Techniques for Aging, Soft Errors and Temperature to Increase the Reliability of Embedded On-Chip Systems

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

an der Fakultt für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Hussam Amrouch

aus Aleppo

Tag der mündlichen Prüfung: 03.06.2015

Erster Gutachter: Prof. Dr. Jörg Henkel
Karlsruher Instituts für Technologie (KIT)

Zweiter Gutachter: Prof. Dr. Montserrat Nafria Maqueda
Universitat Autonoma de Barcelona (UAB)

Acknowledgements

First and foremost, I would like to express my immeasurable appreciation and sincere gratitude to my advisor Prof. Dr. Jörg Henkel for his unprecedented guidance, enthusiastic encouragement and invaluable critiques. He gave me the complete freedom, while constructively challenging me, to explore various research issues. He has been always there to strongly stimulate me, give me invaluable advice, listen to me, and recover when my steps faltered. His patience and encouragement indeed helped me overcome many obstacles towards finishing this work and, more importantly, his careful feedback helped me sort out the technical details of my research.
Last but not least, the enjoyable work environment, that Prof. Henkel creates at the CES, made me believe in his wonderful perspective which says *"research is always fun"*.

I would also like to extend my deepest gratitude to Prof. Dr. Montserrat Nafria for accepting to be my co-examiner and providing erudite feedback. I have been amazingly fortunate to have a collaboration with her. Without the invaluable and very fruitful discussions with her and with her research group, especially Dr. Javier Martin-Martinez, this work would surely not have been what it became.

Special thanks go to my colleague and friend Victor van Santen. The in-depth technical discussions with him played an important role in improving the quality of this work. I also want to present my thanks to Thomas Ebi. I am indebted to him for his support during my work. I am also thankful to Lars Bauer and Volker Wenzel for the numerous technical discussions that helped me improve my knowledge in the topic area.

Furthermore, I would like to express my deep thankful to Talal Bonny who introduced me to Prof. Henkel. Without his advice and support, at the early stage of my Ph.D., my success in this work would have been exceptionally difficult. I would also like to thank to my colleagues and friends Janmartin Jahn, Sammer Srouji, Artjom Grudnitsky, Fazal Hameed and all other members of the Chair for Embedded Systems (CES) for their support, friendship, and advice. I also want to thank all my students whom I supervised in the scope of this thesis. Additionally, I want to thank Martin Buchty and the secretaries for their essential work at our institute.

I also had the honor to participate at the priority program (SPP 1500) from the German Research Foundation (DFG) about Dependable Embedded Systems. The presentations from a wide range of research groups helped me understand my topic area better.

I am deeply indebted to my parents (Asaad and Thanaa) for their guidance, unlimited love, and encouragement. They always believed in me and gave me the inspiration to apply and complete my Ph.D. journey. They have been a constant source of support, love and strength, not only during my Ph.D., but indeed since ever. I am also grateful to my brother Fadi who gives me the correct advice at the correct moment. I am also grateful to my father-in-law, Khaldoun Azrak, who encouraged me throughout all the stages of my Ph.D. His faith in me always pushed me ahead.

Finally and undoubtedly, none of this would have been possible without the support of my wife, Rama, who was extraordinary patient during my Ph.D studies. I would like to express my utmost heartfelt gratitude to her.

I dedicate this dissertation to all of my family members and Prof. Jörg Henkel.

“Hard Work is Rewarded.” Jörg Henkel

Email conversation after the acceptance notification of TCAD, 23rd November 2013

List of Own Publications Included in This Thesis

Transactions/Journals (blind peer reviewed)

[**1**] H. Amrouch, T. Ebi, and J. Henkel, "RESI: Register-Embedded Self-Immunity for Reliability Enhancement," *Computer-Aided Design of Integrated Circuits and Systems (TCAD), IEEE Transactions on*, vol. 33, no. 5, pp. 677–690, May 2014.

Conferences (double-blind/blind peer reviewed)

[**2**] H. Amrouch, J. Martin-Martinez, V. van Santen, M. Moras, R. Rodriguez, M. Nafria, and J. Henkel, "Connecting the Physical and Application Level Towards Grasping Aging Effects," in *Reliability Physics Symposium (IRPS), IEEE International Conference on*, April 2015, pp. 3D. 1.1–3D. 1.8.

[**3**] H. Amrouch and J. Henkel, "Lucid Infrared Thermography of Thermally-Constrained Processors," in *Low Power Electronics and Design (ISLPED), IEEE/ACM International Symposium on*, July, 2015, pp. 347–352.

[**4**] H. Amrouch, V. van Santen, T. Ebi, V. Wenzel, and J. Henkel, "Towards Interdependencies of Aging Mechanisms," in *Computer-Aided Design (ICCAD), IEEE/ACM International Conference on*, November 2014, pp. 478–485.

[**5**] H. Amrouch, T. Ebi, and J. Henkel, "Stress Balancing to Mitigate NBTI Effects in Register Files," in *Dependable Systems and Networks (DSN), 43rd Annual IEEE/IFIP International Conference on*, June 2013, pp. 1–10.

[**6**] H. Amrouch, T. Ebi, J. Schneider, S. Parameswaran, and J. Henkel, "Analyzing the Thermal Hotspots in FPGA-based Embedded Systems," in *Field Programmable Logic and Applications (FPL), 23rd International Conference on*, September 2013, pp. 1–4.

[**7**] H. Amrouch and J. Henkel, "Self-Immunity Technique to Improve Register file Integrity against Soft Errors," in *VLSI Design (VLSI Design), 24th International Conference on*, Jan 2011, pp. 189–194.

List of Other Publications

Publications That Employed the Developed Thermal Setup in the Scope of This Thesis

[**8**] T. Ebi, H. Amrouch, and J. Henkel, "COOL: Control-based Optimization of Load-Balancing for Thermal Behavior," in *Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/ Software Codesign and System Synthesis*, October 2012, pp. 255–264.

[**9**] D. Palomino, M. Shafique, H. Amrouch, A. Susin, and J. Henkel, "HevcDTM: Application-Driven Dynamic Thermal Management for High Efficiency Video Coding," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2014, pp. 1–4.

[**10**] H. Khdr, T. Ebi, M. Shafique, H. Amrouch, and J. Henkel, "mDTM: Multi-Objective Dynamic Thermal Management for On-Chip Systems," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2014, pp. 1–6.

[**11**] A. Amouri, H. Amrouch, T. Ebi, J. Henkel, and M. Tahoori, "Accurate Thermal-Profile Estimation and Validation for FPGA-Mapped Circuits," in *Field-Programmable Custom Computing Machines (FCCM), IEEE 21st Annual International Symposium on*, April 2013, pp. 57–60.
Received European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC) Paper Award.

Invited Papers

[**12**] J. Henkel, T. Ebi, H. Amrouch, and H. Khdr, "Thermal Management for Dependable On-chip Systems," in *Design Automation Conference (ASP-DAC), 18th Asia and South Pacific*, (invited paper), Jan 2013, pp. 113–118.

List of International Collaborations in the Scope of This Thesis

[C1] Dr. Javier Martin-Martinez and Prof. Montserrat Nafria from the Reliability of Electron Devices and Circuits group (REDEC) within the Electronic Engineering Department, Universitat Autònoma de Barcelona (UAB), Spain.

[C2] Josef Schneider and Prof. Sri Parameswaran from the School of Computer Science and Engineering (CSE), University of New Wales (UNSW), Australia.

List of Supervised Student Projects that Partially Contributed to the Simulation

Master Theses (Diplomarbeiten)

[D1] Michael Skender: Evaluating the Mutual Influence between Applications and Aging Effects, 2015.

[D2] Christian List: Evaluating the Impact of Voltage Scaling on Aging Effects, 2015.

[D3] Victor van Santen: Modeling and Evaluating the Impact of Aging Phenomena on Reliability, 2014.

[D4] Derick Beng: Evaluating and Mitigating the Thermal Hotspot in FPGA-based Embedded Systems, 2013.

Semester/Bachelor Theses (Studienarbeiten)

[S1] Steffan Bähr: Thermal Simulation of FPGA and ASIC Microarchitecture, 2012.

