

Invasive and Non-Invasive Detection of Bias Temperature Instability

A Dissertation
Presented to
The Academic Faculty
By
Fahad Ahmed

In Partial Fulfillment
of the Requirement for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
August, 2014

Copyright © 2014 by Fahad Ahmed

Invasive and Non-Invasive Detection of Bias Temperature Instability

Approved by:

Dr. Linda Milor, Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Sung Kyu Lim
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Abhijit Chatterjee
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Date Approved: April 21, 2014

ACKNOWLEDGEMENTS

I would like to thank my Advisor, Dr. Linda Milor, for her guidance and my family for their support.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
SUMMARY	ix
Chapter 1 Scientific Goals	1
Chapter 2 Overview of the Problem	5
2.1 BTI in Data Paths	5
2.2 BTI in Memories	8
2.3 Compiler Level Compensation for Reliability	10
Chapter 3 Wearout Monitoring and Mitigation	13
3.1 Ring Oscillator based Test Structure for Decoupling NBTI/PBTI	13
3.2 Data Path Ring Oscillator	18
3.3 Sleep Transistor Based Scheme	24
3.4 NBTI in SRAM	26
3.4.1 SRAM Stability under Statistical NBTI	26
3.4.2 Cell State for Testability	32
3.5 PBTI in the SRAM	37
3.5.1 SRAM Stability under Statistical PBTI	38
3.4.2 Cell State for Testability	41
3.5 Wearout Aware Compilation	45
3.5.1 Pre-Silicon Wearout Prediction	45
3.5.2 Wearout Aware Compilation	47
3.5.3 Case Study: SPARC V8	56
Chapter 4 Test Chip Implementation for BTI Monitoring	62
4.1 Top Level Overview	63
4.1.1 Memory Module Overview	65
4.1.2 Memory Module Test Features	70
4.1.3 Logic Module Overview	71
4.1.4 Logic Module Test Features	71
4.2 Test Chip Test Results and Discussion	72
4.2.1 Test Memory Macro Testing	72
4.2.2 Logic Block Testing	73
Chapter 5 Conclusions and Suggestions for Future Work	75
5.1 Summary of Research	75
5.2 Suggestions for Future Work	77

5.2.1 Wearout Mitigation at the OS Level	78
5.2.2 Off-Chip BTI Monitoring for Lifetime and Failure Analysis	78
REFERENCES	80

LIST OF TABLES

TABLE 1: SIGNAL VALUES FOR PBTI TM , NBTI TM AND SM(AC/DC).	16
TABLE 2: IMPROVEMENT IN DIFFERENT WEAROUT MECHANISMS AND INCREASE IN CODE SIZE FOR RELIABILITY-AWARE AND THERMAL-AWARE COMPILATION SCHEMES.	61

LIST OF FIGURES

FIGURE 1: A RING OSCILLATOR BASED TEST STRUCTURE.	7
FIGURE 2: (A) MEMORY ARRAY LEAKAGE IN THE ABSENCE OF A GATE-OXIDE BREAKDOWN EVENT. (B) MEMORY ARRAY LEAKAGE IN THE PRESENCE OF A GATE-OXIDE BREAKDOWN EVENT.	9
FIGURE 3: (A) RING OSCILLATOR IN STRESS MODE WITH BOTH THE PULL-UP AND PULL-DOWN NETWORKS IN STRESS MODE. (B) IN PBTI-TM WITH THE PULL-UP NETWORK SWITCHED. (C) IN NBTI-TM WITH THE PULL-DOWN NETWORK SWITCHED.	14
FIGURE 4: THE PROPOSED RING OSCILLATOR BASED NBTI/PBTI MONITOR.	16
FIGURE 5: FREQUENCY DEGRADATION (AND HENCE DELAY DEGRADATION) OF THE OSCILLATOR IN THE NBTI TM AS FUNCTION V_T OF DIFFERENT DEVICES IN THE STRUCTURE.	17
FIGURE 6: FREQUENCY DEGRADATION (AND HENCE DELAY DEGRADATION) OF THE OSCILLATOR IN THE PBTI TM AS FUNCTION V_T OF DIFFERENT DEVICES IN THE STRUCTURE.	18
FIGURE 7: DATA PATH RING OSCILLATOR.	19
FIGURE 8: THE PROPOSED SELF-TEST SCHEME.	20
FIGURE 9: TEST REGISTERS.	20
FIGURE 10: T-R AND DATA PATH CONFIGURATION OF THE DPRO DURING TM.	21
FIGURE 11: A MULTIPLE INPUT, MULTIPLE OUTPUT COMPLEX DATA PATH IN A PIPE-LINED STAGE.	23
FIGURE 12: FUNCTION INVERTING BLOCK.	23
FIGURE 13: DEGRADATION OF A (A) DECODER AND A (B) CLA WITH NBTI.	24
FIGURE 14: SLEEP TRANSISTOR BASED SCHEME.	24
FIGURE 15: (A) DECODER AND (B) CLA TEST WITH SLEEP TRANSISTOR BASED METHOD FOR NBTI.	25
FIGURE 16: 6T SRAM CELL.	28
FIGURE 17: CELL TRENDS WITH TIME.	32
FIGURE 18: CELL STATE FOR TESTABILITY.	33
FIGURE 19: VARIATION IN I_1 AND I_2 AS A FUNCTION OF THE THRESHOLD VOLTAGE OF THE NMOS AND PULL- UP PMOS DEVICES.	35
FIGURE 20: PART OF A MEMORY SYSTEM WITH THE PROPOSED CHANGES.	36
FIGURE 21: TEST RESULT FOR NBTI TEST FOR A MEMORY SYSTEM.	36
FIGURE 22: CELL TRENDS WITH TIME.	41
FIGURE 23: CELL STATE FOR TESTABILITY.	42
FIGURE 24: TEST CURRENT WITH INCREASING PBTI DEGRADATION.	43
FIGURE 25: PART OF A MEMORY SYSTEM WITH THE PROPOSED CHANGES.	44
FIGURE 26: TEST RESULT FOR NBTI TEST FOR A MEMORY SYSTEM.	44
FIGURE 27: GENERAL FRAMEWORK FOR ESTIMATING THE PRE-SILICON SoC WEAROUT PROFILE.	46
FIGURE 28: AN OVERVIEW OF THE PROPOSED COMPILATION FRAMEWORK.	49
FIGURE 29: ADAPTIVE STRESS DISTRIBUTION IN THE REGISTER FILE.	50

FIGURE 30: THE REGISTER ASSIGNMENT FLOW.....	51
FIGURE 31: THE PROPOSED VARIABLE MULTI-WINDOW CONTEXT SWITCHING SCHEME FLOW.....	54
FIGURE 32: NBTI FOR FFT.....	57
FIGURE 33: GATE-OXIDE DEGRADATION FOR FFT.....	58
FIGURE 34: NBTI DEGRADATION ACROSS DIFFERENT BENCHMARKS.....	58
FIGURE 35: PBTI DEGRADATION ACROSS DIFFERENT BENCHMARKS.....	59
FIGURE 36: GATE OXIDE DEGRADATION ACROSS DIFFERENT BENCHMARKS.....	59
FIGURE 37: TEST CHIP LAYOUT FOR BTI MONITORING.....	63
FIGURE 38: MEMORY MODULE OVERVIEW.....	65
FIGURE 39: DEFERENTIAL VOLTAGE SENSING SENSE-AMPLIFIER.....	66
FIGURE 40: DATA OUT PATH DURING READ.....	67
FIGURE 41: TEST CHIP FOR BTI MONITORING.....	67
FIGURE 42: BITCELL SELECTION PATHS.....	68
FIGURE 43: INPUT ADDRESS PATH.....	69
FIGURE 44: NAND GATE BASED DECODER.....	71
FIGURE 45: NBTI/PBTI TEST FOR BITCELL AND THE ACCESS TIME TEST.....	73
FIGURE 46: DECODER DELAY MONITORING USING DPRO.....	74
FIGURE 47: DEGRADATION MONITORING USING SLEEP TRANSISTOR BASED SCHEME.....	74
FIGURE 48: GROUND BOUND IS THE SUM OF DEVICE CURRENTS.....	78
FIGURE 49: EXPECTED VARIATION IN GROUND BOUNCE WITH DEVICE WEAKENING.....	79

SUMMARY

Invasive and non-invasive methods of BTI monitoring and wearout preemption have been proposed. We propose a novel, simple to use, test structure for NBTI /PBTI monitoring. The proposed structure has an AC and a DC stress mode. Although during stress mode, both PMOS and NMOS devices are stressed, the proposed structure isolates the PBTI and NBTI degradation during test mode. A methodology of converting any data-path into ring oscillator (DPRO) is also presented. To avoid the performance overhead of attaching monitoring circuitry to functional block, a non-invasive scheme for BTI monitoring is presented for sleep transistor based logic families. Since, BTI is a critical issue for memories, a scheme for BTI monitoring of 6T SRAM cell based memories is also presented. We make use of the concept of a DPRO and show how a memory system can be made to oscillate in test mode. The frequency of oscillation is a function of the devices in the cell. After validation of the proposed schemes using extensive simulations, we have also validated the results on silicon. We also introduce the concept of wearout mitigation at the compiler level. Using an example of a register file, we present a preemptive method of wearout mitigation using a compiler directed scheme.

CHAPTER 1

SCIENTIFIC GOALS

Device degradation in PMOS due to negative bias temperature instability (NBTI) and in NMOS due to positive bias temperature instability (PBTI) is emerging as a major reliability concern for the state-of-the-art VLSI systems. These degradation mechanisms degrade the maximum operating system frequency (F_{max}) as well as the minimum system operating voltage (V_{min}). Bias temperature instability (BTI), both positive and negative, is a gradual process and increases in severity with the passage of time. Hence, with the passage of time, data paths are expected to become slower and memories increasingly unstable.

The most common method employed to handle the F_{max} and V_{min} shift is the use of additional design margins or guard-bands, as they are commonly called [1]. The addition of extra design margins to mask component wearout is not straight forward. The extra design margins need to insure system functionality until the end of the chip lifetime. However, wearout mechanisms, such as BTI, are a complex function of the chip's PVT (process, voltage, temperature) conditions throughout its lifetime. Hence depending on the operating conditions, the amount of degradation varies significantly. The extra design margins usually come at the cost of a higher operating voltage, since a higher operating voltage means a higher F_{max} and lower V_{min} . But due to the large variation in the amount of degradation, the choice of the amount of voltage bump-up is not so simple.

Let's take the example of a VLSI system with time-zero maximum frequency specification F_{max_i} and time-zero operating minimum voltage specification V_{min_i} . Typically, the worst case degradation is used to calculate the amount of extra design margin. Hence, let's assume ΔV to be the voltage needed to compensate for the degradation in F_{max_i} and V_{min_i} at the end of life. ΔV is calculated assuming the chip operates constantly under the worst case PVT conditions throughout its lifetime. Although an unlikely scenario, it turns out to be the only reliable method given that the real operating conditions are unknown during chip design.

With the extra voltage margin ΔV , the time-zero operating frequency of the chip is F_{max_j} and the minimum operating voltage is V_{min_j} where $F_{max_j} > F_{max_i}$ and $V_{min_j} > V_{min_i}$. The chip is now sold with the frequency and voltage specification of F_{max_i} and V_{min_j} . Hence the chip is operated at a voltage higher than is needed at time-zero and run at frequencies lower than its maximum frequency at time-zero. This ensures that at the end of its life the chip will remain within the specifications. Hence although we have insured system functionality until the end of life, it comes at the cost of higher power due to the higher operating voltage and a lower performance specification. Since we use the worst case degradation for the extra margin, the design/performance cost can be very significant. As we move into the sub-1V VDD technology nodes, these margins may typically exceed 10% in voltage bump-up.

In addition to extra design margins, systems might also require the use of redundancy and error correction and detection for greater fault tolerance. Many applications, ranging from automatic flight control systems, nuclear power control systems, on-line transaction processors for financial institutions, to hospital patient

monitors, require systems to be extremely reliable. Fault-tolerant systems require a way to prevent a physical defect or failure from causing an error in system performance. Such systems incorporate fault detection, fault masking, fault isolation, and recovery procedures.

Fault-tolerant design at the circuit level focuses primarily on fault detection. The other aspects of fault tolerance (fault location, system reconfiguration, and recovery) are generally performed off-chip by the system architecture and software. The typical approach to on-chip fault detection involves the introduction of redundancy, in space, time and information, at the cost of either extra computational time and/or extra hardware components. The cost of such an approach is a degradation in yield (and a concomitant increase in manufacturing cost), together with higher power dissipation, since all modules are operated in parallel.

This work aims to demonstrate an alternative approach, involving *detection of the onset of failure through detection of component degradation over time*. If component degradation can be detected, the cost of building highly reliable systems will be reduced. The additional area needed for fault detection is limited to the design-for-test (DfT) circuitry, which is a small fraction of the component being monitored. Similarly, the cost of operating such a system is similarly reduced, due to the reduced power dissipation, since the DfT circuitry is only turned on intermittently during test.

There are a wide variety of faults that can cause system failures, ranging from static to transient faults. This work focuses on faults due to *component wearout*. The major causes of front end chip wearout have been gate-oxide breakdown, hot carrier injection, bias temperature instability (BTI), while at the backend, electromigration and

backend dielectric breakdown have been the major causes of concern in terms of reliability[1]. The focus of this work has primarily been BTI.

CHAPTER 2

OVERVIEW OF THE PROBLEM

Traditionally, the driving force behind VLSI trends has always been performance. The aim has been to get the most out of a small piece of silicon real estate with long term reliability being a secondary issue. However, the constantly shrinking device dimensions and increasing operating temperatures mean that system lifetime targets are increasingly difficult to meet. Since devices age faster for every new generation, wearout characterization, mitigation and compensation have all become some of the major challenges facing VLSI systems of today.

2.1 BTI in Data Paths

BIAS temperature instability (BTI) along with gate-dielectric breakdown is a source of concern for CMOS circuits [2]. BTI is generally attributed to the formation of charge defects at the Si/SiO₂ interface and in the oxide, known as interface trap charges and oxide trap charges respectively [3]. As a result, devices experience degradation in threshold voltage (V_t), drive currents, transconductance, mobility etc. [4], although BTI is generally associated with an increase in V_t . Although present in both PMOS and NMOS devices, BTI was only a major concern for PMOS devices (NBTI) with NMOS devices (PBTI) showing comparatively negligible degradation[4]. However, with the introduction of high-k metal-gate stacks for sub-45nm technology nodes, degradation in NMOS devices due to positive bias has increased, with large degradation observed for both types of devices [5].

The high operating frequencies of today's VLSI circuits has made off-chip automatic test equipment (ATE) based performance measurements highly inaccurate due to the parasitics introduced by the probe and the test equipment. Apart from the loss of accuracy, an off-chip measurement scheme also renders the idea of performance monitoring during normal device operation highly impractical. A more precise on-chip device monitoring scheme was needed for accurate failure predictions. Hence, the use of on-chip test structures steadily increased. Although playing no part in the chip functionality, these provide a relatively cheaper and efficient means of on-chip characterization.

BTI manifests as increase in delay in data paths. Ring oscillator based structures monitor the degradation in delay (or frequency) of an inverter chain. Use of on-chip ring oscillators to monitor device performance is not new [6],[7]-[8]. Figure 1 shows a simplified view of a ring oscillator based test structure. When the signal TE is low, the structure undergoes DC stress with each gate in a fixed state. When TE goes high, the structure starts functioning as a ring oscillator where the frequency of oscillation is given by

$$f = \frac{1}{2d_{inv}N} \quad (1)$$

where d_{inv} is the delay through a single inverter and N is the total number of inverters in the chain.

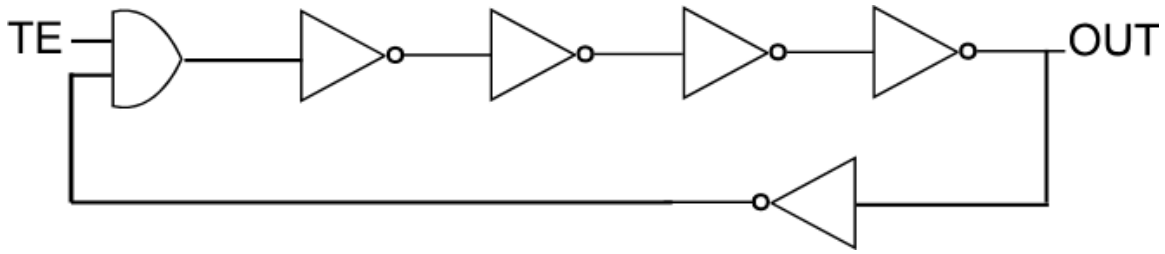


Figure 1: A ring oscillator based test structure.

During normal chip operation, the inverters in the test structure are stressed and undergo wearout similar to other functional units on the chip. It is assumed that the amount of degradation of the functional units can be approximated by the degradation in the test structures. To read out the degradation in the test structure, it is converted into a ring oscillator with the frequency of oscillation representing the device condition through equation 1.

Since the delay through the inverter chain is the sum of the individual delays of alternating PMOS and NMOS devices, it becomes impossible to isolate the delay degradation due to NMOS or PMOS degradation. Moreover, a typical test structure does not take into account the variation in degradation due to varying switching activity and state probabilities of nodes in an actual data path.

Ring oscillators have usually been embedded within the chip with the assumption that during normal operation, they go through the same stress and degradation as the rest of the circuit [6],[7]. Assuming a high correlation between the complex switching activity and stress levels of modern data paths and a simple inverter chain can be inaccurate. The switching activity levels and on chip local temperature may not match. However, monitoring the functional blocks directly is a challenging task and may come at the cost of degraded system performance. The idea of converting data paths into ring oscillators

and directly measuring the frequency was proposed by [9]. The use of actual data paths means that the thermal and electrical stress for lifetime measurements are no longer an approximation as data is read off from the functional block. However, the assumption that data paths can be simply divided into inverting and non-inverting ones is incorrect and limits the applicability of such a scheme.

2.2 BTI in Memories

NBTI results in degraded operating frequencies in data paths and degraded noise margins in memory cells, leading to reduced long-term reliability. NBTI aware design steps, from circuit level NBTI models [10] to NBTI augmented CAD tools [11], have become a necessity. These tools determine the reliability guard bands on the maximum operable frequencies of data paths and the noise margins for memory cells. This analysis has become a critical pre-silicon step [12]. These guard bands are in addition to the significantly larger guard bands for process variations. As a result, they limit performance/area optimization. Post-silicon tuning of circuits with adaptive control is fast emerging as a viable solution to NBTI [13]-[6]. Both, accurate modeling and estimation of the impact of NBTI on circuit performance and NBTI resilient circuit design require highly accurate NBTI monitoring schemes using embedded monitors.

