAMs: Analysis and march test solution”, in *Eur. Test Symp.*, IEEE, May 2004, pp. 140–145. DOI: 10.1109/ETSYM.2004.1347645.

[157] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale and P. Prinetto, “March AB, march AB1: New march tests for unlinked dynamic memory faults”, in *Int. Conf. on Test*, Oct. 2005, pp. 834–841. DOI: 10.1109/TEST.2005.1584047.

[158] G. Tshagharyan, G. Harutyunyan, Y. Zorian, A. Gebregiorgis, M. S. Golanbari, R. Bishnoi and M. B. Tahoori, “Modeling and Testing of Aging Faults in FinFET Memories for Automotive Applications”, in *2018 IEEE International Test Conference (ITC)*, 2018, pp. 1–10. DOI: 10.1109/TEST.2018.8624890.

[159] S. Davidson, “Towards an understanding of no trouble found devices”, in *IEEE VTS 2005*, May 2005, pp. 147–152. DOI: 10.1109/VTS.2005.86.

[160] H. Villacorta, V. Champac, S. Bota and J. Segura, “FinFET SRAM hardening through des. and technol. param. considering process var.”, in *RADECS 2013*, IEEE, Sep. 2013, pp. 1–7, ISBN: 978-1-4673-5057-0. DOI: 10.1109/RADECS.2013.6937372.

[161] R. D. Yates and D. J. Goodman, *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers.* Wiley, 2005, ISBN: 0-471-27214-0.

[162] E. Seevinck, F. List and J. Lohstroh, “Static-noise margin analysis of mos sram cells”, *IEEE Journal of Solid-State Circuits*, vol. 22, no. 5, pp. 748–754, 1987. DOI: 10.1109/JSSC.1987.1052809.

[163] L. Darsen Duane and H. Chenming, “Compact Models for Future Generation CMOS”, Ph.D. dissertation, Univerity of Califionia, Berkeley, 2011. [Online]. Available: https://escholarship.org/uc/item/7qn9x2jd.

[164] R. L. Boylestad and L. Nashelsky, *Electronic devices and circuit theory.* Pearson, 2013.

[165] S. Hamdioui, Z. Al-Ars, G. N. Gaydadjiev and A. J. van de Goor, “An investigation on capacitive coupling in RAM address decoders”, in *Int. Design and Test Workshop*, IEEE, Dec. 2007, pp. 9–14. DOI: 10.1109/IDT.2007.4437418.

[166] S. Hamdioui, A. J. van de Goor, J. D. Reyes and M. Rodgers, “Memory test experiment: Industrial results and data”, *IEE Proc. - Comput. and Digital Techn.*, vol. 153, no. 1, Jan. 2006. DOI: 10.1049/ip-cdt:20050104.

[167] J. Kinseher, L. B. Zordan, I. Polian and A. Leininger, “Improving SRAM test quality by leveraging self-timed circuits”, in *Des., Test, and Automation in Eur.*, IEEE, Mar. 2016, pp. 984–989.

[168] A. Meixner and J. Banik, “Weak write test mode: An SRAM cell stability design for test technique”, in *Proc. Int. Test Conf. (ITC)*, Nov. 1997, pp. 1043–1052. DOI: 10.1109/TEST.1997.639732.

[169] S. Hamdioui, M. Taouil and N. Z. Haron, “Testing open defects in memristor-based memories”, *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 247–259, 2015. DOI: 10.1109/TC.2013.206.

[170] M.-C. Chen, T.-H. Wu and C.-W. Wu, “A BIST Scheme for Detect. Defec. in FinFET-Based SRAM Circuit”, in *Asian Test Symposium*, IEEE, Oct. 2018, pp. 19–24, ISBN: 978-1-5386-9466-4. DOI: 10.1109/ATS.2018.00015.

[171] H. Balachandran and D. M. H. Walker, “Improvement of SRAM-based failure analysis using calibrated Iddq testing”, in *Proc. of 14th VLSI Test Symp. (VTS)*, Apr. 1996, pp. 130–136. DOI: 10.1109/VTEST.1996.510847.

[172] M. Hashizume, T. Tamesada, T. Koyama and A. J. van de Goor, “CMOS SRAM functional test with quiescent write supply current”, in *Proc. 1998 IEEE Int. Workshop on IDDQ Testing*, Nov. 1998, pp. 4–8. DOI: 10.1109/IDDQ.1998.730724.

[173] M. J. M. Pelgrom, A. C. J. Duinmaijer and A. P. G. Welbers, “Matching properties of mos transistors”, *IEEE Journal of Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1439, 1989. DOI: 10.1109/JSSC.1989.572629.