Contents

Acknowledgements **v**

List of Own Publications Included in This Thesis **ix**
Transactions/Journals (blind peer reviewed) . . . ix
Conferences (double-blind/blind peer reviewed) . . . ix

List of Other Publications **xi**
Publications That Employed the Developed Thermal Setup . . . xi
Invited Papers . . . xi

List of International Collaborations **xiii**

List of Supervised Student Projects **xv**
Master Theses (Diplomarbeiten) . . . xv
Bachelor Theses (Studienarbeiten) . . . xv

List of Figures **xxi**

List of Tables **xxvii**

Glossary **xxix**

Acronyms **xxxiii**

Abstract **xxxvii**

Zusammenfassung der Arbeit **xxxix**

The Big Picture behind the Thesis **xliii**

1 Introduction **1**
1.1 Register Files of Microprocessors . . . 2
1.2 SRAM Cells . . . 3
1.2.1 Static Noise Margin . . . 3
1.2.2 Read Access Time . . . 4
1.2.3 Critical Charge . . . 5
1.3 Aging Effects . . . 5

1.3.1 Aging phenomena 5
1.3.2 Impact of Technology Scaling 6
1.4 Soft Errors 9
1.5 On Temperature-related Reliability 12
1.5.1 *Short-terms* Effects of Temperature on Reliability 12
1.5.2 *Long-terms* Effects of Temperature on Reliability 14
1.6 Thesis Contributions 17
1.7 Outline 20

2 Background and Related Work **21**
2.1 Reliability Estimation akin to Aging Effects 21
2.2 Aging Effects Mitigation 24
2.2.1 Aging Stress Balancing 24
2.2.2 State-of-the-art Techniques 26
2.3 Soft Error Mitigation 29
2.4 On-Chip Thermal Investigation 32
2.4.1 FBGA-based On-Chip Systems 32
2.4.2 ASIC-based On-Chip Systems 33
2.5 Summary 34

3 Reliability Estimation Through the Interdependencies of Aging Mechanisms **35**
3.1 Motivation 35
3.2 Problem Formulation 37
3.3 Degradations Modeling 38
3.3.1 Defects due to Aging 39
3.3.2 Superposition of Aging Effects 42
3.4 Reliability Abstraction 44
3.4.1 Data Corruption 44
3.4.2 Timing Violations 44
3.4.3 Interpreting Aging-induced Degradations to Failure Analysis 45
3.4.4 Implementation Details of Our Reliability Estimation 47
3.4.5 Failure Probability Estimation 48
3.5 Validation 50
3.6 Aging Effects Analysis 52
3.6.1 Transistor Electrical Characteristics 52
3.6.2 Impact of Considering Carrier Mobility 53
3.6.3 Abstracting Aging Effects at the System Level 54
3.6.4 Limitations: 56

4 Increasing the Reliability of Register Files against Aging Effects **59**
4.1 Exploration of Aging Effects in Register Files 59
4.2 Our Proposed Technique RISB: Register-Internal Stress Balancing 62
4.3 Implementation 63
4.3.1 Frequent Registers 63
4.3.2 Infrequent Registers 64
4.3.3 Other Microrchitectures 69

5 Increasing the Reliability of Register Files against Soft Errors 71
5.1 Our Proposed Register-Embedded Self-Immunity Technique: Register-Embedded Self-Immunity (RESI) . . . 72
5.1.1 Single Bit Upsets . . . 72
5.1.2 Multiple Bit Upsets . . . 75
5.1.2.1 Reducing Overheads . . . 75
5.1.2.2 Microarchitectural Support . . . 77
5.2 On Effectiveness . . . 78
5.2.1 Impact on Area . . . 78
5.2.2 Impact on Power Consumption . . . 78
5.2.3 Impact on Aging Effects . . . 80

6 Evaluation, Comparison and Advantages 81
6.1 Experimental Setup . . . 81
6.2 RESI Technique Evaluation . . . 82
6.2.1 Impact on Area . . . 82
6.2.2 Power Consumption . . . 83
6.2.3 Impact on temperature and performance . . . 84
6.2.4 Register File Vulnerability . . . 87
6.2.5 Aging Stress Balancing . . . 88
6.2.6 Fault Coverage Estimation . . . 90
6.3 RISB Technique Evaluation . . . 96
6.3.1 Potential Overhead . . . 96
6.3.2 Aging Stress Balancing . . . 97
6.3.3 Sensitivity to input data . . . 98
6.3.4 Failure Analysis and State-of-the-art Comparison . . . 99
6.4 Reliability Evaluation under *Conforming Workloads* . . . 102
6.4.1 Connecting the System Level (i.e. Workloads) and the Physical Level (i.e. Aging Effects) . . . 103
6.4.1.1 Software-based Technique . . . 104
6.4.1.2 Hardware-based Technique . . . 106

7 Lucid Infrared Thermography of Embedded On-Chip Systems 113
7.1 Motivational Case Study . . . 114
7.1.1 Thermal Simulations . . . 114
7.1.2 On-Chip Thermal Diode Sensor . . . 116
7.1.3 Ring Oscillator-based Thermal Sensors . . . 117
7.2 FPGA-based On-Chip Systems . . . 120
7.2.1 Experimental Setup . . . 120
7.2.1.1 Emissivity Aspect . . . 120
7.2.2 Thermal Characteristics Investigation . . . 121
7.2.3 Evaluating the Thermal Impact of Cache . . . 122
7.2.4 Our Proposed Thermal Cache Model . . . 124
7.3 ASIC-based Multi-Core Systems . . . 127
7.3.1 State-of-the-art Liquid-based Setups for IR Thermography . . . 128
7.3.2 Obstacles behind Capturing Lucid IR Images . . . 128

7.3.3 Our Proposed RAMA Technique for Lucid IR Images of Multi-Cores Processors . . . 130
7.3.4 Evaluation, Comparison and Advantages . . . 130
7.3.4.1 Steady-State Temperature Equivalent . . . 132
7.3.4.2 Jeopardy of Equivocal IR Images . . . 135
7.3.4.3 Impact of Inaccurate Thermal Analysis on Reliability . . . 135
7.4 Summary of Our Thermal Investigation . . . 140

8 Conclusion and Outlook 141
8.1 Thesis Summary . . . 141
8.2 Current Limitations and Future Work . . . 143

9 Appendixes 145
9.1 Aging Analysis . . . 145
9.1.1 Parameters of Equations . . . 145
9.1.2 Material Structures . . . 146
9.1.3 Breaking and Healing Si−H Bonds . . . 147
9.2 Conforming Workloads Analysis . . . 148

Bibliography 149

List of Figures

1 The pyramid for dependable embedded systems [CES6] xliii

1.1 Simulated steady-state thermal map of the Alpha CPU showing that the register file component has an elevated temperature 2
1.2 A standard 6-T SRAM cell along with the static noise margin (SNM) that represents its resiliency against noise 4
1.3 Technology scaling results in constant electric fields when both area and V_{dd} scale with the same factor (e.g., $S_L = S_V = 0.5 \Rightarrow E_{new}(Channel) = E_{old}(Channel)$). However, scaling within the nano-CMOS era results in higher fields as area and V_{dd} are not scaled anymore with the same factor (e.g., $S_L = 0.5$, $S_V = 0.8 \Rightarrow S_L < S_V \Rightarrow E_{new}(Channel) > E_{old}(Channel) \Rightarrow$ more induced defects by aging and thus higher reliability degradations) 7
1.4 Impact of scaling on a transistor 8
1.5 Impact of voltage scaling on increasing the susceptibility to soft errors . . 9
1.6 The SRAM susceptibility to radiation due to different operating voltage showing how lower V_{dd} significantly increases the probability of failure (P_{fail}) in 22 nm SRAM cells due to soft errors. Therefore, increasing the resiliency to soft errors in low-power embedded on-chip systems is indispensable 9
1.7 Overview of the register file reliability with respect to aging effects and soft errors 11
1.8 IR image of a chip that violates thermal constrains resulting in *short-term* and *long-term* effects with respect to reliability 12
1.9 Impact of *short-term* effects of temperature on degrading the key reliability metrics of SRAM cells. The same voltage of 1.2 V has been used for the three different scenarios for a fair comparison. The shifts in SNM and Q_{crit} histograms (a and b) towards left indicate less reliability. Whereas, shifts in RAT histogram (c) towards right indicate less reliability 13
1.10 Impact of elevated temperatures on increasing the susceptibility of SRAM cells to soft errors. The same voltage of 1.2 V has been used for the three different scenarios for a fair comparison. Higher temperatures noticeably increase the probability of failure due to soft errors 13
1.11 Evaluating the impact of different operating conditions of operating voltage (V_{dd}) and temperature (T) on the soft-error rate (SER) of 22 nm SRAM cells showing how voltage scaling is a double-edged sword with respect to reliability 14
1.12 Impact of *long-term* effects of elevated temperature on degrading the electrical properties of transistors due to aging phenomena 15