Due to their potential relatively low switching activities, memory arrays are especially prone to NBTI degradation. An existing approach to monitoring of NBTI involves I_{ddq} -based sensing [14]. However, this approach involves sensing circuitry connected to the sleep transistor, which can create problems with the virtual grounds. More importantly, this scheme can only measure the accumulated leakage current of the whole array, containing cells with varying amounts of degradation and under the

assumption that other wearout mechanisms, such as gate-oxide breakdown (GOBD), are not present. An occurrence of a GOBD event can completely throw off the test results, as shown in Figure 2.

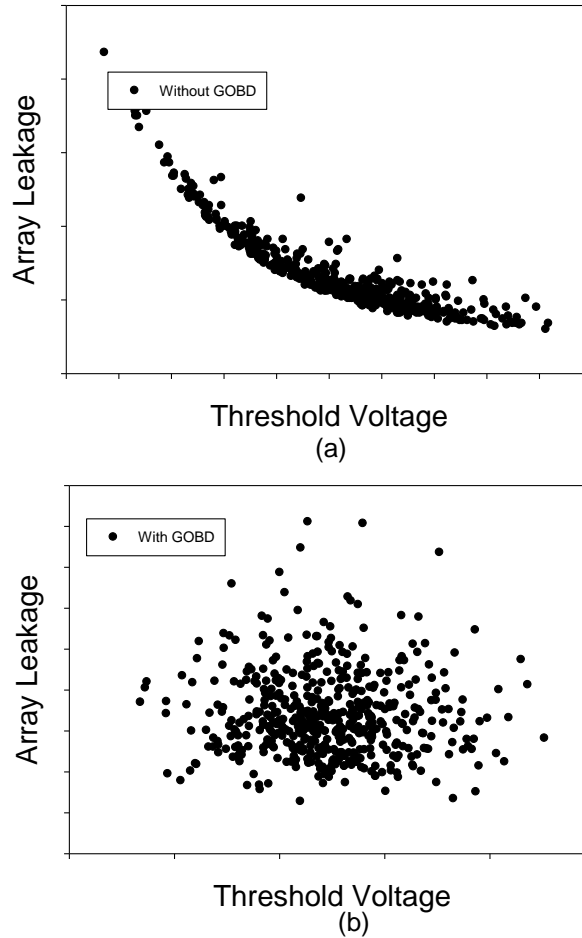


Figure 2: (a) Memory array leakage in the absence of a gate-oxide breakdown event. (b) Memory array leakage in the presence of a gate-oxide breakdown event.

As Figure 2(a) indicates, in the absence of a GOBD event, the I_{ddq} -based sensing scheme is indeed a good representation of the array NBTI degradation with a high correlation between the device threshold voltages and the accumulated array leakage. However, a GOBD event creates a possible low impedance path between the ground and V_{dd} of the array and becomes the dominant factor in determining the array leakage

leading to a significant reduction in the correlation between the array leakage device threshold voltages as seen in Figure 2(b).

Other approaches don't just measure the failure rate due to NBTI, but aim to reduce it. One approach relies on the recovery property of NBTI and introduces circuitry to periodically flip the data in an SRAM cell to reduce its degradation [8]. Another approach to reduce failure rates due to NBTI is the use of error correcting codes (ECCs) [9],[10]. ECCs store data with additional bits so that if there are single bit errors, these are detected and corrected [11]-[13]. However, as memory sizes increase and the probability of failing bits increase, it may still be necessary to reconfigure the SRAMs in the field through the use of redundancy. The appendix illustrates the potential improvement in SRAM yield through the use of redundancy and reconfiguration in the field, in addition of ECCs.

2.3 Compiler Level Compensation for Reliability

Post-silicon reliability is still heavily dependent on reactive measures, such as error detection and error correction. The use of on-chip sensors [15], complemented by post-silicon tuning [16], is also steadily increasing with the aim of extending system lifetime. However, since they do not mitigate the aging process itself, the effectiveness of such measures is always constrained due to the following reasons. Firstly, in the absence of redundancy, they do nothing in case of catastrophic events such as dielectric breakdown. Secondly, if the amount of aging can be reduced during the lifetime of a device, as proposed in this work, then any post-silicon tuning performed to compensate for wearout can also be regarded as *over design*, since the amount of tuning needed could

potentially have been reduced. After all, tuning of a parameter always comes at the cost of reduced system performance.

Modern compilers provide various optimization options for improving performance. However, the flexibility afforded by such compilers is rarely used towards reliability improvement. There has been some focus on compiler-directed thermal-aware register file assignments [17], since their relatively small area and high utilization make them one of the most likely candidates for overheating [18]. Since all wearout mechanisms have a strong dependence on temperature, this inherently leads to improvement in register file reliability. However, to the best of our knowledge, adaptive compilation schemes today are not truly wearout-aware with a primary focus on the extension of the register file's operable lifetime.

Recent work in reliability aware compilation has dealt with improving the thermal profile of the functional units under study. Some researchers have used an indirect approach by improving operating temperatures through improvement in power consumption [19]. Moreover, recent work [10] has shown that better thermal profiles can be achieved by considering global temperatures as the main optimization metric. However, an optimized thermal profile does not necessarily result in an optimized lifetime, since device wearout is not only a function of temperature, but also of stress.

Wearout aware scheduling techniques for stress distribution have been used for extending the lifetime of multi-core systems [20]. While this approach requires an additional central management unit which keeps track of and responds to the gradual system wearout, the use of an under-used computation unit as a system manager has also

been proposed [21]. An alternative to stress distribution is to gracefully drop failing components from operation, given available redundancy [22].

However, in this work we show that the *intra-component compiler-level* stress distribution, coupled with improved operating temperatures, can lead to an optimized component wearout profile. With improvement in component lifetimes as the main metric, we simultaneously reduce component temperatures and achieve a uniform stress distribution for optimal reliability.

CHAPTER 3

WEAROUT MONITORING AND MITIGATION

The aim of this work is to explore new methods of on-chip wearout monitoring. A few improvements in pre-existing methods are also described. We start off with methods aimed towards data paths and finally describe how these methods can also be applied to memories.

3.1 Ring Oscillator based Test Structure for Decoupling

NBTI/PBTI

In this section we propose a new ring oscillator based test structure for BTI monitoring. Although during stress mode (SM), both PMOS and NMOS devices are stressed, the proposed structure isolates the PBTI and NBTI degradation during test mode (TM). Additionally, the proposed structure has three stress modes. In DC stress mode (DSM), the devices are held at a constant DC stress, while in AC stress mode (ASM), the structure is converted into a ring oscillator with the devices undergoing AC stress.

Figure 3 shows the main working principle of our proposed structure. In SM, the pull-up and pull-down devices are stressed as in normal ring oscillators. However, to test for PBTI, the degraded PMOS is replaced by a ‘fresh’ PMOS device. Hence in PBTI TM, only the degradation in the NMOS device is monitored. Similarly in NBTI TM, the stressed NMOS device is replaced by a ‘fresh’ NMOS device. Hence the degradation detected during NBTI TM is only of the PMOS device. Therefore, although this scheme decouples the two dominant degradation mechanisms present in today’s technology nodes, it still retains the simplicity of a conventional ring oscillator.

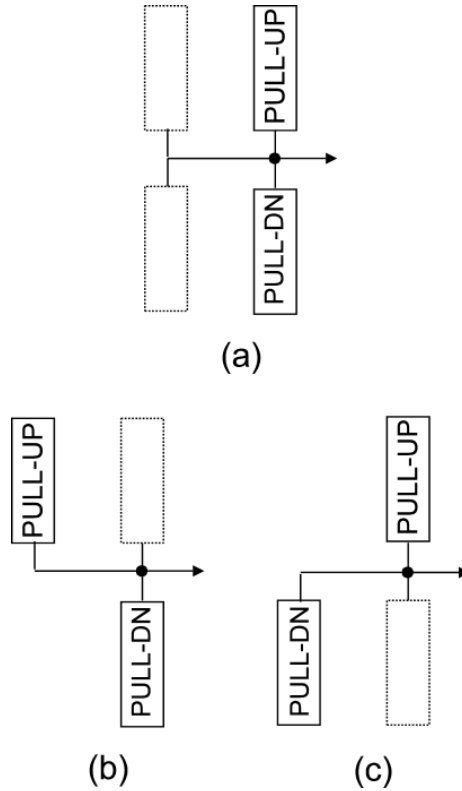


Figure 3: (a) Ring oscillator in stress mode with both the pull-up and pull-down networks in stress mode. (b) In PBTI-TM with the pull-up network switched. (c) In NBTI-TM with the pull-down network switched.

However, implementation of this scheme requires a greater number of transistors compared to conventional inverters. These extra devices are then used for switching between different stress and test modes. Figure 4 shows the implementation of the proposed scheme. Transistors P3 and P4 represent the dual pull-up network and N3 and N4 represent the dual pull-down network shown in Figure 3.

As shown in Figure 4, the signal ‘A’ toggles the structure between AC and DC stress modes. When ‘A’ is at logic 1, the system forms a feedback loop similar to a ring oscillator and starts oscillating. This corresponds to the AC SM. When ‘A’ is at logic ‘0’, the feedback path is broken and DC-IN becomes the input to the system, which corresponds to the DC SM. In this mode, alternating PMOS and NMOS devices in the

structure are held at DC logic levels, while the rest of the devices are in the no-stress mode. We can switch the devices under stress by toggling the signal DC-IN. Depending on the value of DC-IN, one set of devices is under stress for logic ‘1’ and the second set is under stress for logic ‘0’.

Table 1 summarizes the signal values for the two available TMs, NBTI TM and PBTI TM. While P3, P4, N3 and N4 form the dual pull-up and pull-down networks, P5, N5 and the transmission gates XX’ and YY’ form the switches that toggle between the two pull-up and pull-down networks.

As the structure enters PBTI TM, P5 is turned off disconnecting P4 from the output. The T-gate XX’ then connects the input to P3, which is relatively fresh, since it stays in the no-stress state during SM and is only turned on intermittently during PBTI test. Therefore the inverter for PBTI TM is formed by P3, N5 and N4. Similarly, in NBTI TM, N5 disconnects N4 from the output and the inverter is formed by P4, P5 and N3, where again N3 is a relatively fresh device enabling the degradation in N4 to be measured.

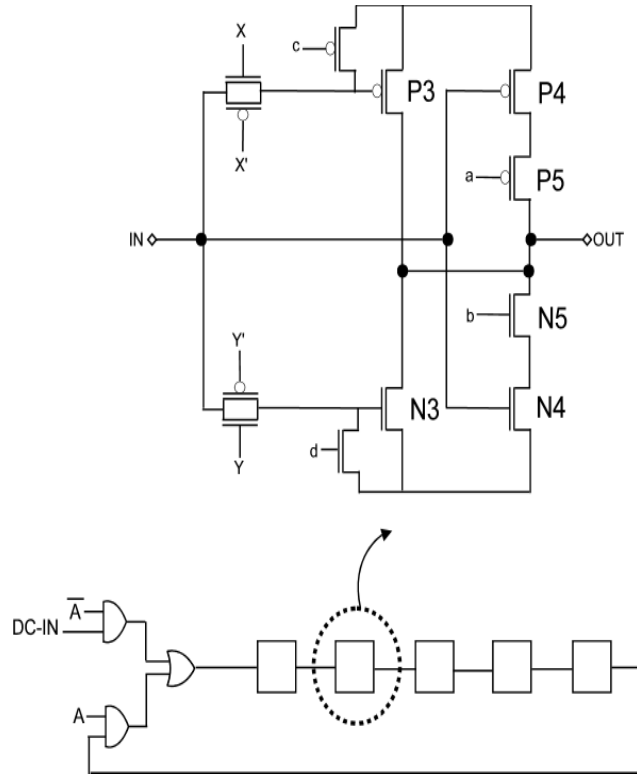


Figure 4: The proposed ring oscillator based NBTI/PBTI monitor.

Table 1: Signal values for PBTI TM, NBTI TM and SM(AC/DC).

	Test NBTI	Test PBTI	Stress Mode
a	0	1	0
b	0	1	1
c	0	1	0
d	0	1	1
X	0	1	0
Y	1	0	0

Unlike conventional ring oscillators, this structure has more than one PMOS/NMOS device undergoing V_t degradation due to BTI. Figures 5 and 6 illustrate the degradation in the oscillating frequency of the oscillator formed by the proposed structure as a function of V_t . Figure 5 considers NBTI TM and Figure 6 considers PBTI TM.

These results clearly indicate that the oscillation frequency in NBTI TM is independent of the NMOS devices, while in PBTI TM, it is unaffected by degradation in PMOS devices. During SM, P4 and N4 act as the devices under test.

Since the devices P5 and N5 are a part of the inverter even in the TMs, V_t degradation in these devices can affect the output in TM. Therefore the sizes on P5/N5 are larger than those of P4/N4 to minimize their effect on the system output during TM. As seen in Figures 5 and 6, this results in negligible sensitivity of the oscillation frequency to degradation in P5 and N5.

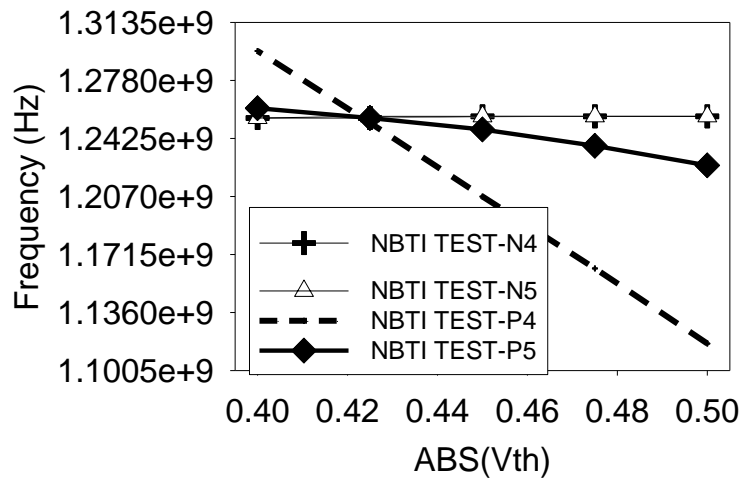


Figure 5: Frequency degradation (and hence delay degradation) of the oscillator in the NBTI TM as function V_t of different devices in the structure.

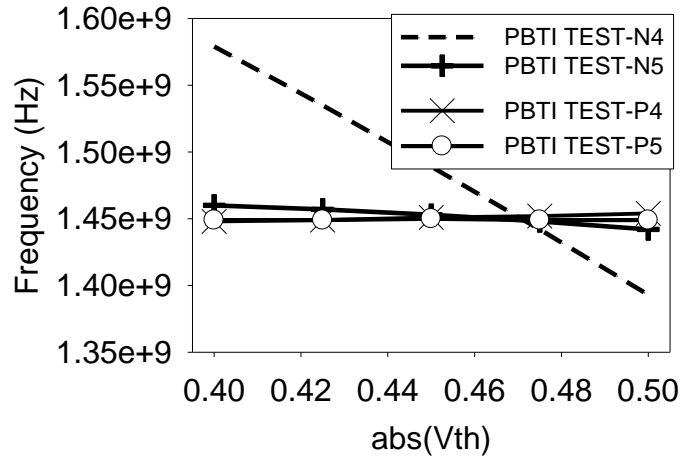


Figure 6: Frequency degradation (and hence delay degradation) of the oscillator in the PBTI TM as function V_t of different devices in the structure.

3.2 Data Path Ring Oscillator

The main idea behind the proposed scheme is illustrated in Figure 7. It is claimed that every data path in a chip can be turned into an oscillator, whose frequency provides a measurement of delay. A path is either an inverting data path, which starts oscillating if connected in feedback, or a non-inverting path, which requires an additional inverter to oscillate. To handle both types of paths, a set of data paths can be converted to ring oscillators by inserting muxes in the paths. The additional inverter inserted in non-inverting paths, together with the muxes, add delay, but since we are only interested in measuring delay degradation, determining the exact delay of the path is not important.

During normal operation the path is connected with the regular input and output data lines. In test mode (TM), the data path is disconnected from its data in/out and is connected in a feedback loop, which may or may not contain an inverter, to form a data path ring oscillator (DPRO), as shown in Figure 7. The frequency of oscillation can then be used to monitor the path degradation.

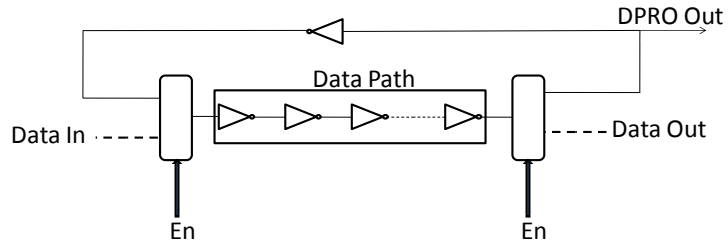


Figure 7: Data path ring oscillator.

However this implementation poses a few problems when we take into account the fact that the data paths are often a part of a pipelined stage with very critical delay margins. A single path cannot be taken into TM without disturbing the whole pipeline. Similarly, the performance overhead introduced by the introduction of the data selection blocks can also prove to be critical and may cause failures if present in critical data paths.

To address these issues, consider the self test scheme shown in Figure 8. The figure shows a normal pipeline stage with the path under test (PUT) highlighted. During TM, clock gating is used to halt the pipeline, ensuring that no erroneous data is fed out during TM. Two specially designed test registers (T-R) are inserted in place of conventional registers (Reg) for the data path being monitored. During TM, the test registers disconnect the PUT from the preceding and subsequent data paths and connect the feedback path.

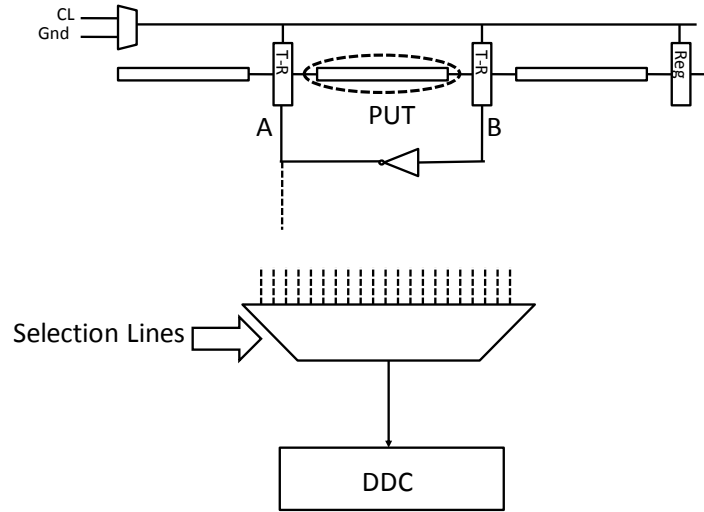


Figure 8: The proposed self-test scheme.