[174] A. S. Sedra and K. C. (C. Smith, *Mic. circuits*. Oxford Univ. Press, 2010, ISBN: 9780195323030.

[175] F. Lavratti, L. Bolzani, A. Calimera, F. Vargas and E. Macii, “Technique based on On-Chip Current Sensors and Neighbourhood Comparison Logic to detect resistive-open defects in SRAMs”, in *LATW*, Apr. 2013, pp. 1–6, ISBN: 978-1-4799-0597-3. DOI: 10.1109/LATW.2013.6562688.

[176] T. Santos Copetti, T. Balen, E. Brum, C. Aquistapace and L. Poehls, “Comparing the impact of power supply voltage on CMOS- and FinFET-based SRAMs in the presence of resistive defects”, *Journal of Electron. Testing*, May 2020. DOI: 10.1007/s10836-020-05869-2.

[177] M. Shah, S. Ghosh and S. Martin, “A Qual. and Rel. Driven DFT and DFR Strategy for Automot. and Ind. Markets”, in *VLSI Test Symp.*, IEEE, Apr. 2019, pp. 1–1, ISBN: 978-1-7281-1170-4. DOI: 10.1109/VTS.2019.8758640.

[178] W. H. McAnney, P. H. Bardell and V. P. Gupta, “Random testing for stuck-at storage cells in an embedded memory”, in *Proceedings of the 1984 International Test Conference on The Three Faces of Test: Design, Characterization, Production*, ser. ITC’84, Philadelphia, PA: IEEE Computer Society, 1984, pp. 157–166, ISBN: 0818605480.

[179] J. Savir, W. McAnney and S. Vecchio, “Testing for coupled cells in random-access memories”, *IEEE Transactions on Computers*, vol. 40, no. 10, pp. 1177–1180, 1991. DOI: 10.1109/12.93752.

[180] T. Bergfeld, D. Niggemeyer and E. Rudnick, “Diagnostic testing of embedded memories using bist”, in *Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537)*, 2000, pp. 305–309. DOI: 10.1109/DATE.2000.840288.

[181] G. Harutunyan, V. Vardanian and Y. Zorian, “An efficient march-based three-phase fault location and full diagnosis algorithm for realistic two-operation dynamic faults in random access memories”, in *26th IEEE VLSI Test Symposium*, 2008, pp. 95–100. DOI: 10.1109/VTS.2008.33.

[182] J.-F. Li, K.-L. Cheng, C.-T. Huang and C.-W. Wu, “March-based ram diagnosis algorithms for stuck-at and coupling faults”, in *Proceedings ITC*, 2001, pp. 758–767. DOI: 10.1109/TEST.2001.966697.

[183] D. Niggemeyer and E. Rudnick, “Automatic generation of diagnostic march tests”, in *Proceedings 19th IEEE VLSI Test Symposium. VTS 2001*, 2001, pp. 299–304. DOI: 10.1109/VTS.2001.923453.

[184] V. Vardanian and Y. Zorian, “A march-based fault location algorithm for static random access memories”, in *Proceedings of the 2002 IEEE International Workshop on Memory Technology, Design and Testing (MTDT2002)*, 2002, pp. 62–67. DOI: 10.1109/MTDT.2002.1029765.

[185] V. Yarmolik, Y. Klimets, A. van de Goor and S. Demidenko, “Ram diagnostic tests”, in *IEEE International Workshop on Memory Technology, Design and Testing,*, 1996, pp. 100–102. DOI: 10.1109/MTDT.1996.782499.

[186] C.-W. Wang, K.-L. Cheng, J.-N. Lee, Y.-F. Chou, C.-T. Huang and C.-W. Wu, “Fault pattern oriented defect diagnosis for memories”, in *International Test Conference*, vol. 1, 2003, pp. 29–38. DOI: 10.1109/TEST.2003.1270822.

[187] K.-L. Cheng, C.-W. Wang, J.-N. Lee, Y.-F. Chou, C.-T. Huang and C.-W. Wu, “FAME: A fault-pattern based memory failure analysis framework”, in *Int. Conf. on Computer Aided Design (ICCAD)*, 2003, pp. 595–598. DOI: 10.1109/ICCAD.2003.159743.

[188] M. d. Carvalho, P. Bernardi, M. S. Reorda, N. Campanelli, T. Kerekes, D. Appello, M. Barone, V. Tancorre and M. Terzi, “Optimized embedded memory diagnosis”, in *14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2011, pp. 347–352. DOI: 10.1109/DDECS.2011.5783109.