1.13 A high-level overview of the contributions within this thesis 17
1.14 The involved abstraction levels within the contributions of this thesis . . . 18

2.1 Impact of not considering the recovery mechanism of BTI on overestimating aging-induced degradation 22
2.2 Abstracting aging effects 23
2.3 Aging stress within a 6-T SRAM cell 24
2.4 Impact of V_{TH} increase in PMOS transistors within a 6-T SRAM on the static noise margin (SNM) [5, 13] 24
2.5 Impact of aging on SNM degradation for the *balanced* and *unbalanced* aging stress cases. As shown, *unbalanced* aging stress results in shifting the SNM distribution more towards the left side which, in turn, makes the SRAMs resiliency against noise less leading to a higher probability of data corruption-induced failure 25
2.6 Vulnerable and invulnerable intervals during accessing a non-protected register, where soft errors in invulnerable intervals have no deleterious effect on the other microarchitectural components within the on-chip system 30

3.1 Our flow for superposing multiple aging effects 38
3.2 Key aging defects in a PMOS transistor 39
3.3 Probability density function (Pdf) of both Q and Q_{crit} to calculate the probability of failure when a particle strikes an SRAM cell $P_f(Qcrit)$ . . . 45
3.4 Employed safety margins define if the aging-induced degradations can be tolerated or they cause failures 47
3.5 Transistor geometries 49
3.6 Failure analysis from our proposed implementation 50
3.7 Soft error analysis in the presence of aging 51
3.8 Validation of V_{TH} shift due to NBTI and PBTI against measurements published in [14] 51
3.9 Validation of mobility (μ) against measurements published in [15]. A slight mismatch was needed to get a good matching in terms of drain current validation presented in Figure 3.10, as the latter has the highest priority due to its prime impact on the MOSFET operation 52
3.10 Validation of the combined aging model against industrial measurements [16]. PBTI alone in green boxes and PBTI *simultaneously* occurring with HCID in brown circles 52
3.11 Transistor degradations due to BTI and HCID aging mechanisms *separately* / *simultaneously* considering 53
3.12 Comparison between our proposed combination of multiple *simultaneous* aging mechanisms and state-of-the-art 54
3.13 Register file reliability estimation comparison showing the underestimation when only an individual aging mechanism is considered even through examining both kinds of failures 55
3.14 Register file reliability estimation comparison showing the underestimation when only an individual aging mechanism is considered together with examining only a single kind of failures 55
3.15 Failure analysis of the register file SRAM cells 56

3.16 Our proposed in-house implementation to bridge the gap between application-induced stress at the system level and induced defects at the physical level together with abstracting the corresponding induced failures to estimate the overall reliability . . . 57

4.1 Register file statistics results gathered throughout all 16 benchmarks show the aging stress analysis (a and b) and the probability of the most significant *16-bits* and *8-bits* of a written value to be contiguous zeros (c) . . . 60
4.2 Flow diagram of our proposed technique. To estimate the register file reliability before and after applying an aging mitigation technique, we employ our presented work in Chapter 3 . . . 64
4.3 Mechanism of selectively balancing the aging stress in the *frequent* registers and *infrequent* registers of a register file according to our proposed technique . . . 65

5.1 Register file statistics gathered throughout various 10 applications show the distribution of the *manipulatable/non-manipulatable* register values and the corresponding fraction of vulnerable intervals . . . 73
5.2 Microarchitectural support for writing and reading a register value when our RESI protection technique against SBUs is applied . . . 74
5.3 Our proposed RESI technique in the case of MBUs for encoding a written value and how the register bits need to be arranged . . . 76
5.4 Distribution of the proposed Available Bits (AvB) of register values for different benchmarks . . . 77
5.5 Hardware flow diagram of our proposed RESI technique against MBUs together with a scenario of an application of it . . . 79

6.1 Power saving (for the case of ASIC estimation) in our RESI technique for MBUs compared to *FullySBU*, i.e. protecting the register file only against SBUs . . . 85
6.2 Total processor power consumption behavior during runtime (for the FPGA implementation) shows that our RESI technique consumes less power . . . 85
6.3 Normalized Overhead results compared to the *Base* version for the FPGA implementation . . . 86
6.4 The used experimental setup for our thermal measurement (left) and the obtained chip thermal image of the three implementations . . . 86
6.5 Comparisons of the register file vulnerability reduction. "N" represents the number of protected registers in the partial protection scheme . . . 87
6.6 Impact of our RESI technique on mitigating the aging effects . . . 89
6.7 System fault coverage comparison for different benchmarks in the case of *SBUs* injection . . . 92
6.8 System fault coverage comparison for different benchmarks in the case of MBUs occurred in *adjacent bits* of the register file . . . 93
6.9 System fault coverage comparison for different benchmarks in the case of *MBUs* occurred in *random bits* of the register file . . . 93
6.10 System fault coverage comparison for different benchmarks in the case of *adjacent MBUs* when a *realistic fault distribution* [17] is applied . . . 94

6.11 System fault coverage comparison for different benchmarks in the case of *random MBUs* when a *realistic fault distribution* [17] is applied 94
6.12 System fault coverage comparison for different benchmarks in the case of *adjacent MBUs* when a *realistic fault distribution* [17] is applied and the entire protected register file is under evaluating 95
6.13 System fault coverage comparison for different benchmarks in the case of *random MBUs* when a *realistic fault distribution* [17] is applied and the entire protected register file is under evaluating 95
6.14 Total power overhead after implementing our introduced RISB technique to mitigate the aging effects in the register file 97
6.15 The aging stress evaluation in the register file after implementing our technique using the *small input* data set (profiling phase) 98
6.16 The aging stress evaluation in the register file after implementing our technique using the *large input* data set (execution phase) 99
6.17 Percentage of the *duty cycle* (λ) values of the individual register file bits that are within the range of [0.4 - 0.6], where the aging-induced reliability degradation is minimum 100
6.18 The reliability degradation for the 22 nm technology node after 10 years compared to state-of-the-art showing that our RISB always achieves better results 100
6.19 Motivational example showing how different applications may result in varied aging-stress and temperature profiles and thus they may differently stimulate aging effects (i.e. generate different number of defects at the physical level). Note that each data point represents an interval of 0.1 msec 103
6.20 *Software-based* technique to extract the impact of running workloads on *driving* aging effects in the SRAM cells within the register file. The Alpha processor has been targeted here, where its register file consists of 80 registers each register has a bit-width of 64 bits 105
6.21 Register file reliability estimation in the case of running an individual application on top of the Linux OS 106
6.22 Our *hardware-based* technique to extract the role of running workload on *driving* aging effects in the register file 109
6.23 The aging stress histograms represented by the *normalized duty cycle* ($\grave{\lambda}$) showing the analysis of different applications *individually* as well as *simultaneously* running on top of the Linux OS 110
6.24 Register file reliability estimation in the scenario of applications running *individually* as well as *in parallel* on top of the Linux OS 111

7.1 Comparison between simulated thermal map and measured IR thermal image showing that the simulated results display larger amounts of thermal variation due to inaccurate simulation of leakage power in blocks without active slices 115
7.2 The simulation flowgraph to obtain the thermal image of an FPGA 115
7.3 The thermal image shows that a single fixed thermal diode sensor may not capture a thermal hot spot when the chip is under an intense stress . 116
7.4 Measured thermal images of a Virtex-5 FPGA for various designs and under different ambient temperatures 117
7.5 A ring oscillator-based thermal sensor which can be used to translate the increase of delay to temperature changes 118