To keep the physical overhead to a minimum, another selection block can be introduced which selects between different monitored paths. The correct PUT is connected by the selection block to the delay/frequency detection circuit (DDC) for delay/frequency monitoring.

The T-Rs were designed to keep the performance overhead to a minimum during the normal operation of the data path. Since the register at the input of the data path has a slightly different function from the register at the output of the data path during TM, two separate registers were designed as shown in Figure 9. During the normal operation, the T-R selection blocks operate as conventional inverters. The register thus operates as a normal master-slave flip-flop with very little performance overhead.

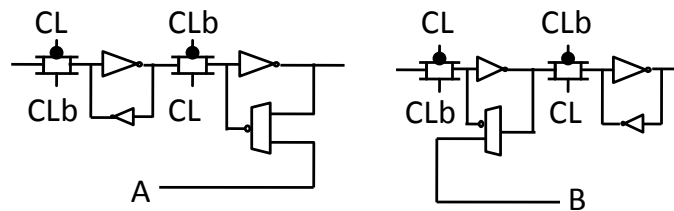


Figure 9: Test registers

The operation of the T-Rs during TM is shown in Figure 10. As the PUT enters TM, the pipeline clock signal (CL) is grounded, halting all operation in the pipeline. The master latch gets disconnected from the slave latch, and the T-gates at the input of the T-Rs are turned on. The slave latch of the input T-R and the master latch of the output T-R have data selection blocks which disconnect the latch configuration and connect the inputs to output lines A and B. The data path thus gets connected in the DPRO configuration and starts oscillating.

If the data path has a simple inverting or non-inverting relationship between its input and output, oscillation initiations can be started using the scheme illustrated in Figure 10.

However this becomes a problem for more complex data paths where it's not possible to formulate an inverting or non-inverting relationship between the inputs and the output. As an example consider a multiple input ($X_1, X_2, X_3, \dots, X_n$) and multiple

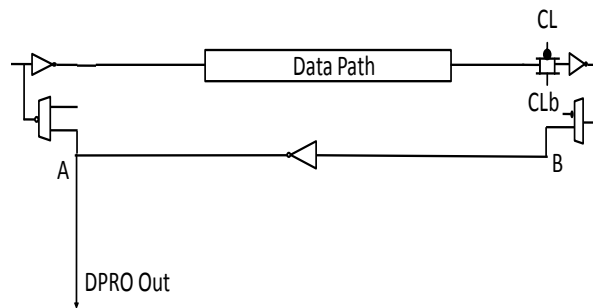


Figure 10: T-R and data path configuration of the DPRO during TM

output ($f_1, f_2, f_3, \dots, f_n$) data path. Let's assume that one of the output functions of this data path (f_x) is to be monitored for wearout.. The output f_x might be a function of multiple inputs and therefore could be '1' or '0' depending on the values at the inputs. Let set S_1 be the set of input values that gives a logic '1' at the output f_x and let S_0 be the set of inputs which give a logic '0' as follows:

$$f_x \{S_1\} = 1 \quad (2)$$

$$f_x \{S_0\} = 0$$

where $S_1, S_2 \in \{X_1, X_2, \dots, X_n\}$.

If we eliminate the inputs that have the same value in both the sets S_1 and S_0 , we're left with the inputs (X_R) that toggle the function f_x and that can make the data path oscillate.

Consider the complex data path in a pipelined stage, shown in Figure 11. Of the total number of outputs, suppose we want to monitor one as a selected wearout monitor. In TM, the T-R register at the output of f_x feeds the selected data path into the function inverting block.

The function inverting block senses the value of the output f_x and generates a set of inputs to be inverted or non-inverted and fed back into the register file at the input of the data path. For these inputs, the regular registers are again replaced by the T-R registers which feedback the output of the function inverting block into the data path, hence causing the value at the output of f_x to toggle.

Since the states of X_R would already be known for the states of function f_x , it is very simple to design the function inverting block. Figure 12 shows one possible implementation of the function inverting block. Depending on the value of f_x , the selection block selects the appropriate values independently of other selection blocks for inverting f_x . It should be noted here that this scheme of initiating oscillations in complex data paths leads to insignificant performance overhead due to the insertion of the specially designed T-R registers.

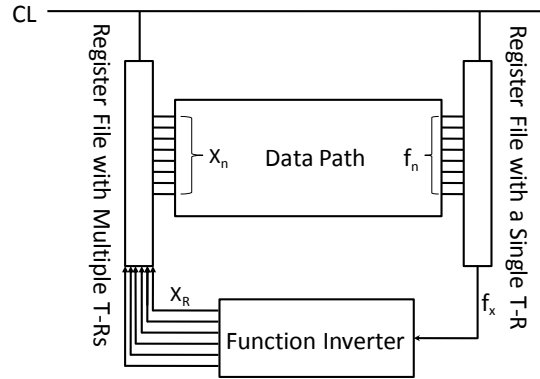


Figure 11: A multiple input, multiple output complex data path in a pipe-lined stage

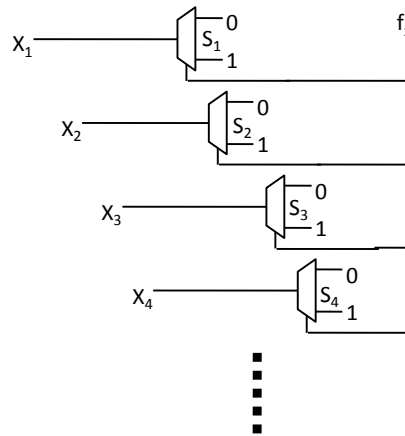


Figure 12: Function inverting block.

The proposed scheme was applied to a 1-out-of- 2^7 decoder and a carry look-ahead (CLA) adder designed in IBM 130nm technology. Out of the possible outputs for the decoder and the adder, one was selected for testing purposes. The structure of a decoder is such that a large number of feedback combinations can be implemented according to equation (2). The result of one of these is shown in Figure13.

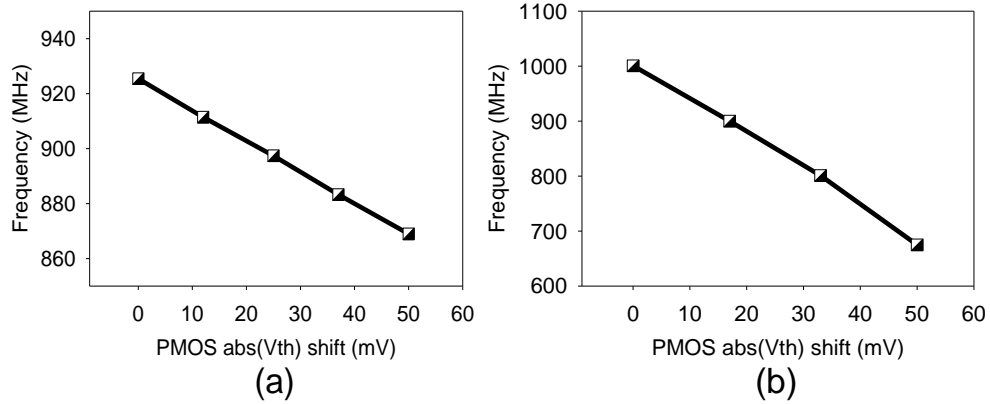


Figure 13: Degradation of a (a) decoder and a (b) CLA with NBTI

3.3 Sleep Transistor Based Scheme

The main issue with converting actual data paths into ring oscillators for wearout monitoring is the performance overhead. Such a scheme would add extra capacitive loads at the nodes connected to the feedback network. Controlling multiple inputs from the feedback path can also have a significant area overhead. In this section we propose a scheme for BTI monitoring which does not impact the delay of the path being monitored while reducing the layout complexities associated with the implementation of the DPRO based scheme. However this scheme is limited to sleep transistor based logic families such as MTCMOS[23].

The proposed scheme is summarized in Figure 14.

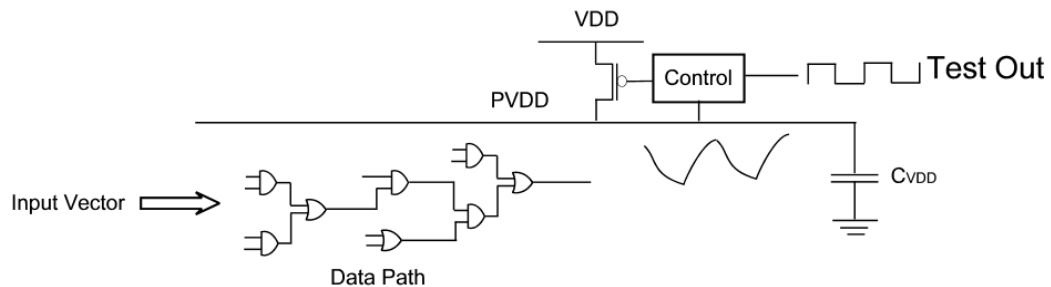


Figure 14: Sleep transistor based scheme.

During normal operation, the sleep transistor is always on with PVDD charged to VDD. However during test mode, the sleep transistor is controlled by the control block as shown in the figure. In test mode, initially PVDD is charged to VDD and the sleep transistor is turned off. The input vector is then fed in to the data path. The transistors in the data path start drawing in charge from the parasitic capacitance formed by the net PVDD (C_{vdd}). This causes PVDD to discharge as seen in Figure 14. The rate of discharge depends on the amount of current drawn by the data path. Once the voltage at PVDD falls below a certain threshold, the control block turns on the sleep transistor, charging PVDD back to VDD.

The frequency at which the control block turns the sleep transistor on and off depends on the rate of discharge of the PVDD which in turn depends on the strength of the devices attached to it. Hence, the signal ‘Test Out’ can be used to monitor BTI degradation in the data path. Since the nodes of the data path being monitored are not loaded by the test circuitry, this scheme does not degrade the performance of the data path.

The scheme was implemented on a decoder and a CLA similar to the one used for DPRO testing. The results are shown in Figure 15.

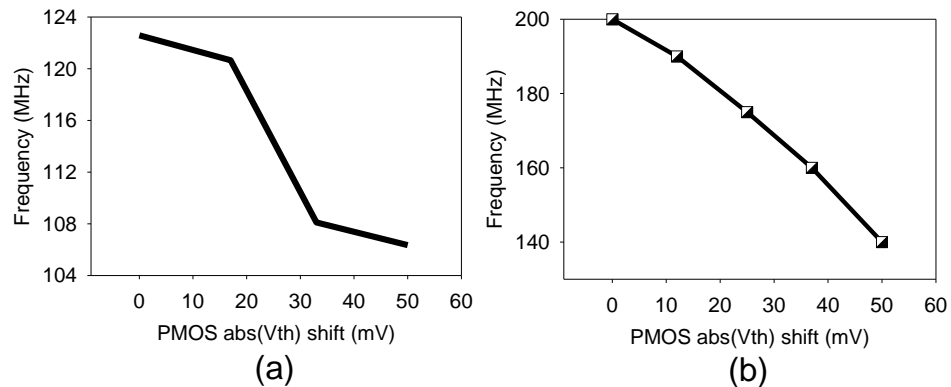


Figure 15: (a) Decoder and (b) CLA test with sleep transistor based method for NBTI

For the decoder under test, both schemes had low area overhead. However, the area overhead for the DPRO test would increase with the increase in the number of inputs fed back. The complexity of the layout may also increase fairly rapidly. Compared to that, the area overhead of the sleep transistor based scheme remains fairly constant irrespective of the complexity of the data path being monitored.

3.4 NBTI in SRAM

In this section we propose to take advantage of redundancy to reconfigure SRAM arrays in the field. To do this, it is necessary to detect degradation. This can be done by periodically testing the array to identify the failing bits, or through the use of ECCs, which identify failing bits. These approaches only detect failure after it has occurred. As an alternative, we propose an on-chip NBTI monitoring scheme with the ability to monitor the degradation of both of the PMOS devices in each individual cell of an SRAM array without affecting the performance of the cell during normal operation and with little physical overhead. Not only does the proposed scheme allow the detection of the NBTI induced cell failures, unlike conventional testing schemes, it also allows the prediction of failure times for error free cells.

3.4.1 SRAM Stability under Statistical NBTI

Most models for NBTI in the literature explain NBTI on the basis of electrochemical reactions. Models involve the activation energy, a forward rate constant (k_f) and a reverse rate constant (k_r) [10]. The total number of interface traps (N_{IT}), as a function of the forward and reverse rate constants and time, t , was derived in [3] and is

$$N_{IT}(t) = \sqrt{\frac{k_f N_0}{k_r}} (Dt)^n \quad (3)$$

where N_0 is the concentration of inversion carriers and D is a constant.

The interface traps result in an increase in charge in the capacitor formed by the channel and the gate. The increase in threshold voltage (V_{tp}) is the following:

$$\Delta V_{tp} = -\frac{qN_{IT}}{C_{ox}} \quad (4)$$

where C_{ox} is gate oxide capacitance per unit area and q is the electron charge. An additional correction factor (m) is introduced to accommodate the decrease in mobility:

$$\Delta V_{tp} = -\frac{(m+1)qN_{IT}}{C_{OX}}. \quad (5)$$

Plugging (1) into (3) gives a power law model for the increase in V_{tp} due to NBTI aging:

$$\Delta V_{tp} = -\alpha \frac{(m+1)q}{C_{ox}} \sqrt{\frac{k_f}{k_r}} N_0 D^n t^n. \quad (6)$$

Equation (6) has an additional correction factor, α , to take into account the affect of variations in frequency and the probability of the PMOS device being in the stress and/or the no-stress state [24].

Unlike data paths, which switch between logic ‘0’ and ‘1’ fairly frequently, internal nodes of memory cells store the same value for years. As a result, the PMOS devices in memory cells are under constant DC stress, which, unlike AC stress, does not allow the device any recovery period. Lifetime differences of up to 4X were reported between devices undergoing NBTI degradation due to AC and DC stress [25]. The long time periods of fixed states in SRAM cells lead to an increasingly ‘skewed’ cell with the

PMOS of one inverter in the cell being largely unaffected, while the other degrades. This leads to severely impaired noise immunity in these cells and unreliable memory operation. Hence NBTI has become a critical reliability concern for cache arrays.

Since SRAM cells are also the smallest for their technology nodes [26], they are the most affected by process variations. One source of variation is the threshold voltage, whose standard deviation is a function of device area (WL):

$$\sigma(V_{th}) = \frac{K}{\sqrt{WL}} \quad (7)$$

where W is the device width, L is the device length, and K is a technology dependent parameter. As will be shown later, within-die variation in threshold voltage leads to cells that become especially susceptible to failures due to PMOS degradation. Hence, to completely understand the impact of NBTI on the reliability of a cache array, it is imperative to consider process variations. A typical 6T SRAM cell, shown in Figure 16, was used for all of our simulations. The initial values of V_{th} for all transistors in the cell were taken from a Gaussian distribution ($V_{t0}(m, \sigma)$), with mean, m , and standard deviation, σ .

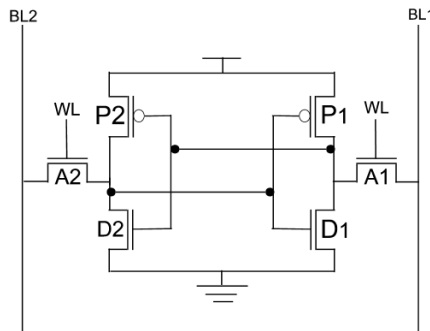


Figure 16: 6T SRAM cell.

Assuming no degradation, V_{th} of the NMOS devices in the SRAM cell was modeled with the time-invariant Gaussian distribution:

$$V_{tD1,2}, V_{tA1,2} = V_{t0}(m, \sigma) \quad (8)$$

with statistics extracted from a 65nm industrial process. The threshold voltages of the two PMOS (P1 and P2) devices in the SRAM cell undergo aging. Our models for the PMOS threshold voltages consist of three parts and are time-dependent, as follows:

$$V_{tp1} = V_{t0}(m, \sigma) + \Delta V_{t\text{deg}1}(t) + V_{tdv}(m_d, \sigma_d) \quad (9)$$

and

$$V_{tp2} = V_{t0}(m, \sigma) + \Delta V_{t\text{deg}2}(t) + V_{tdv}(m_d, \sigma_d). \quad (10)$$

In (9) and (10), the device threshold degradation due to NBTI at time t , $\Delta V_{t\text{deg}1}(t)$ and $\Delta V_{t\text{deg}2}(t)$, is modeled using (6). $V_{tdv}(m_d, \sigma_d)$ corresponds to the random variation in the amount of degradation due to NBTI for identical transistors operating under the same conditions. So far, there is no thorough understanding of the fundamental reasons for random variation, with recent experimental results [11] negating the RDF (random dopant fluctuation) based models. Therefore, the distributions of time zero V_{th} variations and variations in NBTI degradation under identical conditions are assumed to be uncorrelated. The statistics of $V_{tdv}(m_d, \sigma_d)$, estimated in [11], were used in this study.

An SRAM cell has two “complementary” PMOS devices. When one is undergoing degradation due to negative stress, the other is in a no-stress state. Then, because the devices are complementary, as the degradation in one PMOS device increases, the degradation in the other remains constant. Therefore, the amount of degradation of one device is a complementary function of the amount of degradation of the other. Let $MAX(\Delta V_{t\text{deg}}(t))$ be V_{th} degradation due to DC stress at time t , where

$$MAX(\Delta V_{t\text{deg}}(t)) = -\frac{(m+1)q}{C_{ox}} \sqrt{\frac{k_f}{k_r}} N_0 D^n t^n. \quad (11)$$

Then,

$$\Delta V_{t\text{deg}1}(t) + \Delta V_{t\text{deg}2}(t) = \text{MAX}(\Delta V_{t\text{deg}}(t)). \quad (12)$$

SRAMs are characterized with several performance metrics. These include the read and retention static noise margins (SNMs), write margin, access time, and the Vdd-min. The static noise margins are defined as the minimum DC noise voltage necessary to change the state of an SRAM cell. The read SNM is measured with the access transistors turned on, while the access transistors are off for the retention SNM. The write margin is the minimum voltage needed to flip the state of the cell, with the access transistors turned on. Vdd-min is the minimum voltage to which the SRAM retains its state. Finally, access time is the time to perform a read operation. More recently, SRAMs are also characterized by their dynamic noise margins, for read, write, and hold [20]-[24].

It has previously been shown that read and retention static noise margins (SNM) degrade with increasing NBTI degradation, while the writability of cells improves [27]. Using the device threshold voltage models described in the previous sections, simulations were done to investigate the effect of NBTI on SRAM performances. The state probability for each cell was generated using a uniform distribution. Therefore, some cells experience less switching activity and hence increased degradation, while others undergo frequent recovery cycles. The corresponding V_{th} degradation and recovery cycles were then calculated using (5) and (6).