[189] S. Hamdioui, R. Wadsworth, J. Delos Reyes and A. J. Van De Goor, “Memory fault modeling trends: A case study”, *Journal of Electronic Testing: Theory and Applications*, vol. 20, no. 3, pp. 245–255, Jun. 2004, ISSN: 09238174. DOI: 10.1023/B:JETT.0000029458.57095.bb.

[190] G. Harutunvan, V. Vardanian and Y. Zorian, “Minimal march tests for unlinked static faults in random access memories”, in *23rd IEEE VTS*, 2005, pp. 53–59. DOI: 10.1109/VTS.2005.56.

[191] A. van de Goor and I. Schanstra, “Address and data scrambling: Causes and impact on memory tests”, in *Intl. Work. on Elec. Design, Test and Appl.*, 2002, pp. 128–136. DOI: 10.1109/DELTA.2002.994601.

[192] 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)*, 2019, pp. 1–10. DOI: 10.1109/ITC44170.2019.9000134.

Curriculum Vitæ

Guilherme Cardoso Medeiros

Guilherme Cardoso Medeiros was born in 1993 in Porto Alegre, Brazil. He received his B.Sc. degree in Computer Engieering from the Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil, in 2015. He received his M.Sc. degree in Electrical Engineering from the same university in 2017. His master thesis work focused on the development of new test strategies for FinFET SRAMs. In 2017, he joined the Dependable Nano-Computing Group of the Computer Engineering Laboratory at TU Delft under the supervision of Professor Said Hamdioui. He was financially sponsored by RESCUE, an Horizon-2020 MSCA ETN program. His main areas of interest are test strategies for SRAMs, defect and fault modelling for FinFET devices, and emerging memory technologies.

Education

2007–2009	High School Educational Institute João XXIII, Porto Alegre, Brazil
2010–2015	BSc Computer Engineering Pontific Catholic University of Rio Grande do Sul, Porto Alegre, Brazil
2013	Exchange Year at Bucknell University, Lewisburg, U.S.A.
2015–2017	MSc Electrical Engineering Pontific Catholic University of Rio Grande do Sul, Porto Alegre, Brazil
2017–2022	PhD Computer Engineering Technische Universiteit Delft, Delft, the Netherlands *Thesis*: Test and Diagnosis of Hard-to-Detect Faults in FinFET SRAMs *Promotor:* Prof. dr. ir. S. Hamdioui *Copromotor:* Dr. ir. M. Taouil

MAIN PUBLICATIONS

INTERNATIONAL CONFERENCES

6. **G. Cardoso Medeiros**, M. Fieback, A. Gebregiorgis, M. Taouil, L. Bolzani Poehls, S. Hamdioui, "Hierarchical Memory Diagnosis", in *2022 IEEE European Test Symposium (ETS)*, 2022, pp. 1-2, *in press*.

5. **G. Cardoso Medeiros**, M. Fieback, T. Santos Copetti, A. Gebregiorgis, M. Taouil, L. Bolzani Poehls, S. Hamdioui, "Improving the Detection of Undefined State Faults in FinFET SRAMs", in *2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2021, pp. 1-6, doi: 10.1109/DTIS53253.2021.9505130

4. **G. Cardoso Medeiros**, M. Fieback, A. Gebregiorgis, M. Taouil, L. Bolzani Poehls, S. Hamdioui, "Detecting Random Read Faults to Reduce Test Escapes in FinFET SRAMs", in *2021 IEEE European Test Symposium (ETS)*, 2021, pp. 1-6, doi: 10.1109/ETS50041.2021.9465441

3. T. Copetti, **G. Cardoso Medeiros**, M. Taouil, S. Hamdioui, L. B. Poehls, T. Balen, "Evaluating the Impact of Ionizing Particles on FinFET -based SRAMs with Weak Resistive Defects", in *2020 IEEE Latin-American Test Symposium (LATS)*, 2020, pp. 1-6, doi: 10.1109/LATS49555.2020.9093667

2. **G. Cardoso Medeiros**, C. Cem Gürsoy, L. Wu, M. Fieback, M. Jenihhin, M. Taouil, S. Hamdioui, "A DFT Scheme to Improve Coverage of Hard-to-Detect Faults in FinFET SRAMs", in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 792-797, doi: 10.23919/DATE48585.2020.9116278

1. **G. Cardoso Medeiros**, M. Taouil, M. Fieback, L. B. Poehls, S. Hamdioui, "DFT Scheme for Hard-to-Detect Faults in FinFET SRAMs", in *2019 IEEE European Test Symposium (ETS)*, 2019, pp. 1-2, doi: 10.1109/ETS.2019.8791517