7.6 Estimated 3D thermal profile of a design with a low logic activity resulting in smooth temperature changes . . . 119
7.7 Estimated 3D thermal profile of an FPGA die in the case of intense logic activity leading to severe *RO* instability . . . 119
7.8 Our experimental setup used for thermal measurement and the *emissivity* tests of different chips. This shows that measurements of materials with low *emissivity* are very inaccurate as most of the heat measured is reflected from the surroundings . . . 121
7.9 Infrared images of FPGA-based embedded processors obtained by our thermal setup show that the thermal hot spot is located on the chip's border due to the dominant role of the memory interface on the entire FPGA temperature . . . 123
7.10 Analyzing the impact of cache component in FPGA-based embedded systems on the chip's temperature . . . 124
7.11 Our flow diagram to develop the model linking the cache configuration with the peak system temperature . . . 126
7.12 Simulated steady-state thermal maps through the thermal HotSpot tool [18] to demonstrate the potential temperature increase due to removing the cooling (heat sink and packaging) from the chip . . . 127
7.13 Liquid-based thermal measurement setup applying an IR coolant oil on top of the measured chip to prevent overheating . . . 128
7.14 Thermal measurement experiments demonstrating the impact of mineral oil applied to a hot object . . . 129
7.15 Our proposed RAMA technique for lucid IR thermography of processors . 131
7.16 The employment of a thermoelectric device in our RAMA setup to avoid adding any layer on top of the chip . . . 132
7.17 Measured IR images of run-time behavior of the tested processor under an intense stress (left) with the temperatures of individual cores when our proposed cooling technique is applied compared to those when the original heat sink-based cooling is utilized (right) showing that our RAMA technique is representative of the original cooling . . . 133
7.18 Spatial thermal gradient calculation . . . 135
7.19 Measured IR images along with the corresponding spatial thermal gradient maps of Intel 22 nm *octa-core* chip under different scenarios of running an intense workload on two particular cores . . . 136
7.20 Adding a layer of oil on top of the measured chip losses the captured IR images its *lucidity*, due to the thermal convection, which jeopardizes the spatial thermal gradient analysis . . . 137
7.21 Impact of state-of-the-art inaccurate thermal analysis on incorrectly estimating aging for a lifetime of 10 years . . . 138
7.22 Impact of a thermal mis-measurement (as the case for current state-of-the-art thermal setups) of *7°C* on the key reliability aspects of 22nm SRAMs and how it noticeably alters the SRAMs characteristics (i.e. their resiliency against failures). Such a variation comes as a result from analyzing the spatial thermal gradient based on IR images captured *when a thin layer of oil (1mm) is applied on top of the chip* during measurements. 139

9.1 Crystalline and amorphous structure of the SiO_2. The figures are from [19] 146

List of Tables

4.1 The potential effects of the chosen interval to toggle the *infrequent* registers on the execution cycles and percentage of aging-balanced register file SRAM cells . . . 68

6.1 Comparison of area, delay, and power for different Hamming encoders and encoders . . . 84
6.2 Processor behavior for adjacent and random MBUs injection per 1000 simulation runs . . . 91

7.1 Thermal Behavior for different designs showing how the fixed thermal diode may fail in capturing the actual thermal hots pot . . . 117
7.2 Model parameters in our Xilinx Virtex-5 FPGA platform . . . 125
7.3 Maximum estimation error between our model and infrared camera measurements for different benchmarks . . . 125

Glossary

Numbers | A | B | C | D | E | F | H | I | L | M | N | P | R | S | T | U | V | W | Z

Numbers

6-T SRAM A standard six-transistor SRAM cell.

8-T SRAM A standard eight-transistor SRAM cell. It is more resilience against noise than the 6-T SRAM cell but at higher area and power costs.

A

aging stress A MOSFET is considered under the BTI aging stress, i.e. BTI-induced defects are generated, when the vertical electric field over its gate dielectric is applied. It is considered under the HCID aging stress when the horizontal electric field across its channel is applied. When both electric fields are together applied, both BTI and HCID phenomena simultaneously occur.

B

BL The SRAM bit line. The SRAM complementary bit line is called $\overline{BL}$.

BSIM Transistor physics-based models to address the MOSFET physical effects employed for circuit simulation and CMOS technology development [20].

C

conforming workloads They are similar workloads to those that may be caused through the end-user software of the on-chip system under typical operating scenarios. In practice, we define the *conforming workloads* within this thesis as multiple parallel applications along with an operating system running on top of an embedded on-chip system.

CPUBurn A program designed to heat up any x86 CPU to the maximum possible operating temperature that is achievable by using ordinary software [21].

Critical Charge The minimum amount of deposited charges to flip an SRAM.

D

DDR Double Data Rate (DDR) Synchronous Dynamic Random-Access Memory (SDRAM).

Dennard Scaling The Dennard scaling [22] indicates that that as MOSFET transistors become smaller the power density needs to remain constant [23]. In the Deep nano scale, Dennard scaling appears to discontinue due to the challenges that come with the very small MOSFET sizes such as current leakage.

DLX A simplified implementation of the MIPS architecture.

duty cycle The probability (λ) that the logic value '0' is stored within an SRAM cell during runtime. $\lambda = 0.5 \rightarrow$ *well-balanced* aging stress. $\lambda = 0$ or $\lambda = 1 \rightarrow$ *unbalanced* aging stress.

E

emissivity Percentage of heat is emitted from the material in the infrared spectrum. *emissivity* = '1' $\Rightarrow$ perfect emitter (*black body*), *emissivity* = '0' $\Rightarrow$ perfect reflector (*shiny mirror*).

F

frequent A frequent register is a register that is very repeatedly written and thus it has a well-balanced aging stress in its SRAM cells (i.e. λ is close to the value 0.5).

H

high-K As technology nodes were being reduced to below 45 nm, the insulating quality of the gate dielectric became much less and thus leakage currents due to tunneling significantly increased. To solve this problem, manufactures replaced silicon dioxide, that has been traditionally used to form the gate dielectric layer, with a material has a higher dielectric constant K such as hafnium oxides (HfO_2)-based dielectric layer in Intel [24].

I

infrequent An infrequent register is a register that is seldom written and thus it has unbalanced aging stress in its SRAM cells (i.e. λ is far from the value 0.5).

L

LEON3 An open-source synthesisable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture [25].

lower half The half of the register where the least significant bits (LSB) are stored.

lucid thermal images Infrared images captured by a thermal camera under minimal interference with the infrared radiation emitted from the measured object.

M

MediaBench A tool for evaluating and synthesizing multimedia and communications systems [26].

MiBench A free, commercially representative embedded benchmark suite [27].

MIPS Reduced Instruction Set Computer (RISC) Instruction set (ISA) developed by MIPS Technologies.

Moore Gordon Earle Moore (born 1929) is a co-founder of Intel, a semiconductor chip maker corporation.

N

nano-CMOS era Continuous technology scaling made the transistors feature sizes become in the nanometer regime.

NMOS N-Type Metal-Oxide-Semiconductor Logic.

normalized duty cycle Normalizing the *duty cycle* ($\grave{\lambda}$) of an SRAM cell for a clearer representation of the aging stress within SRAM-based components. In practice, $\grave{\lambda} = |\lambda - 0.5| * 2 \in [0, 1]$, $\grave{\lambda} = 0 \rightarrow$ *well-balanced* aging stress, $\grave{\lambda} = 1 \rightarrow$ *unbalanced* aging stress.

P

PARSEC A benchmark suite composed of multithreaded programs. The suite focuses on emerging workloads and was designed to be representative of next-generation shared-memory programs for chip-multiprocessors [28].

PMOS P-Type Metal-Oxide-Semiconductor Logic.

Probability Density Function It is a function that describes the relative probability for this random variable to take on a given value [29]. It is defined as the derivative of the (cumulative) distribution function [30].

R

Read Access Time The required time until the SRAM properly provides its stored data.

reliability It is defined as the ability of a system or component to function under stated conditions for a specified period of time [31].

S

safety margin It is chosen by the system designer to allow for variability in the design (either at the beginning due to manufacturing or later on due to aging degradations).

SPARC V8 Scalable Processor ARChitecture developed by Sun Microsystems.

SPICE Simulation Program with Integrated Circuit Emphasis. It has been developed at the Electronics Research Laboratory of the University of California, Berkeley.