Monte Carlo sampling was used to simulate the effect of statistical NBTI on SRAM read stability, minimum retention voltage, write margin and access time with increasing NBTI degradation at the worst case PVT corner. All the SNMs were calculated by fitting squares between the voltage transfer curves (VTC) for the inverters

in the SRAM latch and by observing the diagonal length of the smaller of the two squares [28].

All the resulting performance plots were normalized against the ideal specification i.e.

$$Y = \frac{X - X_{spec}}{X_{fault-free} - X_{spec}} \quad (13)$$

where X is the performance parameter under study, $X_{fault-free}$ is the same parameter assuming no breakdown and X_{spec} is the minimum design specification to be met by the parameter X for fault free operation.

Figure 17 plots the Monte Carlo simulation results for the trends of the read stability margin, minimum retention voltage, write margin and access time against time. Variation in each parameter, due to σ process variation, is indicated with the curves. As can be seen from the figure, process variations can cause large variation in the initial value of the cell's performance parameters. Many cells have high initial performance margins, while a large number of cells start out with lower stability. In addition, some cells may also fail to meet the required performance criteria for error-free operation at time zero, and hence should be replaced by redundant cells using memory reconfiguration.

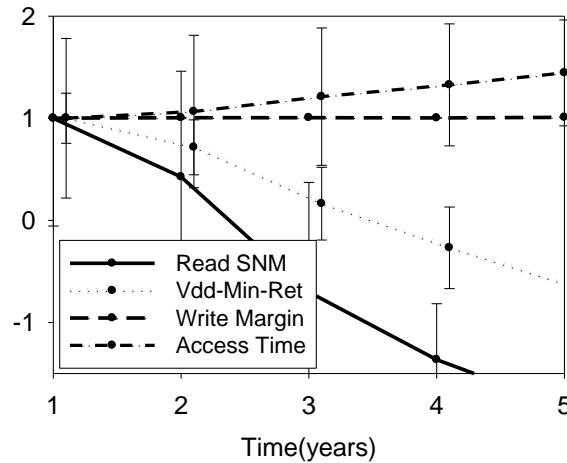


Figure 17: Cell trends with time.

As our results indicate, read stability had the most significant effect due to NBTI while the minimum retention voltage was also severely affected. Both write margin, and access time were relatively unaffected by NBTI with access time showing a slight improvement with the weakening of the PMOS devices.

Perhaps, the most significant result from the trends shown in Figure 3 is the expected failure time of the cells due to NBTI. Under worst case operating conditions, cells may start failing at around the 2 year mark, due to instable reads, which is well short of the required system lifetime.

3.4.2 Cell State for Testability

The design of an SRAM cell makes it difficult to detect PMOS weakening during normal device operation. The pull-up PMOS devices are the weakest of the three types of devices in the cell.

Consider a write operation. A write operation involves discharging the node storing ‘1’ to a voltage well below the tripping point of the inverter. For a short interval

of time, the current on the bit line is dominated by the PMOS current. But as the node discharges, the feed-forward effect of the latch turns off the PMOS and turns on the NMOS, causing the rest of the discharge through the NMOS device. The detection of PMOS drive current in such a short time period is challenging.

The read operation, on the other hand, does not involve the PMOS current, since a read involves a discharge of the bit-line through the access and the pull-down device.

Our proposed state for an SRAM cell for testability under NBTI degradation is shown in Figure 18. Writability of a cell insures that the output nodes of the two inverters forming the latch at the centre of the cell can be brought close to zero, while keeping their inputs also at zero. To force a cell into the test state, both bit-lines are forced to ground, as shown in Figure 18. This causes the cell to enter a meta-stable state where the inputs and the outputs of the inverters are forced to a '0'. This is a destructive operation, since the state of the cell cannot be predicted once normal operation resumes.

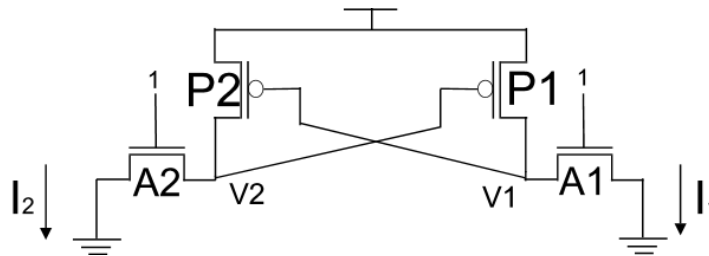


Figure 18: Cell state for testability

Level 1 BSIM models were then used to derive analytical expressions for the change in currents I_1 and I_2 as a function of changes in V_{th} of the pull-up and the access devices in the test state, where the pull-up transistors (P1 and P2) were assumed to be in velocity saturation, while the access transistors (A1 and A2) were assumed to be in the linear region. The sensitivities of the discharge currents to the threshold voltages of the devices are the following:

$$\frac{dI_{1,2}}{dV_{tp1,2}} = -S_p \beta_p (1 + \lambda V_{DD} - \lambda V_{1,2}) \quad (14)$$

and

$$\frac{dI_{1,2}}{dV_{tA1,2}} = -S_A \gamma_A V_{1,2}. \quad (15)$$

where “ S ” is the transistor size, and “ β ,” “ γ ” and “ λ ” are constants. As mentioned earlier, writability of the cell insures that the access devices are stronger than the pull-up PMOS devices. Hence, the node voltages “ $V_{1,2}$ ” can be approximated as $V_{1,2} \approx 0$, leading to further simplifications as follows:

$$\frac{dI_{1,2}}{dV_{tp1,2}} \approx -S_p \beta_p (1 + \lambda V_{DD}) \quad (16)$$

and

$$\frac{dI_{1,2}}{dV_{tA1,2}} \approx 0. \quad (17)$$

Figure 19 plots the simulated SPICE results for the change in cell output currents I_1 and I_2 with varying NMOS access and pull-up PMOS device threshold voltages for the cell state shown in [16], in the presence of process variation. Simulation results confirm our initial assumption that the output current in the proposed test state for an SRAM cell is relatively insensitive to variations in the NMOS devices in the cell, but is a strong indicator of the current state of the PMOS devices in the cell.

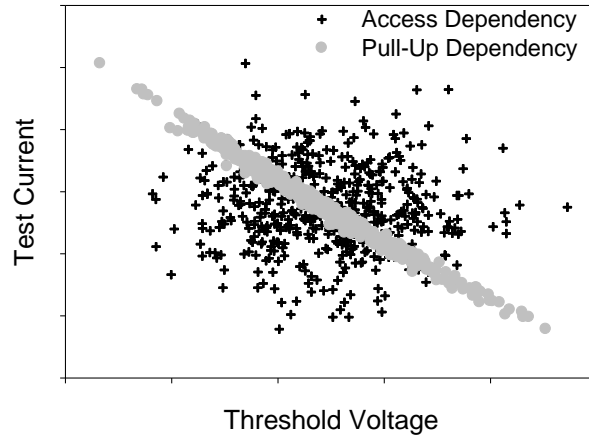


Figure 19: Variation in I_1 and I_2 as a function of the threshold voltage of the NMOS and pull-up PMOS devices.

We can now apply the proposed methodology of the DPRO from section 3.2. The cell test state consists of two steps i.e., grounding of the bit-lines and the consequent charging of the bit-lines by the cell PMOS devices. If we can toggle the system between these two states, then the system would start oscillating. The test currents can then be read out as the oscillating frequency of the system.

Figure 20 shows a part of a memory system with the proposed changes. During normal operation, the test blocks do not interfere with the system operation. Once the system enters test mode, the pre-charge circuitry is deactivated. The data lines are grounded and the write enable (WE) signal is now controlled by the DFT block. Initially the DFT block enables WE, forcing the bit-lines to '0' and the cell enters the test state. Once the bit-lines are grounded, the DFT block disables WE, floating the bit-lines. The cell starts sourcing test current which starts charging up the bit-lines. Once the bit-line voltage crosses a pre-defined threshold, the DFT block activates WE, forcing the bit-lines to '0' again. Hence, the system starts oscillating. The frequency of oscillation is function of the cell PMOS strength and hence is an indicator of NBTI.

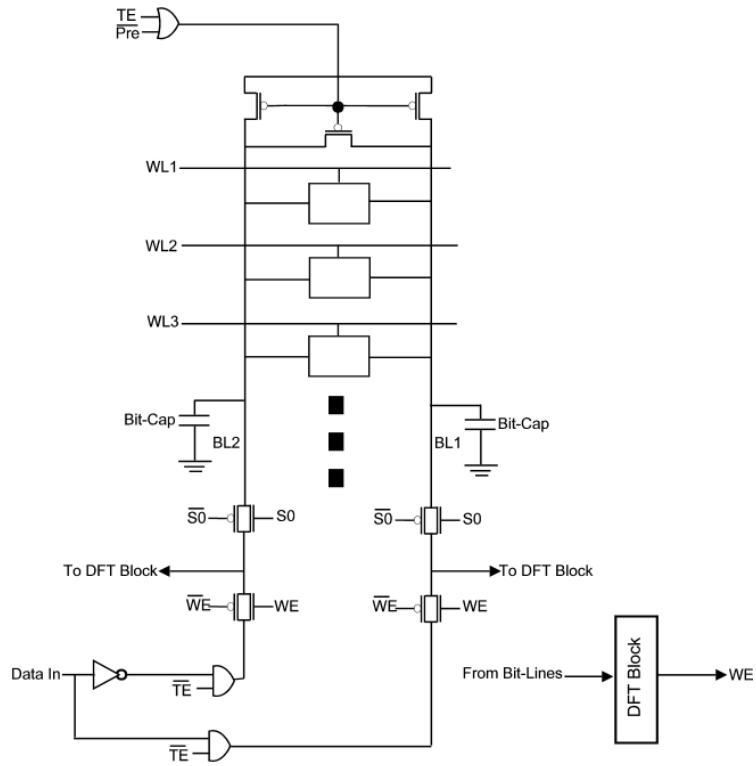


Figure 20: Part of a memory system with the proposed changes.

A memory array with 128x128 cells was simulated with the proposed NBTI monitoring scheme. The result is shown in Figure 21.

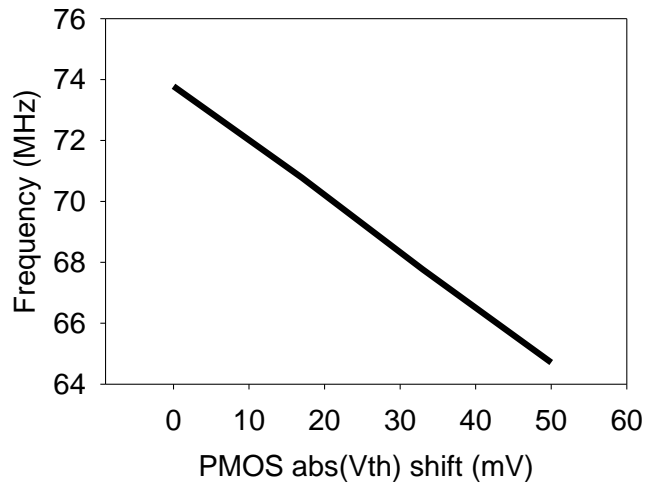


Figure 21: Test result for NBTI test for a memory system.

3.5 PBTI in the SRAM

While NBTI is the degradation of a PMOS device under negative stress, PBTI on the other hand is the degradation of an NMOS device under positive bias stress. PBTI, like NBTI, results in shifts in device parameters, such as threshold voltage, transconductance, device mobility, etc., but can also be identified by shifts in the threshold voltage.

Historically, BTI was only a major concern for PMOS devices with NMOS devices showing comparatively negligible degradation. However with the introduction of high-k metal gate stacks for sub-45nm technology nodes, degradation in NMOS devices due to positive bias has increased with large degradation observed for both types of devices [4]. This means that both PMOS and NMOS devices show significant degrading trends for state of the art technologies. For data paths this results in increasing delays as both NMOS and PMOS devices degrade.

The NBTI monitoring scheme presented for the memories only monitors the strength of the PMOS devices. Since PBTI is equally critical in high-k metal-gate technologies, we now look into methods for monitoring NMOS degradation in 6T cell based memories.

Since the read current of a cell is effectively the currents of the pull-down and the access NMOS devices, PBTI monitoring is similar to monitoring the read current of the cell. The read current would degrade if either of the two NMOS devices degraded. However, the scheme has to be independent of the cell PMOS device strength

For a complex memory cell, device sizing and hence their strengths, for both the PMOS and the NMOS, is extremely critical to correct cell operation. As shown in the analysis for NBTI in the earlier section, degrading PMOS devices mainly degrades read

SNM. However with both the NMOS and PMOS devices degrading, the situation is far more complex. Various scenarios can arise from all read, retention and write margins degrading to all three improving with time. If any of the access devices undergo larger degradation than the PMOS connected to them, the cell might have write issues. Another possibility is for the NMOS pull down to degrade more than the NMOS access devices which might lead to read SNM violations. Of course, the converse of these scenarios is also possible with the margins improving with time.

3.5.1 SRAM Stability under Statistical PBTI

Based on the well-known reaction-diffusion (RD) theory, PBTI is directly related to the formation of interface traps and their subsequent diffusion into the dielectric. The corresponding increase in threshold voltage (V_{tp}) follows a power law model [5][10].

$$\Delta V_{tp} = A_0 \exp(-E_a / kT) V^\alpha t^n \quad (1)$$

where E_a is the activation energy, k is Boltzmann's constant, T is temperature, V is the voltage stress, and t is time. A_0 , α , and n are fitting parameters and were obtained from experimental results. Upon the release of the voltage stress, there is some recovery of the device degradation. The fraction of the recoverable component (r) is modeled as shown in equation (2):

$$r = \left(1 + \beta \left(\frac{t_{relax}}{t_{stress}} \right)^\phi \right)^{-1} \quad (2)$$

where β is a scaling parameter, t_{relax} is the total recovery time, t_{stress} is the total stress time and ϕ is a dispersion parameter [24]. Since r is the fraction of the total possible recovery remaining, equation (2) can now be used to divide the total degradation during

stress from (2) into a recoverable component and a permanent component [10] with the recoverable component given by:

$$\Delta V_{t_rec} = r\Delta V_p \quad (3)$$

The recovery was also modeled according to the RD framework. When the stress is released, a recovery phase is initiated and the amount of threshold voltage shift decreases according to:

$$\Delta V_{tr} = \Delta V_{t_rec} \frac{\zeta_1 + \zeta_2 \exp(-E_a/kT)t_r}{\zeta_3 t} \quad (4)$$

where $\zeta_{1,2,3}$ are constants which depend on the oxide thickness and the back diffusion rate of hydrogen and temperature, t is the total time, t_r is the recovery time, and V_{t_rec} is the maximum possible recovery[10] calculated from equation (3). Equations (1) and (4) are adequate for capturing the temperature and voltage dependency of the stress and recovery phase of degradation due to PBTI.

In summary, the device degradation during periods of stress is estimated using equation (1) while the recovery is estimated using equation (4) enabling accurate cycle-by-cycle tracking of device degradation during system operation.

The degradation of threshold voltage results in longer delays at the circuit level, which eventually results in failure of circuit performances. For any circuit component, a threshold can be determined such that shifts in the threshold voltage results in circuit-level failure, as was demonstrated in[15].

For our current analysis we look at PBTI in isolation while ignoring the effects of NBTI for now. This greatly reduces the problem complexity and helps in understanding the effect of PBTI on cell margins.

Using the device threshold voltage models described in equation 4, simulations were done to investigate the effect of PBTI on SRAM performances. The state

probability for each cell was generated using a uniform distribution. Therefore, some cells experience less switching activity and hence increased degradation, while others undergo frequent recovery cycles. The corresponding V_{th} degradation and recovery cycles were then calculated using (4).

Monte Carlo sampling was used to simulate the effect of statistical PBTI on SRAM read stability, minimum retention voltage, write margin and access time with increasing PBTI degradation at the worst case PVT corner. All the SNMs were calculated by fitting squares between the voltage transfer curves (VTC) for the inverters in the SRAM latch and by observing the diagonal length of the smaller of the two squares [28].

All the resulting performance plots were normalized against the ideal specification i.e.

$$Y = \frac{X - X_{spec}}{X_{fault-free} - X_{spec}} \quad (13)$$

where X is the performance parameter under study, $X_{fault-free}$ is the same parameter assuming no breakdown and X_{spec} is the minimum design specification to be met by the parameter X for fault free operation.

Figure 22 plots the Monte Carlo simulation results for the trends of the read stability margin, minimum retention voltage and write margin against time. Variation in each parameter, due to σ process variation, is indicated with the curves. As can be seen from the figure, process variations can cause large variation in the initial value of the cell's performance parameters. Many cells have high initial performance margins, while a large number of cells start out with lower stability. In addition, some cells may also fail to meet the required performance criteria for error-free operation at time zero, and hence should be replaced by redundant cells using memory reconfiguration.

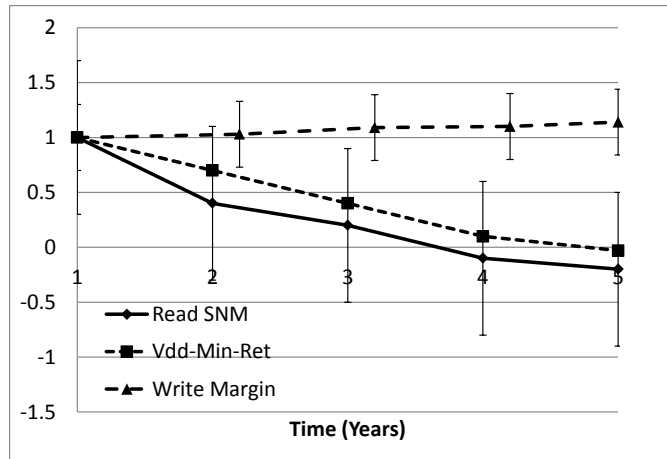


Figure 22: Cell trends with time.

As our results indicate, read stability had the most significant effect due to PBTI while the minimum retention voltage was also severely affected. Write margin was relatively unaffected by PBTI with write margin showing slight improvement with the weakening of the NMOS devices.

3.4.2 Cell State for Testability

Our proposed state for an SRAM cell for testability under PBTI degradation is shown in Figure 23. Of the 6 transistors of a 6-T bitcell, 4 are NMOS devices and play a far more significant role in cell operation than the PMOS devices. In-fact, the PMOS device can even be replaced by passive resistors all the while maintaining full functionality of the bitcell. We detect the degradation of the NMOS device due to PBTI through a pseudo-read operation. The read operation consists of turning on the access devices while pre-charging and floating the bit-lines connected to the access devices. The

bitline on the side of the cell storing a '0' discharges through the NMOS pull-down device. Hence, read current is just the current through a series connection of two NMOS devices. If we can detect the strength of the read current, we can estimate the degradation in the NMOS devices because of PBTI.