INTERNATIONAL JOURNALS

4. **G. Cardoso Medeiros**, M. Fieback, A. Gebregiorgis, M. Taouil, L. Bolzani Poehls, S. Hamdioui, "Hierarchical Memory Diagnosis: Dealing with Static and Dynamic Faults in All Parts of a Memory", in *IEEE Transactions on Very Large Scale Integration Systems (**TVLSI**)*, 2022, *submitted for publication.*

3. T. Copetti, **G. Cardoso Medeiros**, M. Taouil, S. Hamdioui, L. Bolzani Poehls, T. Balen, "Evaluation of Single Event Upset Susceptibility of FinFET-based SRAMs with Weak Resistive Defects", in *Journal of Electronic Test: Theory and Application (**JETTA**)*, vol. **37**, pp. 383–394, 2021, doi: 10.1007/s10836-021-05949-x

2. **G. Cardoso Medeiros**, M. Taouil, L. Bolzani Poehls, S. Hamdioui, "Hard-to-Detect Fault Analysis in FinFET SRAMs," in *IEEE Transactions on Very Large Scale Integration Systems (**TVLSI**)*, vol. **29**, no. 6, pp. 1271-1284, June 2021, doi: 10.1109/TVLSI.2021.3071940

1. **G. Cardoso Medeiros**, L. Bolzani Poehls, M. Taouil, F. Luis Vargas, S. Hamdioui, "A defect-oriented test approach using on-Chip current sensors for resistive defects in FinFET SRAMs", in *Microelectronics Reliability*, vol. **88–90**, pp. 355-359, 2018, doi: 10.1016/j.microrel.2018.07.09

OTHER PUBLICATIONS

INTERNATIONAL CONFERENCES

5. M. Fieback, C. Münch, A. Gebregiorgis, **G. Cardoso Medeiros**, M. Taouil, S. Hamdioui, M. Tahoori, "PVT Analysis for RRAM and STT-MRAM-based Logic Computation-in-Memory", in *2022 IEEE European Test Symposium (ETS)*, 2022, pp. 1-6, *in press*.

4. A. Balakrishnan, **G. Cardoso Medeiros**, C. C. Gürsoy, S. Hamdioui, M. Jenihhin, D. Alexandrescu, "Modeling Soft-Error Reliability Under Variability", in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2021, pp. 1-6, doi: 10.1109/DFT52944.2021.9568295 **(Outstanding Student Research Award)**

3. M. Fieback, **G. Cardoso Medeiros**, A. Gebregiorgis, H. Aziza, M. Taouil, S. Hamdioui, "Intermittent Undefined State Fault in RRAMs", in *2021 IEEE European Test Symposium (ETS)*, 2021, pp. 1-6, doi: 10.1109/ETS50041.2021.9465401 **(Best Paper Award)**

2. M. Fieback, L. Wu, **G. Cardoso Medeiros**, H. Aziza, S. Rao, E. J. Marinissen, M. Taouil, S. Hamdioui, "Device-Aware Test: A New Test Approach Towards DPPB Level", in *2019 IEEE International Test Conference (ITC)*, 2019, pp. 1-10, doi: 10.1109/ITC44170.2019.9000134

1. L. Wu, S. Rao, **G. Cardoso Medeiros**, M. Taouil, E. J. Marinissen, F. Yasin, S. Couet, S. Hamdioui, G. Sankar Kar, "Pinhole Defect Characterization and Fault Modeling for STT-MRAM Testing", in *2019 IEEE European Test Symposium (ETS)*, 2019, pp. 1-6, doi:10.1109/ETS.2019.8791518

INTERNATIONAL JOURNALS

2. Moritz Fieback, **Guilherme Cardoso Medeiros**, Lizhou Wu, Hassen Aziza, Rajendra Bishnoi, Mottaqiallah Taouil, Said Hamdioui, "Defects, Fault Modeling, and Test Development Framework for RRAMs", ACM Journal on Emerging Technologies in Computing Systems (**JETC**) 2022, *in press*.

1. L. Wu, S. Rao, M. Taouil, **G. Cardoso Medeiros**, M. Fieback, E. J. Marinissen, G. Sankar Kar, S. Hamdioui, "Defect and Fault Modeling Framework for STT-MRAM Testing", in *IEEE Transactions on Emerging Topics in Computing (**TETC**)*, vol. **9**, no. 2, pp. 707-723, 1 April-June 2021, doi: 10.1109/TETC.2019.2960375

TU Delft

Quantum & Computer Engineering

RESCUE
European Training Network

9 789463 665650