Static Noise Margin The resiliency of an SRAM cell against noise.

T

TSMC Taiwan Semiconductor Manufacturing Company.

U

UART Universal Asynchronous Receiver/Transmitter.

UnACE It represents the vulnerable intervals of execution time where the bit is not susceptible against soft errors.

upper half The half of the register where the most significant bits (MSB) are stored.

V

V_{dd} Supply voltage.

ΔV_{TH} Transistor Threshold Voltage (V_{TH}) Shift.

VHDL (Very High Speed Integrated Circuit) Hardware Description Language.

W

WL The SRAM write line.

workload We define a workload as software running on the embedded on-chip system. This may be a single application which is executed on bare metal or single/multiple applications which are executed on top of an operating system (i.e. *conforming workloads*).

Z

zero register The *zero register* within the register file is to provide a constant zero value. It is often implemented using "hard-wired" SRAM cells through *tied-to-gnd* or *tied-to-Vdd* standard cells.

Acronyms

default | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **L** | **M** | **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V**

default

μ Carrier Mobility.

A

ABP Activity Bits Probability.

ACE Architecturally Correct Execution.

ASIC Application-Specific Integrated Circuit.

AvB Available Bits.

AVF Architectural Vulnerability Factor.

B

BRAMs Block RAMs.

BTI Bias Temperature Instability.

C

CPU Central Processing Unit.

D

DSPs Digital Signal Processings.

E

ECC Error Correction Code.

F

FFs Flip-Flops.

FPGA Field-Programmable Gate Array.

G

g_m Transconductance.

Geant4 Passage of Particle Through Matter.

GND Ground.

H

HCID Hot Carrier Induced Degradation.

I

I_D Drain Current.

IC Integrated Circuit.

IR Infrared.

ITRS International Technology Roadmap for Semiconductors.

L

LUTs LookUp Tables.

M

MBUs Multiple Bit Upsets.

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor.

MTTF Mean-Time-To-Failure.

N

N_{HT} Hole Traps.

N_{IT} Interface Traps.

N_{OT} Oxide Traps.

NBTI Negative Bias Temperature Instability.

NCD Native Circuit Description.

O

OS Operating System.

P

P_{fail} probability of failure.

PBTI Positive Bias Temperature Instability.

PC Program Counter.

Pdf Probability Density Function.

PTM Predictive Technology Model.

Q

Q_{crit} Critical Charge.

R

RAMA Rear-side Thermoelectric-based IR Thermography.

RAMP Reliability-Aware MicroProcessor.

RAT Read Access Time.

RDF Random Dopant Fluctuation.

RESI Register-Embedded Self-Immunity.

RF Register File.

RISB Register-Internal Stress Balancing.

RO Ring Oscillator.

RTL Register-Transfer Level.

RTN Random Telegraph Noise.

RVF Register File Vulnerability.

S

SBUs Single Bit Upsets.

SEC Single Error Correction.

SEC-DED Single-Error Correcting and Double-Error Detecting.

SER Soft Error Rate.

SNM Static Noise Margin.

SOFR Sum-Of-Failures-Rate.

SRAM Static Random-Access Memory.

T

TDDB Time-Dependent-Dielectric Breakdown.

TDP Thermal Design Power.

TMR Triple Modular Redundancy.

U

UCF User Constraints File.

V

V_{TH} Threshold Voltage.

VCD Value Change Dump.

Abstract

Advances in technology have paved the way for making embedded on-chip systems ubiquitous in our daily life. Unfortunately, compared to previous generations, the current so-called nano-CMOS era introduces reliability challenges at an increased pace. As a matter of fact, technology scaling is reaching its limits where certain aspects endanger the correct functionality of hardware/software on-chip systems. They have been enumerated by the International Technology Roadmap for Semiconductors (ITRS) [32]. Of these aspects, soft errors and aging effects are at the forefront and thus there is an indispensable need to increase the reliability of on-chip systems with respect to them.

Aging effects, for instance, are caused by various physical phenomena such as Negative Bias Temperature Instability (NBTI), Positive Bias Temperature Instability (PBTI) and Hot Carrier Induced Degradation (HCID) etc. These phenomena simultaneously occur within the gate dielectric of a transistor where their induced effects alter the electrical characteristics of transistors (e.g., Threshold Voltage (V_{TH})). Particularly, when dealing with structure sizes in the deep nano scale, even small changes may have a significant impact on transistors behavior, making them unreliably operate. Additionally, soft errors – due to energetic particles such as neutrons – also jeopardize the reliability of on-chip systems as they may cause spurious bit flips that influence the applications correctness. This holds even more when operating the circuits at lower voltages as the difference between supply and threshold voltages becomes narrower leading to smaller safety margins (i.e. even few particle-induced charges can disturb the transistor's reliability). Despite the fact that both aging effects and soft errors originate from the physical level (i.e. underlying hardware), they are spatially and temporally *driven* by the running workloads (i.e. software) at the system level. Importantly, they may propagate through different levels, from the physical up to the system level, to finally have a deleterious impact there and jeopardize the reliability of the entire on-chip system. The rise in susceptibility to both aging effects and soft errors makes the register file within on-chip systems one of the important concerns when it comes to reliability. The key reason behind this is its high utilization (very often accessed) that allows faults induced by aging effects and/or soft errors to spread from there throughout the entire on-chip system. Therefore, mitigating the deleterious effects of aging and soft errors within register files is imperative to maintain a reliable operation of on-chip systems during their lifetime. On the other hand, technology scaling has also increased on-chip power densities making current and upcoming chips thermally constrained due to the negative impact of elevated temperatures on reliability (e.g., an increase in the chip's temperature contributes to faster transistor aging). Therefore, accurately exploring the thermal characteristics of chips

is fundamental to correctly estimating their reliability and it is also a prerequisite for developing reliability-aware hardware/software techniques. For that purpose, Infrared (IR) camera-based thermal measurement setups have been deployed. As it is inevitable, in such setups, to remove the IR-opaque cooling (i.e. packaging, heat sink, etc.) during measurements, the majority of state-of-the-art employs a coolant oil to prevent the chip from overheating. The problem is that several aspects like thermal convection may interfere with the measured IR radiation resulting in equivocal IR images. Thus, the accuracy is decreased in a way that leads to incorrectly estimating reliability.

This thesis investigates the challenge of providing an abstracted, yet sufficiently accurate reliability estimation along with proposing novel techniques to increase the reliability of register files of embedded on-chip systems against aging effects and soft errors. It also introduces a novel thermal measurement setup that perspicuously captures the IR emissions of chips, i.e. without adding any layer on top of the measured chip.
The most important techniques, that are developed within this thesis, are as follows:

- Abstracting the various degradations induced by multiple simultaneous aging phenomena towards a probabilistic fault analysis that has a meaningful interpretation of reliability at the system level. Through the proposed abstraction, it becomes evident that targeting only one kind of failures induced by aging tied to a single aging phenomenon results in a non-negligible (up to 70%) reliability underestimation.

- Analysis of the aging stress that is induced by a wide range of applications shows that aging stress in different registers needs to be tackled using different strategies corresponding to their access patterns. For that purpose, a new technique called Register-Internal Stress Balancing (RISB) is proposed to selectively increase the resiliency of individual registers against aging effects.

- Register file analysis shows that some register bits are not continuously used to represent a value stored within a register. Based on that observation, a new technique called RESI is introduced that exploits the unused bits within registers to increase their resiliency against soft errors.

- Towards lucid IR thermography of processors, a novel Rear-side Thermoelectric-based IR Thermography (RAMA) technique is presented which was developed to protect multi-core chips during measurements. Unlike state-of-the-art, it allows the camera to perspicuously capture the IR emissions as no additional layer in between impedes the radiation.

Evaluations of the RISB and RESI techniques are conducted through simulations and partially through implementations on an FPGA platform. Additionally, the RAMA technique is applied to actual multi-core chips. On average, RISB increases the reliability of the register file with respect to aging effects by 19% compared to state-of-the-art. On the other hand, RESI increases the resiliency of the register file against Multiple Bit Upsets (MBUs) by 97% resulting in a high system fault coverage under various scenarios. The achieved result is 63% better compared to state-of-the-art. Finally, the investigation of the thermal characteristics of Intel 22 nm *octa-core* processor through our RAMA technique demonstrates how state-of-the-art inaccurate thermal analysis leads to incorrectly estimating reliability in multi facets.