The test cycle of the PBTI test starts off as a normal read operation. First, during the precharge cycle, the bitlines are driven to logic '1'. However, unlike a precharge cycle, the word line is on, turning on the access transistors A1 and A2. Then the bit-lines are allowed to float, and the degraded bit-line read voltage is converted to current.

As the NMOS devices weaken, they choke the read current through the series access and pull-down current path formed by the devices Ax and Dx. The read current is a strong function of the access devices as well as the NMOS pull-down devices. Degradation in any of these would cause the read current to degrade.

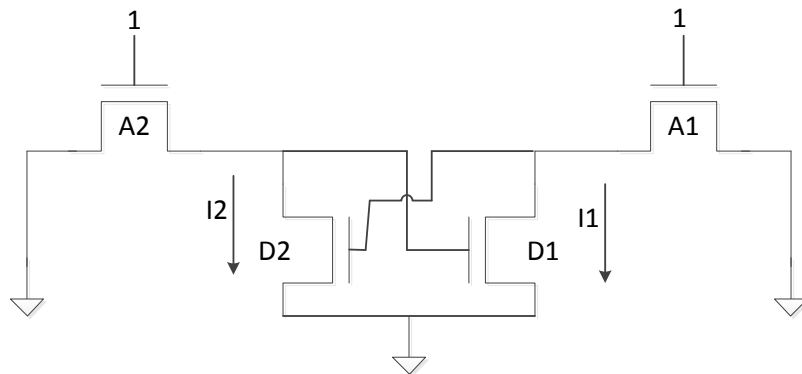


Figure 23: Cell state for testability

Although the degradation can come from both, this current is an indicator of the NMOS device strengths A1 and A2 as shown in Figure 24.

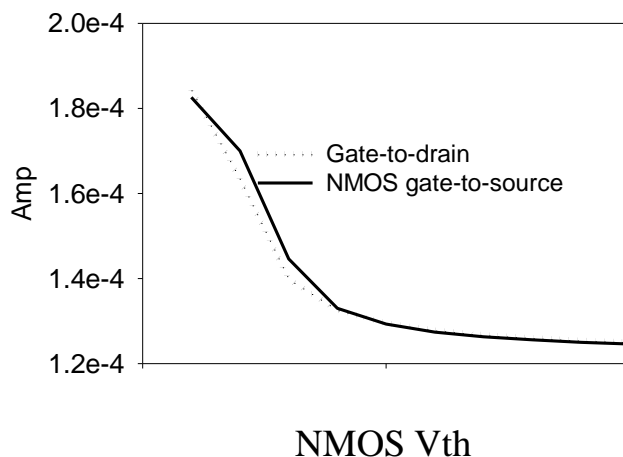


Figure 24: Test current with increasing PBTI degradation

We can now apply the proposed methodology of DPRO from section 3.2. The cell test state consists of two steps i.e., pre-charging of the bit-lines and the consequent discharging of the bit-lines by the cell NMOS devices. If we can toggle the system between these two states, then system would start oscillating. The test currents can then be read out as the oscillating frequency of the system.

Figure 25 shows a part of a memory system with the proposed changes. During normal operation, the test blocks do not interfere with the system operation. Once the system enters test mode, the pre-charge circuitry is deactivated. The data lines are pre-charged to VDD. Initially the DFT block enables PRE, forcing the bit-lines to VDD and the cell enters the test state. Once the bit-lines are charged to VDD, the DFT block disables PRE, floating the bit-lines. The cell starts sinking test current which starts discharging the bit-lines. Once the bit-line voltage crosses a pre-defined threshold, the DFT block activates PRE, forcing the bit-lines to VDD again. Hence, the system starts oscillating. The frequency of oscillation is function of the cell PMOS strength and hence an indicator of NBTI.

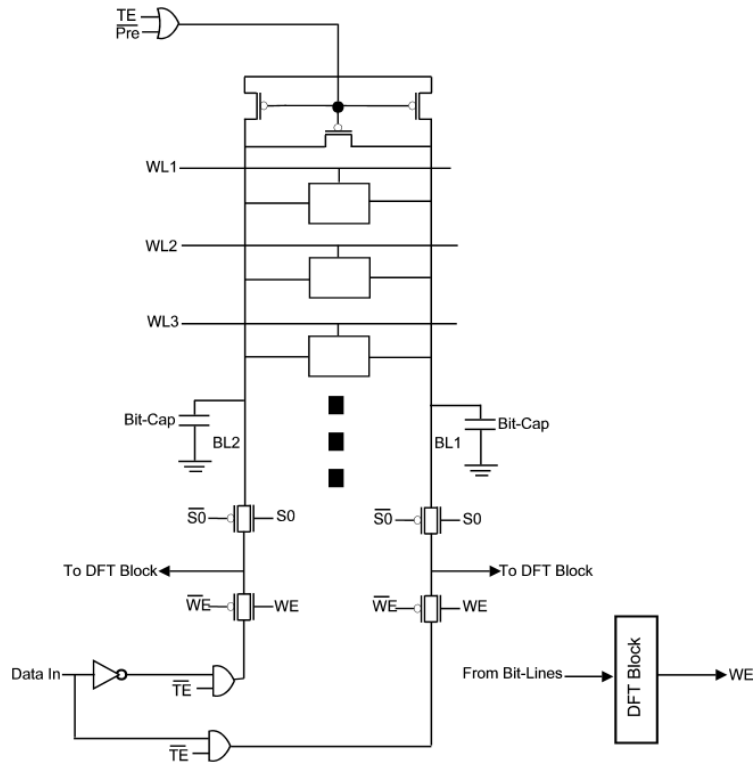


Figure 25: Part of a memory system with the proposed changes.

A memory array with 128x128 cells was simulated with the proposed NBTI monitoring scheme. The result is shown in Figure 26.

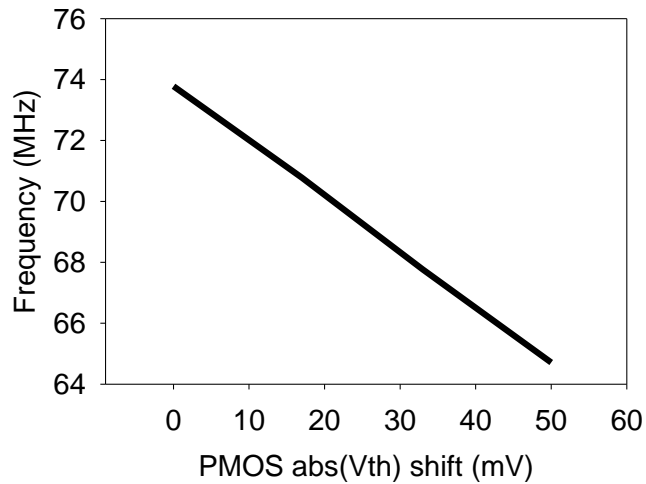


Figure 26: Test result for NBTI test for a memory system.

3.5 Wearout Aware Compilation

The work presented here proposes a wearout-aware compiler directed register assignment scheme that distributes the electrical stress throughout the register file, at compilation time, with the aim of minimizing wearout. The proposed compilation flow consists of two phases of optimization. During phase 1 of the flow, crude wearout estimates are used to perform reliability optimization during the compilation process, while phase 2, which is more time intensive, performs the optimization using detailed wearout models. To support register-window based architectures, this is done at two levels i.e. at the register window level using an adaptive register window assignment, called variable multi-window context switching (VMWCS), and at the register level using an adaptive intra-window register assignment scheme, called adaptive variable assignment (AVA). Moreover, as temperature is so critical for device reliability, the proposed wearout-aware register assignment also reduces the temperature of the registers or the register windows.

3.5.1 Pre-Silicon Wearout Prediction

Figure 22 summarizes the framework employed for estimating the pre-silicon wearout profile of the register file. The HW/SW emulation platform, similar to the one presented in [29], was used to extract the required thermal profile and block level switching activity for a chip running benchmark software. The activity at the input of the block under study (the register file in this case) was converted to stress probabilities and transition probabilities at the inputs. These probabilities were then propagated within the block, depending on its logic behavior, for a complete stress/transition probability profile

of the internal nodes of the block under study, as shown in Figure 22. Thus we have the probability of a transition occurring at any node, which can then be translated into the flow of current through the interconnects connected to that node. Similarly, the probability of stress at a node within the block is translated into the probability of voltage stress for the transistors connected to that node. Then, using the thermal profile from the HW/SW emulation platform and the calculated probability of current flow and voltage stress, we can use device level models to characterize any wearout mechanism in the block under study to determine the wearout profile of the system.

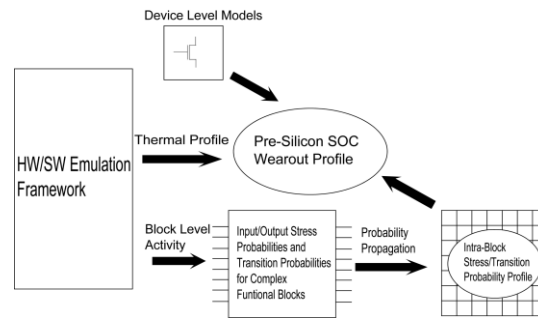


Figure 27: General framework for estimating the pre-silicon SoC wearout profile.

Finally, we rely on established device level models for the modeling of different wearout mechanisms [10],[30],[31],[32],[33]. Due to the lack of higher level models for the progressive effect of these mechanisms, it was necessary to first model their effect at the device level and then abstract the models to the micro-architecture level. For this work we have focused on three critical wearout mechanisms, namely NBTI, PBTI and transistor gate oxide breakdown (GOBD).

Gate oxide breakdown is modeled as a leakage (low resistance) path through the oxide. Several frameworks describing the physics behind the phenomenon of dielectric degradation and breakdown have been proposed. The thermo-chemical model, however,

provides a deterministic description of the process of oxide trap generation. It has been shown that the application of electric field on molecular dipole moments lowers the activation energy for bond breakage [30]. This in turn leads to enhanced trap formation in the dielectric. It has been shown [30] that trap generation depends on the electric field and is described by an Eyring equation:

$$N_t = \kappa \exp\left(\frac{(E_a + \kappa_2 E_0)}{kT}\right) t^n, \quad (25)$$

where κ is the total density of traps available for generation, E_a is the activation energy, k is the Boltzmann's constant, κ_2 is a device constant dependent on the field acceleration factor, and E_0 is the oxide electric field. Traps form throughout the oxide, and when the number of traps exceeds a threshold, a leakage path forms through the oxide.

The formation of a leakage path does not necessarily result in circuit-level failure. However, a threshold can be found when the number of traps links directly to circuit-level failure, as was demonstrated in [15].

3.5.2 Wearout Aware Compilation

With the high-level system wearout monitoring framework described in the previous section, we can measure and compare the effects of different compilation techniques on the wearout of a register file. Before we describe the actual algorithms that were employed for wearout mitigation, we need to characterize and understand the functioning of a register file and its wearout over time.

With the models from equations 6 and 25, the first step in determining the wearout profile of the register file is its division into sectors with similar wearout behavior. Sectors are similar if they undergo the same operations, i.e. a read operation, a write operation, or neither with the current data being retained. Hence, when a register is

being read, the transistors undergoing positive/negative voltage bias stress are identified. A similar process is repeated for the write operation. When neither a read or a write occurs, the latches are identified as being under constant DC stress, resulting in worst-case BTI and GOBD damage.

Typically, any wearout mechanism ‘ X ’ can be defined as

$$X = f(s, t, T) \quad (26)$$

where ‘ s ’ is the stress profile in terms of the number of reads and/or writes, ‘ t ’ is the timing information associated with the profile ‘ s ’ and the set T is the thermal profile. As an example, consider a register that undergoes two writes, and one read at times t_{wr1} , t_{wr2} and t_{rd} during a total run-time of t_r while the temperature variation in the register during t_r is represented by T_r . Then for equation (26), $s = \{writes=2, reads=1\}$, $t = \{t_{wr1}, t_{wr2}, t_{rd}\}$ and $T = T_r$.

The aim of the wearout mitigating algorithms proposed is a compiler-directed spreading of device wearout at runtime. However, at compile-time, we lack the necessary information to derive a complete wearout profile for the wearout mechanisms. Note that in a typical compilation framework, the complete stress profile of the register file is available only after the completion of the compilation process, while the thermal and timing information is available at runtime. As a result it becomes impossible to accurately predict the device wearout at the compilation phase. To overcome this issue, we propose the compilation framework with two phases shown in Figure 23, which are run sequentially.

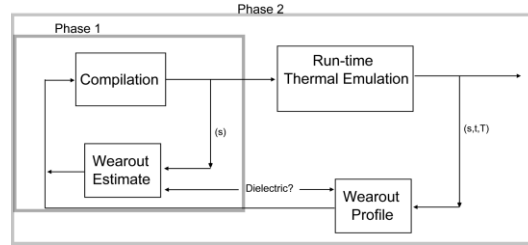


Figure 28: An overview of the proposed compilation framework

Phase 1 consists of a reliability optimization scheme consisting of a compilation module and a wearout estimating module, while phase 2 also takes into account the run-time information and the thermal profile for a more complete wearout profile.

The wearout estimating module of phase 1 estimates the degradation due to the wearout mechanisms under study using the stress profile from the compilation module. A constant time window is assigned to each variable and a constant temperature is assumed for each register. The estimate of the total wearout under the current compilation run is then fed into the compilation module which then reassigns the registers depending in the wearout estimate from the last compilation run. Even though a constant temperature is assumed for the wearout models at this point in time, hot spots are taken into account during compilation and avoided by keeping apart registers with high activity levels. In phase 2, the stress timing information and thermal profile of the register file is included in the framework and the detailed wearout profile is used to verify if the required wearout distribution has been achieved or if further optimization is required.

To accommodate register window based architectures, two levels of wearout mitigation, within windows and within registers inside each window, are supported in the proposed compilation framework, as shown in Figure 29.

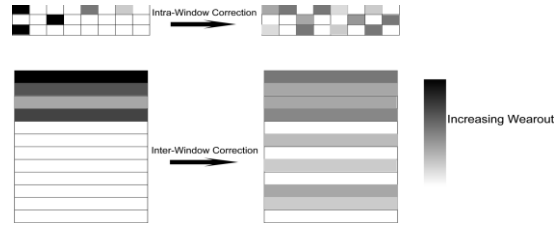


Figure 29: Adaptive stress distribution in the register file

Although this work targets a register-window-based architecture, our first approach for wearout reduction can be just as effective for any other architecture, since the aim here is the adaptive assignment of the active registers, irrespective of whether they are in a window or the whole register file is accessed at once.

Wearout awareness was introduced in the compiler in such a way that the original optimizations employed in the compiler itself are not compromised in any way, while still leaving room for reliability optimization. Typically, for every new variable assignment to registers, the compiler accesses the color graph and finds a set of registers that are available for assignment [34]. The flow is then transferred to a series of filters, as shown in Figure 30. These filters introduce further reductions in the set of available registers, according to some pre-determined optimization criteria, resulting in what is known as the reduced availability set (S_r). Once the reduced availability set is formed, typically a compiler picks the first register from the list, making no distinction between the registers in this set. Hence, in the case of a reduced availability set larger than one, there is still room for the introduction of improved register assignment for reliability.

The proposed reliability filtering process can be introduced before or after the reduction of the set to S_r depending on how reliability critical the target application is. Introducing the reliability filter before reduction of the availability set gives the highest precedence to reliability optimization.

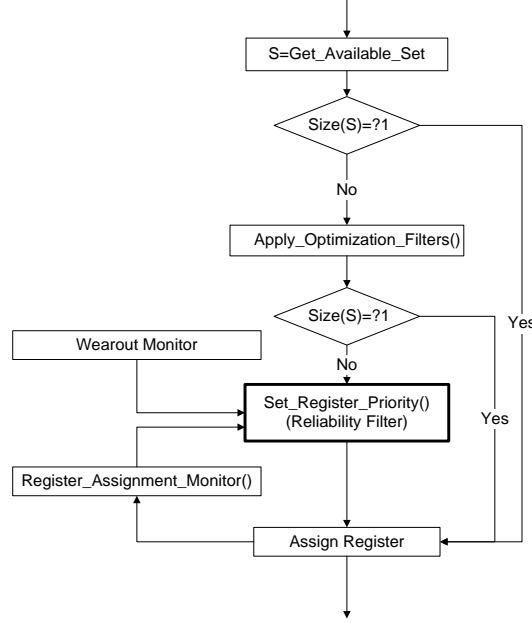


Figure 30: The register assignment flow

As can be seen, without extensive changes in the compiler flow, we have inserted a new reliability filter in the register assignment flow. The reliability filter assigns a gain value to all the registers in the reduced set according to the input from the register wearout monitor, which keeps track of the register wearout from the last run. The gain δn of a register at position R_n is the difference between its computed wearout W_n and the wearout of the register at position 1. W_n was computed using a weighted sum of the parameters associated with the wearout mechanisms under study, i.e.

$$\delta n = W_n - W_1 \quad (7)$$

$$W_n = \Gamma_{NBTI} V_{iPMOS}(s, t, T) + \Gamma_{PBTI} V_{iNmos}(s, t, T) + \Gamma_{GOBD} N_i(s, t, T)$$

where Γ_x is the weight assigned to wearout due to mechanism X. Hence, a higher weight for NBTI means a higher precedence is given for NBTI reduction during compilation.

Variable assignments for each register are also tracked in the register assignment monitor block shown in Figure 30. The algorithm starts with a set of registers R for variable assignment. R is the reduced availability set. In assigning registers to variables,

precedence is given to registers that are yet assigned any variables and whose wearout results indicate a lack of activity from the last run. This forces the spreading of activity across the register file. For the case when no such register exists, i.e. has previously not been assigned variables, the wearout of the first register in the set of available registers is compared with the wearout of all the other registers and the gain for each register is computed. The first register in the set is then replaced if the gain of the current register is greater than a predetermined value Δ . However, a replacement is only executed if it does not result in a *hot spot*. A register is labeled as a hot spot if it or any of its surrounding registers exceeds a pre-determined number of variable assignments.

In summary, the algorithm passes through the register set of length n once, and replaces the first register if it satisfies the conditions mentioned above. At the end of the pass, we have the best candidate for register assignment in position 1 in the set. The algorithm has a time complexity of $O(n)$. Once the variable is assigned to a register, the register assignment monitor and the hot spot labels are updated and the process is repeated for the next variable.