Zusammenfassung der Arbeit

Technologische Fortschritte haben dazu geführt, dass eingebettete On-chip Systeme allgegenwärtig in unserem täglichen Leben zu finden sind. Allerdings steigen die Anforderungen an die Verlässlichkeit der Chips in dieser sogenannten nano-CMOS Ära schneller als in vergangenen Generationen. Die Skalierung der Integrationsgrößen erreicht die physikalischen Grenzen der korrekten Funktionsfähigkeit von On-chip Systemen. Diese Aspekte sind in der International Technology Roadmap for Semiconductors (ITRS) aufgezählt [25]. "Soft Errors" und Alterungseffekte sind hierbei besonders im Vordergrund, weshalb eine dringende Notwendigkeit entsteht, die Zuverlässigkeit von On-chip Systemen durch die Inangriffnahme dieser Effekte zu steigern.

Alterungseffekte werden beispielsweise durch unterschiedliche physikalische Phänomene wie "Negative Bias Temperature Instability (NBTI)", "Positive Bias Temperature Instability (PBTI)" und "Hot Carrier Induced Degradation (HCID)" verursacht. Diese Phänomene entstehen simultan innerhalb des Gate-Dielektrikums eines Transistors und verursachen eine Veränderung der elektrischen Eigenschaften des Transistors (z.B. der Spannungsschwellwert (V_{TH})). Insbesondere bei Strukturgrößen der sogenannten "deep nano"-Größe können auch kleine Änderungen großen Einfluss auf das Verhalten der Transistoren haben, was dazu führt, dass diese nicht mehr verlässlich arbeiten.

Zusätzlich wird die Verlässlichkeit von On-chip Systemen durch "Soft Errors" eingeschränkt, die durch hochenergetische Teilchenstrahlung wie z.B. durch Neutronenstrahlung verursacht werden. Diese können zum Beispiel zum "Umkippen" von Bits führen, was die Korrektheit von Anwendungen beeinträchtigen kann. Dies trifft insbesondere auf Schaltkreise zu, die bei niedrigen Spannungen betrieben werden, da die Differenz zwischen Versorgungsspannung und dem Spannungsschwellwert sinkt, was zu kleineren Sicherheitsabständen führt. Daher können bereits kleine, durch Partikel induzierte Spannungen, die Verlässlichkeit eines Transistors beeinträchtigen. Obwohl sowohl Alterungseffekte wie auch "Soft Errors" auf der physikalischen Ebene (nämlich in der Hardware) verursacht werden, werden ihr Auftreten sowohl örtlich wie auch zeitlich durch die Arbeitslast auf Systemebene (der Software) beeinflusst. Sie können sich durch die unterschiedlichen Ebenen von der physikalischen Ebene bis hoch zur Systemebene propagieren, wo sie sich schließlich in Fehlern manifestieren können, die das gesamte System unzuverlässig werden lassen. Die wachsende Anfälligkeit gegenüber Alterungseffekten sowie gegenüber "Soft Errors" führt dazu, dass der Registersatz als Teil von On-chip Systemen besonders im Vordergrund steht. Der Hauptgrund hierfür ist seine häufige Nutzung (auf ihn wird sehr häufig zugegriffen), weswegen sich durch Alterungseffekte und "Soft

Errors" induzierte Fehler von hier aus durch das gesamte On-chip System ausbreiten können. Daher ist es unerlässlich, die schädlichen Auswirkungen von Alterungseffekten und von "Soft Errors" innerhalb des Registersatzes anzugehen, damit ein zuverlässiger Betrieb von On-chip Systemen während ihrer gesamten Lebensdauer möglich ist.

Weiterhin hat das Skalieren der Integrationsgrößen zu einem Anwachsen der Energiedichte geführt, wodurch aktuelle und zukünftige Chips thermischen Beschränkungen durch den negativen Einfluss von erhöhter Temperatur auf die Verlässlichkeit des Systems unterliegen. Ein Beispiel hierfür ist die Tatsache, dass höhere Temperaturen zum schnelleren Altern von Transistoren führen. Daher müssen die thermischen Eigenschaften von Chips präzise erforscht werden, um ihre Verlässlichkeit korrekt einschätzen zu können. Dies ist auch die Voraussetzung, um Hardware/Software-Techniken mit Hinblick auf Verlässlichkeit zu Entwickeln. Zu diesem Zweck wurden Infrarotkamera-basierte (IR) Versuchsanordnungen eingesetzt. Da es für solche Versuchsanordnungen unerlässlich ist, IR-undurchlässige Kühlungen (z.B. das Packaging, den Kühlkörper, etc.) zu entfernen, benutzen die meisten Verfahren des Standes der Wissenschaft thermisch leitfähiges Öl, um das Überhitzen des Chips zu verhindern. Allerdings führt dies zu dem Problem, dass einige Aspekte wie thermische Konvektion die Messung der IR-Strahlung beeinflussen, was zu ungenauen IR-Aufnahmen führen kann. Dies verringert die Genauigkeit soweit, dass es zu unrichtigen Einschätzungen der Verlässlichkeit führt. Diese Dissertation erforscht die Herausforderung, eine abstrahierte und trotzdem ausreichend genaue Schätzung der Verlässlichkeit zu treffen, und schlägt zudem neuartige Techniken vor, um die Verlässlichkeit des Registersatzes von eingebetteten On-chip Systemen gegenüber Alterungseffekten und "Soft Errors" zu steigern. Zudem stellt sie eine neue Versuchsanordnung vor, um die IR-Emissionen von Chips deutlich aufzuzeichnen, nämlich ohne weitere Schichten, die den Chip bedecken.
Die wichtigsten Techniken, die innerhalb dieser Dissertation entwickelt werden, sind wie folgt:

- Die Abstraktion der verschiedenen Verschlechterungseffekte durch mehrere, simultan auftretende Alterungseffekte, hin zu einer probabilistischen Fehleranalyse, die eine bedeutsame Interpretation von Verlässlichkeit auf Systemebene zulässt. Durch die vorgeschlagenen Abstraktionen wird deutlich, dass das Angehen nur einer Art von Fehlern, die mit einem einzelnen Alterungseffekt einhergehen, zu einer nicht vernachlässigbaren Unterschätzung (bis zu 70%) von Verlässlichkeit führt.

- Eine Analyse der Belastung im Sinne von Alterungseffekten durch eine große Vielfalt von Anwendungen zeigt, dass diese Belastungen in unterschiedlichen Registern durch unterschiedliche Strategien angegangen werden müssen, die den einzelnen Zugriffsmustern entsprechen. Zu diesem Zweck wird eine neue Technik namens "Register-Internal Stress Balancing (RISB)" vorgestellt, die selektiv die Widerstandskraft einzelner Register gegenüber Alterungseffekten erhöht.

- Die Analyse des Registersatzes zeigt, dass einige Bits im Register nicht kontinuierlich benutzt werden, um Werte zu speichern. Basierend auf dieser Beobachtung wird eine neue Technik namens "Register-Embedded Self-Immunity (RESI)"

vorgestellt, die die ungenutzten Bits verwendet, um innerhalb der Register die Widerstandskraft gegenüber "Soft Errors" zu erhöhen.

- Eine neuartige "Rear-side Thermoelastic-based IR Thermography (RAMA)" wird vorgestellt, um den Schutz von Prozessoren bei klarer IR Thermographie zu ermöglichen. Im Gegensatz zum Stand der Wissenschaft erlaubt die Kamera deutliche Aufnahmen der IR-Emissionen, da keine weiteren Schichten oberhalb des Chips das Bild beeinflussen.