The new AVA is limited by the fact that it can only monitor and select registers, for variable assignment, from the set of registers visible to the compiler at that instant. This would be of no consequence if the compiler had access to the whole register file for every register/variable assignment. This, however, is not the case for register window based architectures, such as SPARC, where the register file is divided into smaller sets. Hence, when assigning a variable, the compiler can only access a single window.

Therefore, we propose a variable multi-window context switching (VMWCS) scheme which balances the switching activity among the register windows in the register

file. In general, a program includes a number of functions (or procedures) that are called in a hierarchal manner, where the *main* function is at the top level of the hierarchy and uses register-window $i=1$ in execution. All the functions found at the same level of the hierarchy use the same register-window, which is sequentially next to the one used by the ancestor functions.

Such methods of assigning register windows are inefficient in terms of reliability and thermal behavior, since they lead to highly uneven activity levels among the windows, i.e. the levels in the hierarchy are not equally weighted in terms of switching activity. This leads to an uneven distribution of electrical stress throughout the register file and may result in high temperatures due to the close proximity of highly-accessed register windows. Thus VMWCS balances the inter-window stress profile by studying the structure of the program, by exploring the functional interdependencies, and by assigning various register windows to functions at the same hierarchal level at compiler preprocessing time.

Figure 31 shows the flow chart of VMWCS. The proposed scheme executes in three main stages, (1) information extraction, (2) window setting, and (3) window assignment.

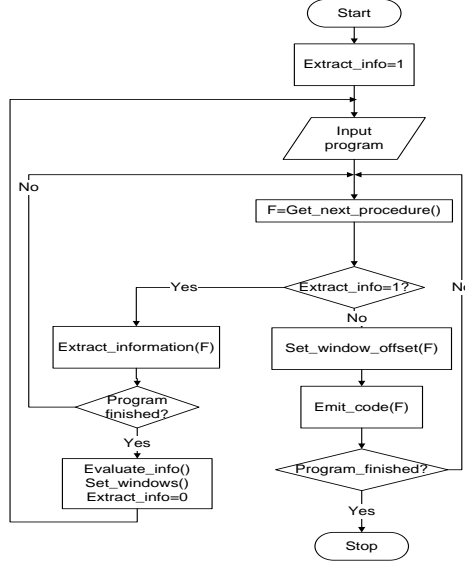


Figure 31: The proposed variable multi-window context switching scheme flow.

The information extraction phase is performed initially by setting a flag, named *Extract_info*, to 1. At this stage, all the necessary information needed to construct a hierarchal tree of the program (functional calls and access weight for each procedure) is extracted and exported to files to be used later (left branch in Figure 31). Then, in the windows setting stage, the exported data is used for the generation of the hierarchal tree and the subsequent initialization of a factor, called weight, for each function (F_p) which is calculated using:

$$weight(F_p) = Access(F_p) \times \prod_{i=2}^P repeat(F_i) \quad (8)$$

where *Access* is the extracted access weight of the current function, which is the calculated degradation factor based on the function utilization of a register window. The factor *repeat* is an estimate of the number of calls to the function F_i . We get an estimate of the number of function calls from the number of iterations in which this function call is embedded and the number of explicit calls to this function. It is important to note that we extract these values at compile time. Although runtime profiling of these functions is an

adequate methodology [35], our approach is more time efficient, and the estimated number of calls at compile time is close to the actual number of calls at runtime. Once calculated, this weight factor is then used to select the window for each function, with the aim of uniformly distributing the wearout across the register file.

Based on the impact of each function on register window wearout, a number k , the window step size, is assigned to each function. k represents the step size relative to the window of the immediate ancestor of the function F_p (right branch in Figure 31). Thus, the output from this stage is a list of all the functions and their assigned window step sizes.

The list generated by the window setting stage is then passed onto the window assignment stage, which generates the instructions required to shift a function from one window to another. Therefore, depending on the window step sizes, the window assignment stage then lays out the functions across the register file in such a way as to minimize wearout during runtime (right branch in Figure 26).

VMWCS has a computational complexity which is related to two main factors, the number of available windows (N_w) and the number of functions or procedures in the compiled program (N_f). The algorithm starts with computing the average degradation of the register file (Deg_w) if the function degradation impacts are equally distributed between the windows. From this value, the algorithm passes through each window and allocates a number of functions from the available set, such that the overall degradation impact is close to Deg_w . The computational complexity of VMWCS is $O(N_w \cdot N_f)$.

VMWCS introduces a few additional instructions and some compile time overhead. The instruction overhead is translated into an increase in code size and execution time. However, the overhead has a negligible impact, as will be shown later.

3.5.3 Case Study: SPARC V8

The SPARC V8 is a RISC based 32-bit architecture with as many as 128 general purpose registers. This window based architecture was selected as a case study for this work. The windowing nature of this architecture allows us to evaluate the effectiveness of both the AVA and the VMWCS wearout mitigation schemes.

At any point, only 32 registers are visible to the software. Out of these, eight are global, while the rest of the 24 form the register window. This register window moves up and down the register stack with every function call or return. Each window has eight local registers, with 16 being shared with the neighboring windows, with the aim of passing data between function calls.

The viability of the proposed wearout mitigation scheme was assessed across a variety of benchmarks using the HW/SW emulation platform presented in [29]. While the AVA was used for wearout balancing inside the active window, VMWCS performed the inter-window wearout balancing. NBTI, PBTI and gate-oxide degradation were monitored for each benchmark, and the results for the proposed reliability aware compilation scheme, presented in this work, were compared to the default compilation scheme [34] and the thermal aware compilation scheme presented in [17]. The comparison with [4] enables us to determine the impact of stress on lifetime, separately from temperature.

Let's first consider a detailed analysis of the effect of the proposed scheme on NBTI and GOBD for one of the benchmarks, FFT, which uses a limited number of windows. Analysis for PBTI was left out since PBTI is similar in behavior to NBTI. Figure 32 presents the normalized difference between the threshold voltage degradation of PMOS devices in the register file for the default compilation scheme [34] and the proposed reliability aware compilation scheme, i.e.,

$$Z = \frac{(Vt_default) - (Vt_reliability)}{(Vt_default)} \quad (9)$$

where Z represents the average wearout of the devices in a single register after the completion of a single run of the benchmark. A positive value indicates a improvement as compared to the default scheme. Figure 32 indicates that some registers show clear improvement in mitigating NBTI by as much as 10% during runtime. However, this comes at the cost of an increase in the degradation of other registers. The difference between the magnitude of improvement at some locations and the magnitude of the increase in wearout in others is due to the improvement in the thermal profile of the register file at run-time and the recoverable nature of NBTI.

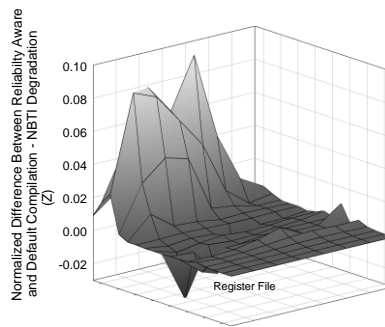


Figure 32: NBTI for FFT

Compared to BTI, gate-oxide degradation is not recoverable, and hence there is a direct tradeoff between reducing the wearout in one region of the register file vs. another,

as seen in Figure 33. However, overall, the register file showed improvement in gate oxide degradation during runtime.

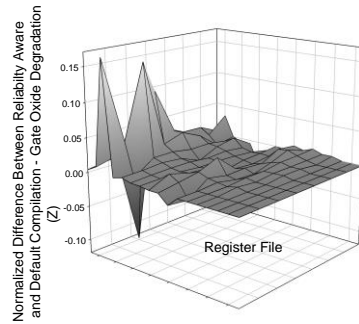


Figure 33: Gate-oxide degradation for FFT

The compilation process keeps track of wearout of registers, on top of thermal-awareness, making our scheme a truly reliability aware scheme. Figures 34-36 plot the amount of degradation due the proposed reliability-aware and thermal-aware schemes normalized against the default compilation scheme. Hence, for example, a value of 0.8 on these figures indicates a 20% improvement for a specific reliability mechanism over the default compilation scheme. Our results indicate that we were able to achieve significant improvement over the thermal-aware compilation method across different benchmarks.

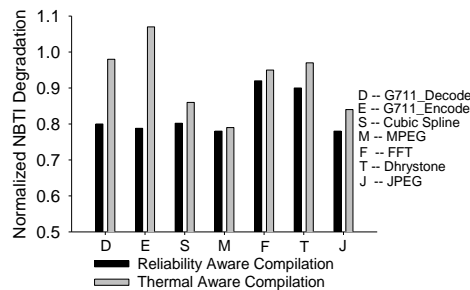


Figure 34: NBTI degradation across different benchmarks

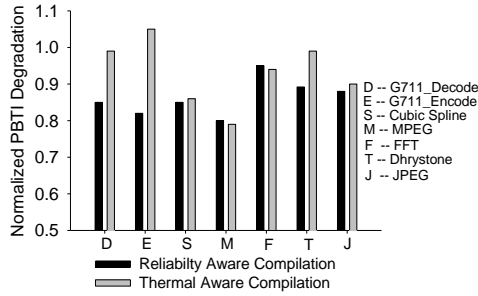


Figure 35: PBTI degradation across different benchmarks

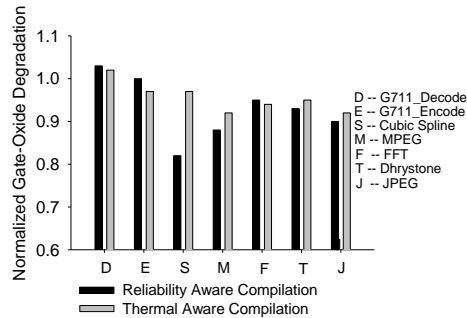


Figure 36: Gate oxide degradation across different benchmarks

Figure 34 indicates the increase in PMOS threshold voltages due to NBTI normalized against the increase in PMOS threshold voltages due to the default compilation scheme i.e. $\frac{Vt_reliability/thermal}{Vt_default}$. On average, NBTI was significantly reduced for the reliability-aware scheme with around 15-20% improvement across the benchmarks. Compared to that, thermal-aware compilation [17] did not show any clear trend, with the MPEG showing significant improvement, while Encode experiencing worsened NBTI. The significant improvement in the NBTI degradation for the reliability aware scheme is due to the healing nature of NBTI.

Stress balancing across the register file helps in distributing the workload throughout the register file, hence reducing the number of completely inactive registers, which result in worst case NBTI degradation for the latches. The fact that more devices in the register file experience periods of no-stress helps in the recovery of some of the

degraded threshold voltages for these devices. The thermal-aware scheme failed to show any significant improvement for benchmarks where register assignment had little impact on the thermal profile [17] (Decode and Encode), while the comparatively better results for the reliability-aware scheme for the two can be solely attributed to stress balancing.

Similarly, Figure 35 shows the results for NMOS degradation due to PBTI across different benchmarks. Due to their similar nature, PBTI followed a similar trend to PMOS degradation due to NBTI. Our results indicate slightly better results for NBTI as compared to PBTI. However, it should be noted that the relative magnitudes of NBTI and PBTI degradation and their recovery is a strong function of the choice of dielectric.

Gate oxide degradation, with its non-healing nature, also showed encouraging results across the benchmarks. However the improvement in gate oxide degradation was the least among the three mechanisms considered. Gate oxide degradation is also the hardest to predict, since not only does the amount of degradation depend on the activity intensity, but also on the type of activity, with a cell being thrice toggled having a possibility of both, higher degradation and lower degradation, in comparison with a cell that is toggled twice, depending on the arrangements of the toggles with respect to time.

Table 2 summarizes our results across different benchmarks for the presented reliability-aware scheme presented in this work and the thermal-aware scheme of [17]. Additionally it also compares the increase in code size and compile-time for both schemes. Hence, this table shows that the proposed reliability-aware scheme significantly improves register-file lifetime by combining stress and thermal balancing, while having a limited code overhead increase. Although the increase in compile-time is significant, it is

in fact negligible in comparison with the program lifetime, where program lifetime is calculated as a single compile-time and numerous execution-times.

Table 2: Improvement in different wearout mechanisms and increase in code size for reliability-aware and thermal-aware compilation schemes.

Compilation Scheme	Code Size %	Compile time	NBTI	PBTI	GOBD
Proposed reliability-aware method	0.19%	120%	20%	14%	7%
Thermal-aware method [17]	0.113%	20%	7%	9%	4%

CHAPTER 4

TEST CHIP IMPLEMENTATION FOR BTI MONITORING

A test chip was implemented to verify the effectiveness of the BTI monitoring schemes presented. The aim of the test chip was to integrate the BTI monitoring schemes into actual VLSI building blocks. A multi-bank memory module and a logic block consisting of standard decoder logic were chosen for this purpose. The BTI monitoring test proposed were tested on the chip to verify the methodology on silicon.

The test chip has been implemented in IBM 130 nm CMOS technology which mainly was intended for RF and analog mixed-signal applications. This is a twin well CMOS technology with a non-epi P- substrate with shallow trench isolation. The test chip was implemented using 3 thin, 2 thick and 3 RF metal layers.

Since this is a twin well process, the body biasing option was included for PMOS devices only and body biasing not was supported for the NMOS devices. Body bias support for the PMOS devices means that we don't have to stress the chip to accelerate device degradation. The threshold voltage of a device is a function of its body-to-source voltage. Body biasing can be used to modulate the PMOS device V_t , mimicking the behavior of NBTI without damaging the chip. However for PBTI, the chip needs to be stressed to accelerate the degradation process. However, light stress can be applied, ensuring that other wearout mechanisms are not triggered.

4.1 Top Level Overview

The test chip included a memory macro and a logical data path section as shown in Figure 32.

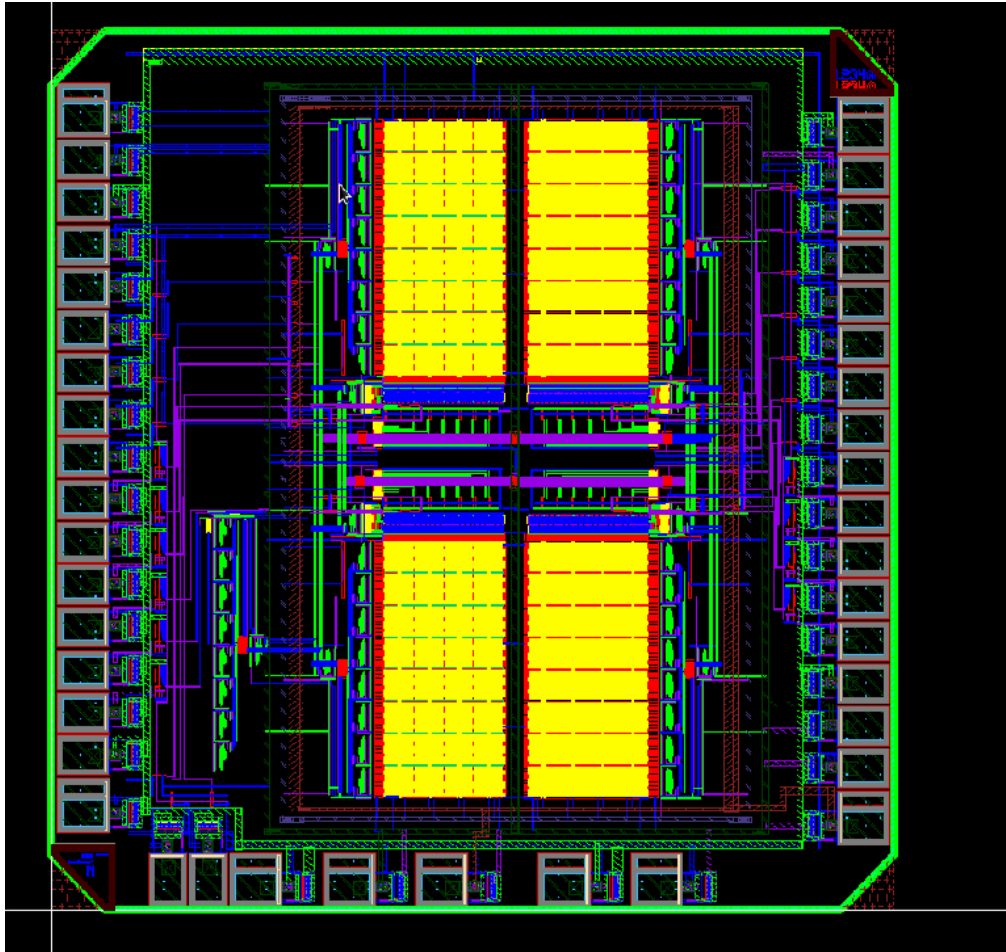


Figure 37: Test Chip layout for BTI monitoring

The monition system was implemented on a 2x2mm die.

Figure 32 shows the top level physical layout of the chip. Most of the chip is occupied by the memory macro. The decoder serving as the logic block can be seen in the bottom left corner. The outer edges of the chip are occupied by the ESD protection circuitry, the PAD drivers and the PADs.

The chip has a total of 41 pads, with 6 for the IO supply, 4 reserved for the internal core supply, 3 for ground, 2 for PMOS body biasing and 25 reserved for IOs. The IOs can be used for acquiring BTI test data. They can also be used for testing normal functionality of the modules implemented.

Testability of both NMOS degradation due to PBTI and PMOS degradation due to NBTI was ensured during the design of the memory module. Depending on the address selected, the selected bitcell can be monitored for pull-up PMOS degradation as well as NMOS degradation of the remaining devices. Both these tests can be performed independently of each other. During NBTI testing, we ensure the testability of PMOS devices only and during PBTI testing; only the NMOS device strength is tested. This means that NBTI and PBTI can be analyzed in isolation for the memory bitcells, which is an extremely challenging task.

Multiple test features were included in the memory macro offering varying testability options. The PBTI testing mode for the bitcell only, as well as the NBTI testing mode for the bitcell was included. BTI monitoring for the bitcell helps in better understanding the shifts in bitcell parameters. This is extremely critical for ensuring bitcell functionality over the passage of its intended operational lifetime. However, the periphery delay is equally important for memory functionality and is also affected by BTI. Therefore, BTI tracking was also implemented in some parts of the periphery for the analysis of the effect of BTI on cycle time and access time degradation. The test mode to monitor the delay in the data path for data-in from address/data to the target bitcell was also included. This meant that the changes in the periphery as well as the bitcell could be monitored and characterized for BTI degradation.

For BTI monitoring of data paths, a standard CMOS static logic based 1-out-of 2^M 128-bit decoder was selected. This is a commonly used block in CPU logic data paths as well as memory systems. Both the data path ring oscillator scheme and the sleep transistor based scheme were implemented to track BTI degradation.

4.1.1 Memory Module Overview

The memory module that was implemented in the test chip was a 64kb macro with a 8192x8 configuration with a mux-4 implementation. An overview of the memory module is shown in Figure 33.

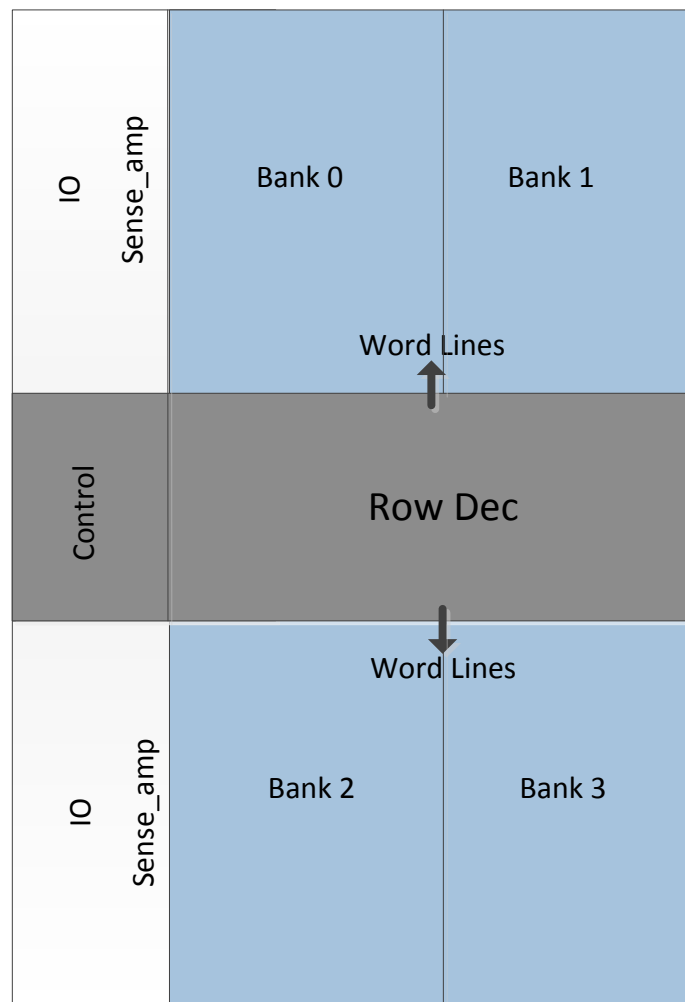


Figure 38: Memory module overview

Each of the four banks in the memory consists of a memory array, a local sense amp, local bit-line drivers for write and IO drivers. Each bank array is a 128x128 configuration with 128 cells per bit-line and 128 cells per word-line. 4 bit-line pairs are muxed into global read and write lines, hence forming a mux-4 configuration. The global read bitline then goes to a sense-amp for read detection. A differential voltage sensing amplifier was used for read detection. The implementation of the sense amp is shown in Figure 34. The differential voltage sensing amplifier takes a small-signal differential input from the array local bit-lines and amplifies it into a rail to rail single ended output.

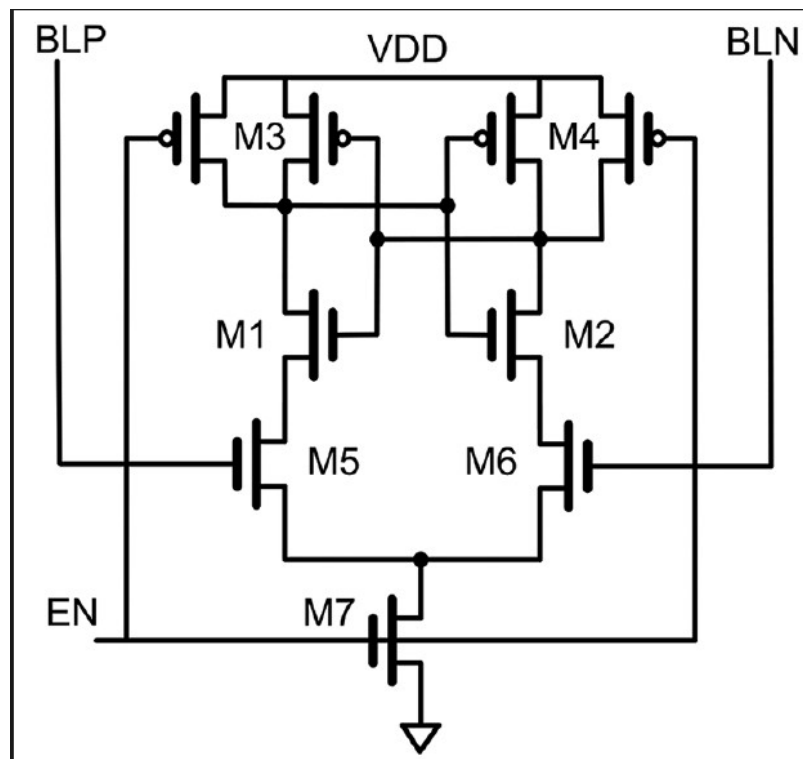


Figure 39: Deferential voltage sensing sense-amplifier

The output of the sense amp is then fed into memory IO drivers which drive the global lines to the chip pad drivers.

A block level representation of the data-out path during a read operation is shown in Figure 40.

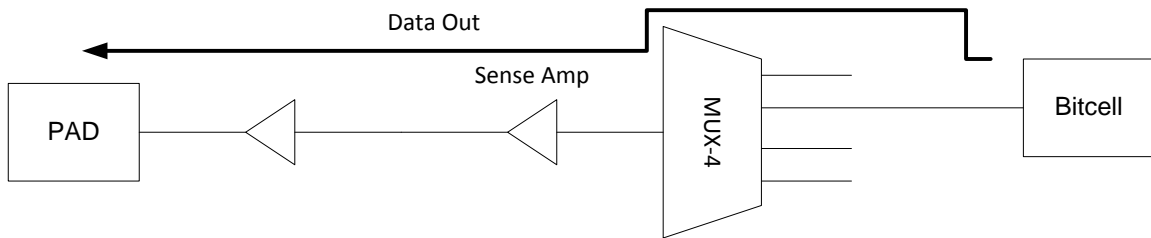


Figure 40: Data out path during read

Similarly the input data and the addresses are driven by memory IO drivers at the input of the memory module. Input data from the IO driver goes to the global write line. The global write-line is connected to a 4-way Mux. Depending on the input address, the write-line goes to one of the 4 possible lines through the Mux and into the local bit-lines. The local bitline is directly connected to bitcell columns and depending on the bitcell row selected, data gets written into the activated bitcell as shown in Figure 41.

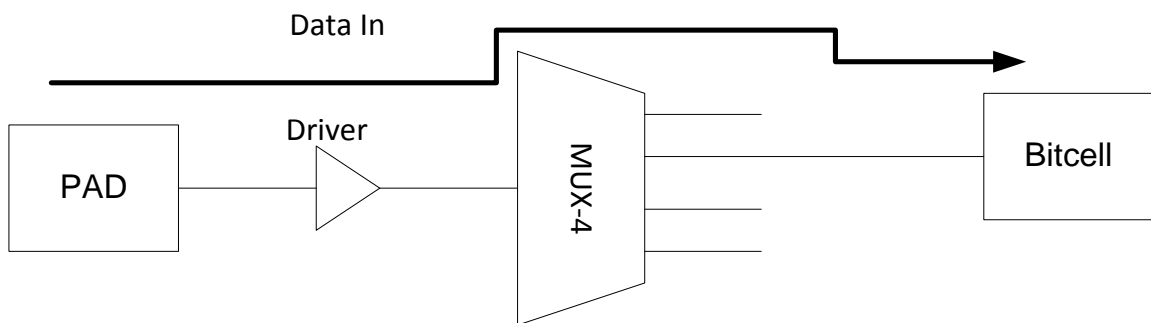


Figure 41: Test Chip for BTI monitoring

As seen from the data-in and data-out paths, two levels of address decoding is required. To select the correct bitcell in the memory array, the correct column as well as the correct row has to be selected. One selection is made for the row through the row decoder while the correct column is selected by the M-way MUX as shown in Figure 43.

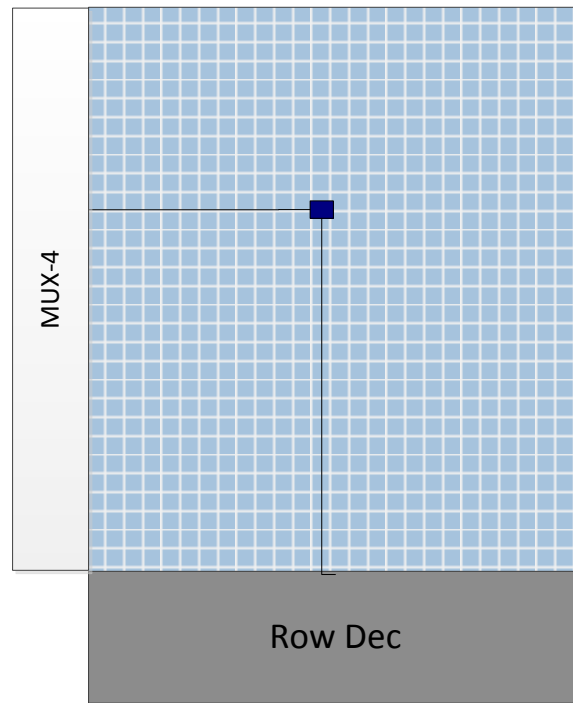


Figure 42: Bitcell selection paths

The two groups of address decoders are known as the column address decoder and the row address decoder. The input address to the chip gets divided into two sections as shown in Figure 44. One of the sections goes into the decoder for the MUX selection signals for the data-in during write and the data-out during read. This selection is known as the column selection. An important observation here is that the column decoder selected is only dependent on the MUX configuration used in the memory macro and is independent of the number of data-in to the memory. A memory with a MUX-1 configuration would not require any column decoding at all and all the column data

would be valid during all operations. A transmission gate type decoder was implemented for the column decoding.

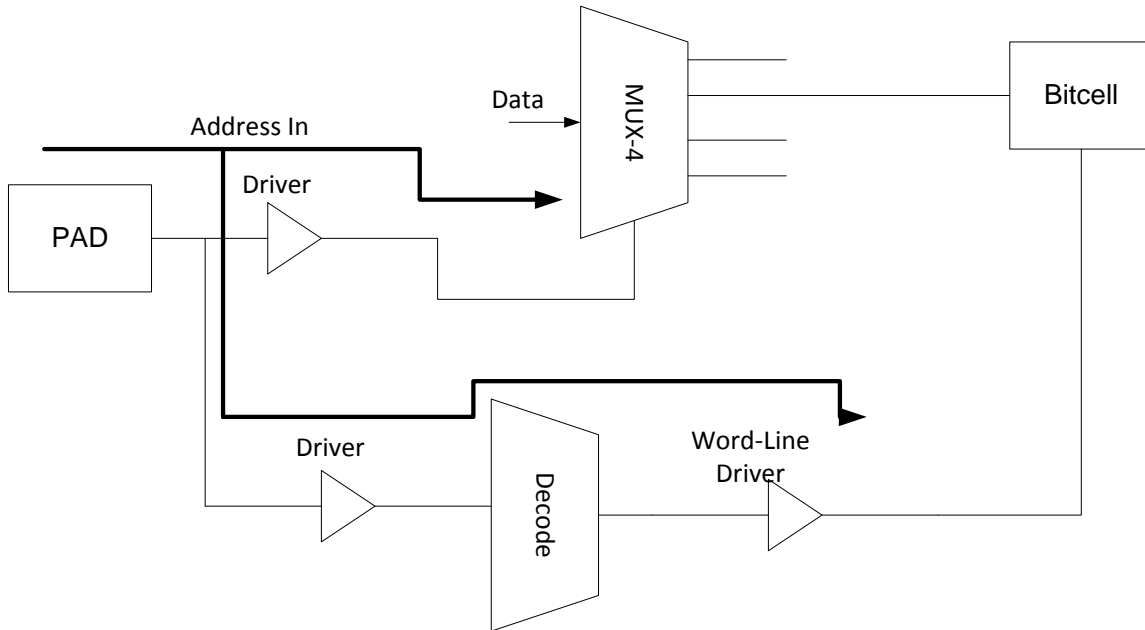


Figure 43: Input address path

The second section of decoding is used for the correct row selection. This is also known as the word-line address since words are arranged in the same row in most memory configurations. Depending on the MUX configuration, we might have one or more words in each memory array row. A MUX-1 configuration would have one word per array row while a MUX-4 configuration like the one used in the test chip had 4 words per memory array row. The second section of address gets decoded depending on the total rows in the array. Since only one row is selected, the decoding only depends on the total number of rows in the memory array. A NAND gate based decoding scheme was used for the row decoder.

4.1.2 Memory Module Test Features

The memory macro had the following test features:

1. **NBTI test for the selected bitcell in the memory macro:** Depending on the input address and the bitcell selected, the memory system can be switched to NBTI-Bitcell test mode from operational mode in which the degradation of the PMOS devices in the bitcell can be monitored.
2. **PBTI test for the selected bitcell in the memory macro:** Depending on the input address and the bitcell selected, the memory system can be switched to PBTI-Bitcell test mode from operational mode in which the degradation of the NMOS devices in the bitcell can be monitored.
3. **BTI test for the address decoders:** Depending on the input address and the row/column selected, the memory system can be switched to BTI-Periphery test mode from operational mode in which the degradation of the NMOS/PMOS devices in the periphery path can be monitored.
4. **Access Time Test for Memory Macro:** Depending on the input address, the row/column selected, and the bitcell selected, the memory system can be switched to PBTI-Periphery+Bitcell test mode from operational mode in which the degradation of the NMOS/PMOS devices in the periphery path can be monitored. In addition, in this mode, any speed degradation of the bitcell NMOS is also added to the output hence combining the affect of the periphery and the bitcell degradation.

4.1.3 Logic Module Overview

The logic module that was implemented in the test chip was a 128 bit 1-to- 2^M decoder macro implemented using a static CMOS NAND-Gate logic based implementation. This is one of the more commonly used decoder implementations owing to its simplicity since it is the direct implementation of the min-terms in the decoder Boolean logic function. A NAND gate based 2-input decoder implementation is shown in Figure 44 as an example. A similar implementation with 7 inputs and 128 outputs was used in the test chip.

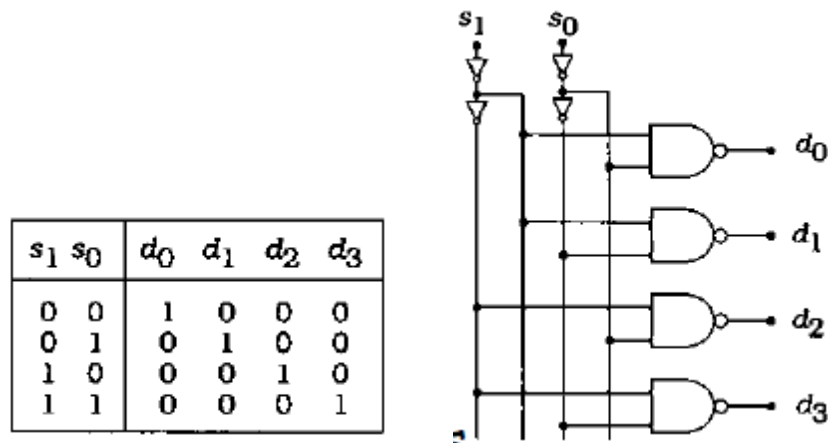


Figure 44: NAND gate based decoder

4.1.4 Logic Module Test Features

The logic macro had the following test features:

1. **BTI test for the logic path using DPRO:** Using the proposed DPRO scheme, every data path in a chip can be turned into an oscillator, whose frequency provides a measurement of delay. Depending on the input address and the logic path selected, the decoder can be switched to BTI-

DPRO test mode from operational mode in which the degradation of the PMOS/NMOS devices in the logic path can be monitored directly by converting the data path into a ring oscillator.

2. **BTI test for the logic path using the sleep transistor based scheme:**

Using the proposed scheme, every head-switched data path in a chip can be turned into an oscillator, whose frequency provides a measurement of delay. Depending on the input address and the logic path selected, the decoder can be switched to BTI-Sleep transistor based test mode from operational mode in which the degradation of the PMOS/NMOS devices in the logic path can be monitored directly by converting the data path into a ring oscillator.

4.2 Test Chip Test Results and Discussion

The results of the test chip for BTI monitoring is presented in the following section.

4.2.1 Test Memory Macro Testing

The results for NBTI/PBTI testing for the bitcell-only, as well as access time test are shown in Figure 46. The test chip was stressed to accelerate the device degradation. As the device threshold voltage is increased, the output frequency slowly decays, showing a clear dependency on device strength.

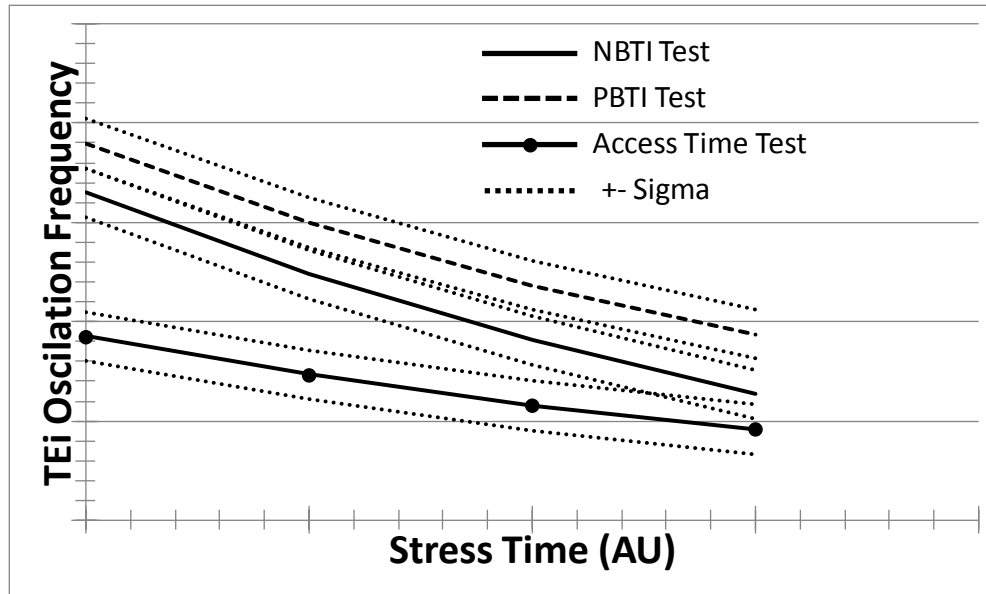


Figure 45: NBTI/PBTI test for bitcell and the access time test

4.2.2 Logic Block Testing

The results for BTI testing for the decoder is shown in Figure 46 for the data path ring oscillator scheme. The test chip was stressed to accelerate the device degradation. As the device threshold voltage is increased, the output frequency slowly decays, showing a clear dependency on device strength.