Die Evaluation der RISB und RESI-Techniken werden mit Hilfe von Simulationen und partiell mit Implementierung auf einer FPGA-Plattform durchgeführt. Die RAMA-Technik wird zusätzlich auf realen Mehrkern-Chips verwendet. Im Schnitt kann RESI die Verlässlichkeit des Registersatzes gegenüber Alterungseffekten zu 19% gegenüber dem Stand der Wissenschaft verbessern. Zusätzlich steigert RESI die Widerstandskraft gegenüber sogenannten "Multiple Bit Upsets (MBUs)" zu 97%, was zu einer hohen Abdeckung von Fehlern in verschiedensten Szenarien führt. Abschließend zeigen die Untersuchungen der thermischen Eigenschaften von Intel 22nm *octa-core* Prozessoren durch unsere RAMA-Technik, wie die ungenaue thermische Analyse des Standes der Wissenschaft zu einer Fehleinschätzung von Verlässlichkeit in vielerlei Hinsicht führt.

The Big Picture behind the Thesis

The Chair for Embedded Systems (CES) at the Karlsruhe Institute of Technology (KIT) has its core expertise in design and architectures for reliable embedded systems. Major research projects deal/dealt with generating reliable software on unreliable hardware components (e.g., [CES1]), building reliable reconfigurable architectures (e.g., [CES2]) and developing low-power design of multimedia systems based on dynamic power and thermal managements (e.g., [CES5]). In addition, several works (e.g., [CES3, CES4]) focused on the Dark Silicon problem where only a subset of processing cores within an on-chip system may operate at full performance at the same time due to power and thermal constraints.

Furthermore, the CES played an essential role in the creation of the Priority Program (*Schwerpunktprogramm*, SPP 1500) and "*Invasive Computing*" (InvasIC):

DFG SPP 1500 "Dependable Embedded Systems"

In fact, the key reason behind the significant focus on reliability at CES is the German national research program "Design and Architectures for Dependable Embedded Systems" (German Research Foundation, DFG SPP 1500) that aims to find new means to overcome the inherent challenges within the nano-CMOS era [CES6]. Actually, since feature sizes of transistors began to approach atomic levels, technology scaling has entered an inflection point, where optimizing on-chip systems for reliability is at least as important as optimizing them for performance and cost.

FIGURE 1: The pyramid for dependable embedded systems [CES6]

This new paradigm pushed diverse dependability concerns to the forefront and drew the attention of several research groups in Germany to increasingly focus on design and architectures for dependable embedded systems which are the essence of this priority program.

The SPP 1500 consists of diverse individual projects throughout Germany, working together to deal with the various dependability challenges within the current and upcoming technology nodes [CES6]. One of the contributions of the CES is the VirTherm-3D project, which is in collaboration with TU München (TUM). The key focus of this project is to increase the dependability of future 3D-layered architectures with a special focus on the thermal and aging challenges. The SPP 1500 also cooperates with NSF Variability Expedition in USA which, in turn, focuses on developing variability-aware software for nano-scale devices.

The main challenges that the DFG SPP 1500 Priority Program addresses are [CES6]:

- Design-Time Effects: As transistor feature sizes scale down, the process of manufacturing starts to follow statistical variance and thus identical transistors may have very different electrical characteristics.
- Run-Time Effects: As a matter of fact, transistors increasingly become more susceptible to *aging effects* as we move to smaller technology nodes with each new generation. In addition, the increase in on-chip power densities increases *on-chip temperatures* and thus transistors become under severe thermal stress that makes them over time unreliably behave. Finally, the susceptibility of memory and logic circuits to *soft errors* also keeps increasing due to the lower operating voltages.

The main goals of the DFG SPP 1500 Priority Program are [CES6]:

- Technology Abstraction.
- Dependable Hardware Architectures.
- Dependable Embedded Software.
- Operation, Observation, and Adaptation.
- Design Methodologies.

The Thesis Contributions in the scope of the DFG SPP 1500:

As earlier mentioned in the Abstract, this thesis focuses on increasing the reliability of on-chip systems with respect to aging effects, soft errors and temperature. In fact, these three objectives are indeed an integral part of the key challenges that DFG SPP 1500 Priority Program addresses. For instance, the proposed aging and soft errors techniques within this thesis (see Chapters (4, 5)) contribute to the goal of "Dependable Hardware Architectures". Additionally, the proposed reliability estimation/abstraction (see Chapter 3) contributes to the goal of "Technology Abstraction". Finally, the infrared-camera based thermal measurement setup that was developed in the scope of this thesis (see Chapter 7) contributes to the goal of "Observation" since it provides designers with an accurate baseline to compare against as well as it enables them to obtain real-time observations of the thermal behavior of their multi-core systems.

DFG Transregional Collaborative Research Center (TCRC) 89 "*Invasive Computing*" (InvasIC)

In addition to the DFG SPP 1500 Priority Program, another ambitious project is currently running at the CES which is known as "*Invasive Computing*" (InvasIC). It is a large collaboration project between the Karlsruhe Institute of Technology (KIT), Technische Universität München and Friedrich Alexander University in Erlangen (FAU). This project focuses on addressing the upcoming challenges in the future many-core on-chip systems such as self-adaptive and resource-aware programming through providing new means and concepts regarding programming languages, operating systems and compilers [CES6]. The projects cover a wide range of challenges from the architecture level to the compiler level through the operating-system level. An example of the challenges that InvasIC aims to address is dealing with the dark silicon problem [CES3, CES4] which may be a key problem in the future many-core systems.

[CES1] S. Rehman, M. Shafique, F. Kriebel, J. Henkel. "Reliable Software for Unreliable Hardware: Embedded Code Generation aiming at Reliability", in IEEE International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS), Taipei, Taiwan, pp. 237-246, 2011.

[CES2] H. Zhang, M.A. Kochte, M.E. Imhof, L. Bauer, H.-J. Wunderlich, J. Henkel, "GUARD: GUAranteed Reliability in Dynamically Reconfigurable Systems", in ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, Jun, pp. 1-6, 2014.

[CES3] S. Pagani, H. Khdr, W. Munawar, J. Chen, M. Shafique, M. Li, J. Henkel, "TSP: Thermal Safe Power - Efficient Power Budgeting for Many-Core Systems in Dark Silicon", in IEEE International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS), New Delhi, India, pp. 1-10, 2014.

[CES4] M. Shafique, S. Garg, J. Henkel, D. Marculescu, "The EDA Challenges in the Dark Silicon Era: Temperature, Reliability, and Variability Perspectives," in ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, pp.1-6, 2014.

[CES5] D. Palomino, M. Shafique, A. Susin, and J. Henkel, "TONE: adaptive temperature optimization for the next generation video encoders," In International Symposium on Low Power Electronics and Design (ISLPED), pp. 38-33, 2014.

[CES6] J. Henkel, L. Bauer, J. Becker, O. Bringmann, U. Brinkschulte, S. Chakraborty, M. Engel, R. Ernst, H. Hartig, L. Hedrich, A. Herkersdorf, R. Kapitza, D. Lohmann, P. Marwedel, M. Platzner, W. Rosenstiel, U. Schlichtmann, O. Spinczyk, M. Tahoori, J. Teich, N. When, H. Wunderlich "Design and architectures for dependable embedded systems," in IEEE International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS), pp. 69-78, 2011.

[CES6] J. Henkel, A. Herkersdorf, L. Bauer, T. Wild, M. Huebner, R. K. Pujari, A. Grudnitsky, J. Heisswolf, A. Zaib, B. Vogel, V. Lari, and S. Kobbe, "Invasive Manycore Architectures," in ASP-DAC, pp. 193-200, 2012.

Chapter 1

Introduction

Since the invention of Integrated Circuit (IC), IC chips manufacturing remarkably followed a continued trend of becoming denser and more complex akin to Moore's law that predicted a shrinking in the dimensions of transistors roughly by half every two years [33] in order to gain more advances in several aspects like performance. However, this trend along with the relentless demand for reducing costs caused, with each new generation, a noticeable decrease in the reliability of on-chip systems due to a wide range of aspects. Even though the majority of these aspects had been known for a long time, their deleterious impact had not reached the crucial point where they would jeopardize the reliability of an entire on-chip system. An inflection point occurred around a decade ago while entering the nano-CMOS era. At that time, technology roadmaps provided evidences that the aggressive scaling of technology nodes would steadily increase the susceptibility of chips to varied kinds of failures [34] – especially in deep nano scale (i.e. $\leq 45\,\text{nm}$). This makes current and upcoming on-chip systems constrained by reliability as maintaining the correct functionality of them during their lifetime is fundamental and cannot be compromised. Therefore, designers consider reliability as one of the major design challenges and it is expected that conventional constraints such as cost and performance may be overtaken by reliability and lifetime concerns [35].