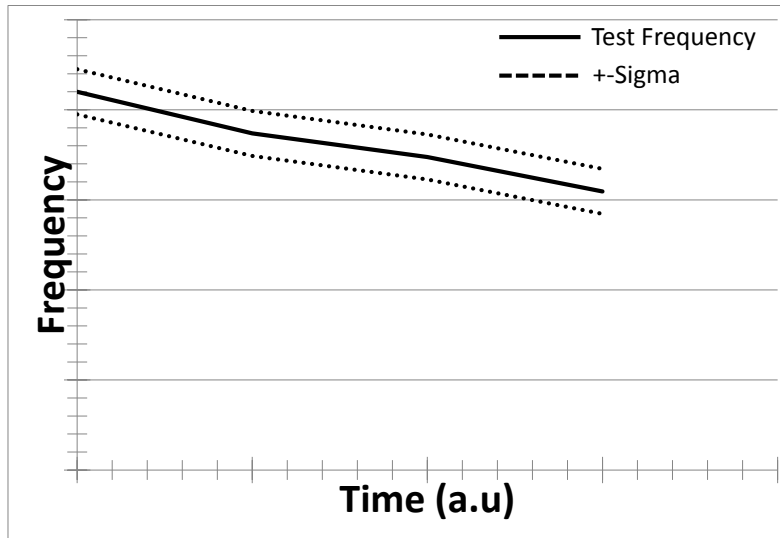


Figure 46: Decoder delay monitoring using DPRO

The results for BTI testing for the decoder is shown in Figure 47 for the sleep transistor based scheme. As the device threshold voltage is increased, the output frequency slowly decays, showing a clear dependency on device strength.

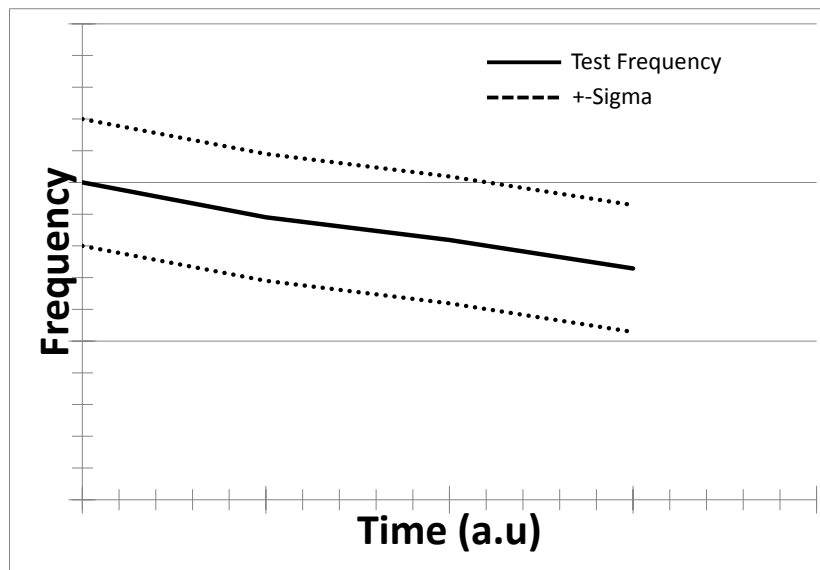


Figure 47: Degradation monitoring using sleep transistor based scheme

CHAPTER 5

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

This dissertation aims to demonstrate the detection of the onset of failure through detection of component degradation over time. Fault-tolerant systems require finding a way to prevent a physical defect or failure from causing an error in system performance. If component degradation can be detected, the cost of building highly reliable systems will be reduced. The additional area needed for fault detection is limited to the design-for-test (DfT) circuitry, which is a small fraction of the component being monitored.

5.1 Summary of Research

The objective of this work is to investigate invasive and non-invasive methods of BTI monitoring. Since the most prevalent method of wearout monitoring is the use of test structures, we propose a novel, simple to use, test structure for NBTI /PBTI monitoring. The proposed structure has an AC and a DC stress mode. Although during stress mode, both PMOS and NMOS devices are stressed, the proposed structure isolates the PBTI and NBTI degradation during test mode. Hence, NMOS and PMOS degradation can be decoupled during testing.

A methodology of converting any data-path into ring oscillator (DPRO) is also presented. This methodology means that the thermal and electrical stress for life time measurements are no longer an approximation as data is directly read off from the functional block.

To avoid the performance overhead of attaching monitoring circuitry to functional block, a non-invasive scheme for BTI monitoring is presented for sleep transistor based logic families. This scheme carries no performance overhead and the complexity of the test block does not change with the complexity of the data path being monitored.

Since, NBTI is a critical issue for memories, a scheme for NBTI monitoring of 6T SRAM cell based memories is also presented. We make use of the concept of a DPRO and show how a memory system can be made to oscillate in test mode. The frequency of oscillation is a function of the PMOS devices in the cell.

The NBTI monitoring scheme presented for the memories only monitors the strength of the PMOS devices. Since PBTI is equally critical in high-k metal-gate technologies, we have also presented a method of using the DPRO scheme to monitor NMOS degradation in 6T cell based memories.

Since the read current of a cell is effectively the currents of the pull-down and the access NMOS devices, PBTI monitoring is similar to read current characterization of a cell. The read current would degrade if any of the two NMOS devices degraded. We have also shown that the scheme for monitoring the NMOS strength is independent of the cell PMOS device strength. Hence the PBTI monitoring scheme senses the degradation in the NMOS devices in an SRAM cell without being affected by the PMOS devices.

Lastly, after validation of the proposed schemes using extensive simulations, we have also validated the results on silicon. We have designed, laid-out and taped-out all the monitors that have been proposed. The tape-out was done in IBM 130nm technology and includes a combinational data paths as well as a memory system.

A 6T cell based memory system with an NBTI and PBTI monitor that monitored the strengths of the PMOS pull-up devices and the NMOS pull-down and access devices at the memory cell level, has been implemented.

The DPRO methodology has been implemented on the combinational memory peripherals such as the row/column decoders. These blocks were made to oscillate, with the frequency of oscillation representing the state of the devices.

The sleep-transistor based methodology has also been implemented on the same combinational peripherals. We were able to assess the effectiveness of both these schemes for monitoring NBTI and PBTI. We have presented a comparison for the performance/area overhead incurred due the introduction of these monitors and the layout complexities associated with each.

We also introduce the concept of wearout mitigation at the compiler level. Using an example of a register file, we present a preemptive method of wearout mitigation using a compiler directed scheme. This scheme can also be extended to other function blocks on a system ensuring extended component lifetime without any hardware involvement.

5.2 Suggestions for Future Work

Our results have shown the effectiveness of BTI wearout monitoring schemes. We have also shown that wearout can be mitigated at the compiler level. Both the schemes have a lot of potential for the future and can open up different directions for future research.

5.2.1 Wearout Mitigation at the OS Level

Multi-core systems are slowly becoming more prevalent. With single core systems, load management has always been very limited. However, the emergence of multi-core systems has been followed by the emergence of power, performance and thermal aware load management schemes at the operating system (OS) level. With device wearout becoming increasingly critical, similar schemes can be implemented at the OS level for wearout mitigation. Together, with compiler level wearout mitigation, we can significantly increase system lifetime using OS level wearout mitigation schemes.

5.2.2 Off-Chip BTI Monitoring for Lifetime and Failure Analysis

This part of the research will explore a novel method of failure analysis.

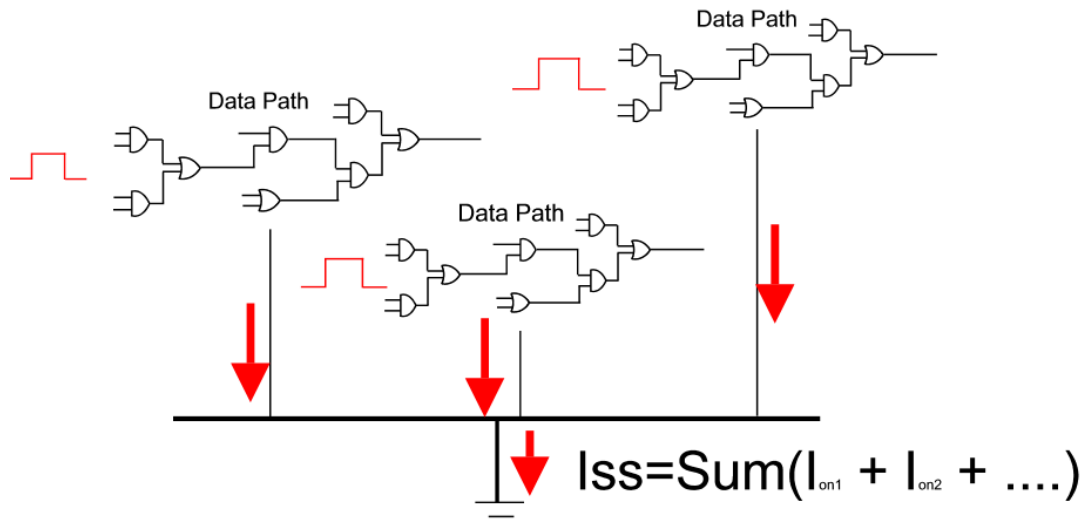


Figure 48: Ground bound is the sum of device currents.

Figure 48 summarizes the concept behind this scheme. VDD and VSS variation is a function of the device currents as shown in Figure 48. As the devices degrade and their

'on' current weakens, the characteristics of VDD and VSS variations are also expected to change. It is proposed that by tracking the changes in ground bounce and voltage droop, we can assess the state of the system being monitored. As an example, Figure 49 shows the expected change in VSS characteristics with increasing device weakening.

This process would involve high precision off-chip measurement techniques to monitor the slight VDD and VSS variations reliably. We would also need advanced DSP and data mining techniques for extracting the necessary useful data from the noisy VDD and VSS signals.

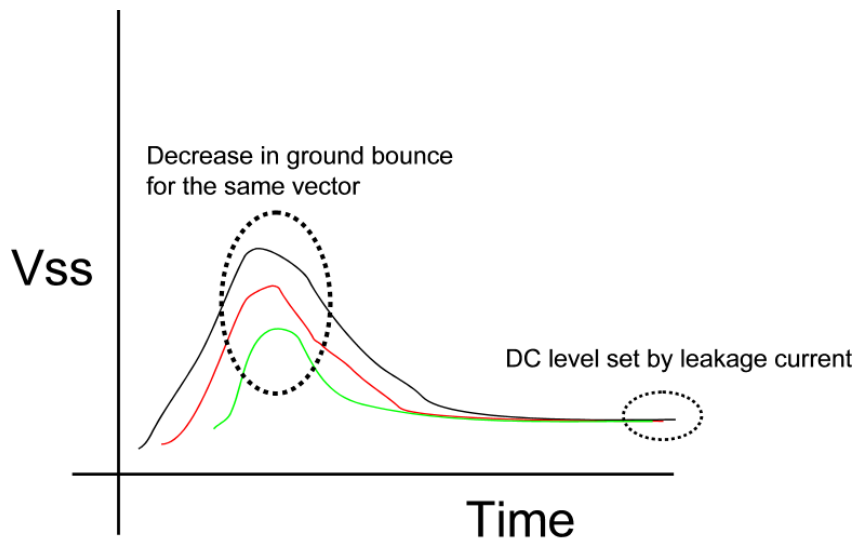


Figure 49: Expected variation in ground bounce with device weakening.

REFERENCES

- [1] M. H. Woods, "MOS VLSI reliability and yield trends," *Proceedings of the IEEE*, vol. 74, pp. 1715-1729, 1986.
- [2] N. Kimizuka, et al., "The impact of bias temperature instability for direct-tunneling ultra-thin gate oxide on MOSFET scaling," in *VLSI Technology, 1999. Digest of Technical Papers. 1999 Symposium on*, 1999, pp. 73-74.
- [3] V. Huard, et al., "Evidence for hydrogen-related defects during NBTI stress in p-MOSFETs," in *Reliability Physics Symposium Proceedings, 2003. 41st Annual. 2003 IEEE International*, 2003, pp. 178-182.
- [4] M. Denais, et al., "Interface trap generation and hole trapping under NBTI and PBTI in advanced CMOS technology with a 2-nm gate oxide," *Device and Materials Reliability, IEEE Transactions on*, vol. 4, pp. 715-722, 2004.
- [5] J. Mitard, et al., "Large-Scale Time Characterization and Analysis of PBTI In HFO₂/Metal Gate Stacks," in *Reliability Physics Symposium Proceedings, 2006. 44th Annual., IEEE International*, 2006, pp. 174-178.
- [6] K. Stawiasz, et al., "On-Chip circuit for monitoring frequency degradation due to NBTI," in *Reliability Physics Symposium, 2008. IRPS 2008. IEEE International*, 2008, pp. 532-535.
- [7] M. Nourani and A. Radhakrishnan, "Testing On-Die Process Variation in Nanometer VLSI," *Design & Test of Computers, IEEE*, vol. 23, pp. 438-451, 2006.
- [8] V. Reddy, et al., "Impact of negative bias temperature instability on digital circuit reliability," *Microelectronics Reliability*, vol. 45, pp. 31-38, 2005.
- [9] X. Wang, et al., "Path-RO: a novel on-chip critical path delay measurement under process variations," in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 640-646.
- [10] M. Alam and S. Mahapatra, "A comprehensive model of PMOS NBTI degradation," *Microelectronics Reliability*, vol. 45, pp. 71-81, 2005.
- [11] W. Wenping, et al., "An efficient method to identify critical gates under circuit aging," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, 2007, pp. 735-740.

- [12] L. Yung-Huei, et al., "Managing Bias-Temperature Instability for Product Reliability," in *Proc. Int. Sym on VLSI Technology, Systems and Applications*, 2007, pp. 1-2.
- [13] K. Tae-Hyoung, et al., "Silicon Odometer: An On-Chip Reliability Monitor for Measuring Frequency Degradation of Digital Circuits," *IEEE j. of Solid-State Circuits*, vol. 43, pp. 874-880, 2008.
- [14] K. Kunhyuk, et al., "Characterization of NBTI induced temporal performance degradation in nano-scale SRAM array using I ∞ DDQ," in *Proc. IEEE Int. Test Conference*, 2007, pp. 1-10.
- [15] F. Ahmed and L. Milor, "Reliable cache design with on-chip monitoring of NBTI degradation in SRAM cells using BIST," in *28th VLSI Test Symposium (VTS)*, 2010, pp. 63-68.
- [16] T. Kim, et al., "Silicon odometer: an on-chip reliability monitor for measuring frequency degradation of digital circuits," in *IEEE Symposium on VLSI Circuits*, 2007, pp. 122-123.
- [17] M. Sabry, et al., "Thermal-aware compilation for system-on-chip processing architectures," in *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, 2010, pp. 221-226.
- [18] J. Srinivasan and S. Adve, "Predictive dynamic thermal management for multimedia applications," in *Proceedings of the 17th annual international conference on Supercomputing*, 2003, pp. 109-120.
- [19] H. Yun and J. Kim, "Power-aware modulo scheduling for high-performance VLIW processors," in *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, 2001, pp. 40-45.
- [20] D. Sylvester, et al., "Elastic: An adaptive self-healing architecture for unpredictable silicon," *IEEE Design & Test of Computers*, vol. 23, pp. 484-490, 2006.
- [21] B. Hamidzadeh, et al., "Dynamic scheduling techniques for heterogeneous computing systems," *Concurrency: Practice and Experience*, vol. 7, pp. 633-652, 1995.
- [22] N. Aggarwal, et al., "Configurable isolation: building high availability systems with commodity multi-core processors," in *Proceedings of the 34th annual international symposium on Computer architecture*, 2007, pp. 470 - 481.

- [23] S. Mutoh, et al., "1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS," *IEEE Journal of Solid-State Circuits*, , vol. 30, pp. 847-854, 1995.
- [24] K. KANG, ET AL., "IMPACT OF NEGATIVE-BIAS TEMPERATURE INSTABILITY IN NANOSCALE SRAM ARRAY: MODELING AND ANALYSIS," *IEEE TRANS. COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS*, VOL. 26, PP. 1770-1781, 2007.
- [25] M. A. ALAM, "A CRITICAL EXAMINATION OF THE MECHANICS OF DYNAMIC NBTI FOR PMOSFETS," IN *PROC. IEEE INT. ELECTRON DEVICES MEETING*, 2003, PP. 14.4.1-14.4.4.
- [26] T. FISCHER, ET AL., "A 65NM TEST STRUCTURE FOR THE ANALYSIS OF NBTI INDUCED STATISTICAL VARIATION IN SRAM TRANSISTORS," IN *PROC. EUROPEAN SOLID-STATE DEVICE RESEARCH CONFERENCE*, 2008, PP. 51-54.
- [27] V. HUARD, ET AL., "NBTI DEGRADATION: FROM TRANSISTOR TO SRAM ARRAYS," IN *PROC. IEEE INT. RELIABILITY PHYSICS SYMPOSIUM*, 2008, PP. 289-300.
- [28] E. Seevinck, et al., "Static-noise margin analysis of MOS SRAM cells," *Proc. IEEE J. of Solid-State Circuits*, vol. 22, pp. 748-754, 1987.
- [29] D. Atienza, et al., "HW-SW emulation framework for temperature-aware design in MPSoCs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, pp. 1-26, 2007.
- [30] D. Qian and D. Dumin, "The Electric Field, Oxide Thickness, Time and Fluence Dependences of Trap Generation in Silicon Oxides and Their Support of the E-model of Oxide Breakdown," in *Proceedings of the 1999 7th International Symposium on the Physical and Failure Analysis of Integrated Circuits*, 1999, pp. 145-150.
- [31] T. Grasser, et al., "Simultaneous Extraction of Recoverable and Permanent Components Contributing to Bias-Temperature Instability," in *IEEE International Electron Devices Meeting*, , 2007, pp. 801-804.
- [32] S. Bhardwaj, et al., "Predictive modeling of the NBTI effect for reliable design," in *Custom Integrated Circuits Conference, CICC*, 2006, pp. 189-192.
- [33] L. Xiaojun, et al., "Compact Modeling of MOSFET Wearout Mechanisms for Circuit-Reliability Simulation," *IEEE Transactions on Device and Materials Reliability*, vol. 8, pp. 98-121, 2008.
- [34] "ACE Cosy Compiler <http://www.ace.nl/compiler/cosy.html>"

- [35] X. Zhou, et al., "Temperature-aware register reallocation for register file power-density minimization," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, pp. 1-26, 2009.