The reliability can be defined as the ability of a system or component to function under stated conditions for a specified period of time [31] and the key sources of failures that jeopardize the on-chip systems reliability may be categorized as follows:

- *Time-Dependent Failures:* They are dependent on the on-chip system lifetime and mainly because of degradations due to aging phenomena that are able to change, over runtime, the electrical characteristics of transistors within the on-chip system. Such changes make the transistors no longer reliably operate and the most observed is V_{TH} shift which is often induced by the Bias Temperature Instability (BTI) aging phenomenon [36].
- *Time-Independent Failures:* They are independent of the on-chip system lifetime and mainly because of manufacturing variability and soft errors. Manufacturing variability makes transistors with identical specifications have different electrical

characteristics and hence they may unreliably operate. On the other hand, soft errors – due to e.g., energetic particles – may transiently result in spurious bit flips in the underlying hardware (Single Bit Upsets (SBUs) or MBUs) that may later lead to failures at the system level according to when and where they occurred.

It is noteworthy that the aforementioned sources of failures are *interrelated* with each other. For instance, while on the one hand aging-induced degradations in the transistors electrical characteristics increase the susceptibility of them to soft errors, manufacturing variability, on the other hand, randomly influences the transistors and thus it makes them unevenly affected (i.e. degraded) by aging effects over their lifetime.

1.1 Register Files of Microprocessors

A Register File (*RF*) is a small set of memory cells inside the microprocessor. It consists of N registers each capable of storing a word of K bits. For instance, the MIPS architecture has ($N = 32$ registers, $K = 32$ bits) [37] and the Alpha architecture has ($N \geq 80$ registers, $K = 64$ bits) [38]. The Central Processing Unit (CPU) uses the register file for storing operands of instructions and, additionally, it may also store special-purpose registers (e.g., Program Counter (*PC*)). Therefore, the register file is very frequently accessed [39] during the execution of an application.

The intensive utilization of the *RF* makes it one of the essential microarchitectural components with respect to reliability. This is due to the fact that faults within the *RF* can be deleterious as they may spread from there throughout the entire on-chip system leading to different serious problems including CPU crashing. An analysis presented in [40] showed that a considerable amount of faults, that disturb the microprocessor's reliability, often stems from the *RF*. This makes increasing the *RF*'s resiliency against aging effects and soft errors – as both of them are able to induce faults within the *RF*'s cells – imperative to maintain reliability.

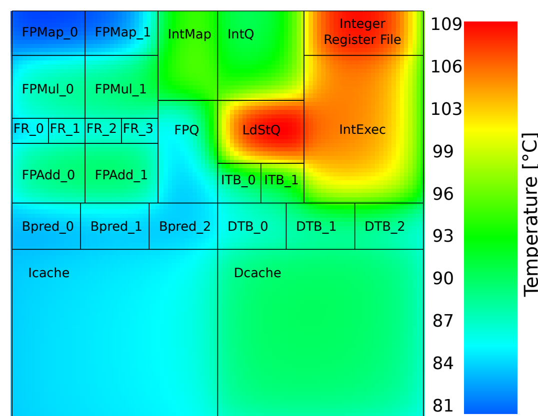


FIGURE 1.1: Simulated steady-state thermal map of the Alpha CPU showing that the register file component has an elevated temperature

RF component noticeably contributes to the total CPU power. For instance, on the Intel 32 nm Westmere Core [41], the *RF* accounted for 30% of leakage and dynamic power on a typical benchmark [42]. Additionally, the overall area footprint of the *RF* is rather small (e.g., only around 6% of the total area of an Alpha CPU [43]). The latter along with the higher power consumption makes the power density of the *RF* relatively high compared to other CPU components [44]. In turn, the high power per area directly produces elevated on-chip temperatures and therefore the temperatures at the register

file are often high [44, 45]. This observation in literature is consistent with our thermal simulations of the Alpha CPU at the 22 nm as Figure 1.1 shows for the case of running the x264 benchmark [28] on top of the Linux operating system. Additionally, the real (i.e. not simulated) IR-based thermal measurements presented in [46] also showed that the thermal hotspot is often located in the register file component which achieves the highest average temperature.

On the other hand, elevated on-chip temperatures strongly disturb the reliability of on-chip systems as they, for instance, affect the key parameters of circuits (e.g., delays, noise resiliency, etc.) increasing their susceptibility to failures due to timing violations and/or data corruption. Importantly, higher temperatures stimulate aging phenomena and thus they contribute to higher aging-induced reliability degradations (e.g., higher shift in the transistors V_{TH}). Further details regarding the impact of temperature on reliability will be presented in Section 1.5.

All in all, register files are considered one of the most susceptible CPU components to failures and therefore increasing their resiliency against aging effects and soft errors is inevitable for embedded on-chip systems where the reliability cannot be easily compromised.

1.2 SRAM Cells

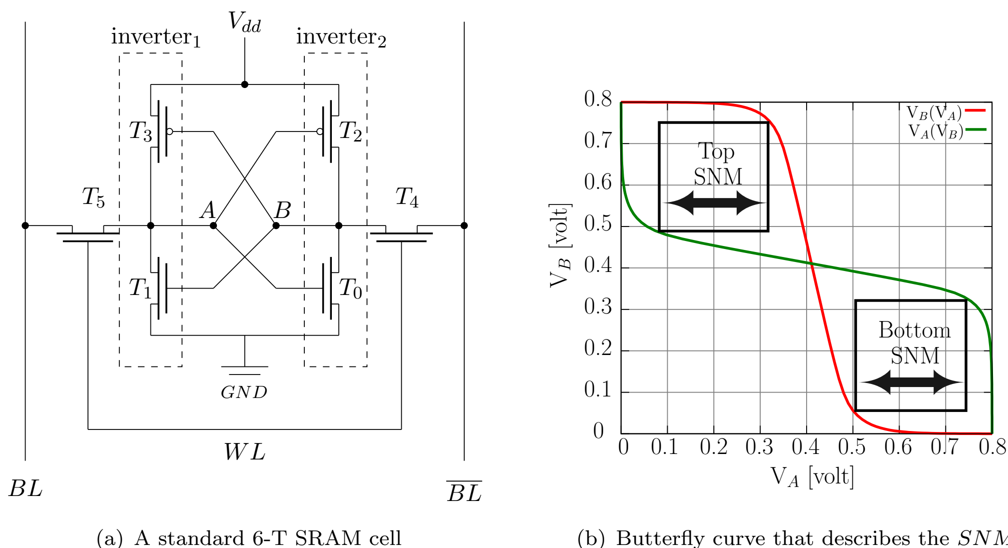
Static Random-Access Memory (SRAM) cells are often used to implement RFs in microprocessors [47–49]. For instance, Intel employs an SRAM-based register file in their commercial 32 nm Westmere Core [41]. This is because their ultra speed and regular structure: SRAM cells are typically implemented using smaller transistor sizing than computational logic of the same technology [50].

A standard 6-T SRAM cell consists of six transistors as illustrated in Figure 1.2(a): Four transistors (T_0, T_1, T_2, T_3), which form two cross-coupled inverters and, additionally, two access transistors (T_4, T_5) to drive the SRAM's read/write accesses. The cross-coupled pair of inverters maintains the continuous storage of complementary bits within the 6-T SRAM cell during its operation.

In the following, we briefly discuss the key metrics that describe the reliability of SRAM cells and later we investigate how temperature and/or aging effects degrade these metrics.

1.2.1 Static Noise Margin

The Static Noise Margin (SNM) of an SRAM expresses its resiliency against voltage noise which is primarily due to the effects of parasitic-coupling (e.g., bit-line coupling) and intrinsic noise sources (e.g., thermal noise, random telegraph noise, etc.). Such a voltage noise can jeopardize the data integrity of an SRAM cell depending on its SNM (i.e. larger SNM results in higher resiliency against noise and vice versa).



(a) A standard 6-T SRAM cell

(b) Butterfly curve that describes the SNM

FIGURE 1.2: A standard 6-T SRAM cell along with the static noise margin (SNM) that represents its resiliency against noise

The SNM can be measured from the so-called *butterfly* curve that describes the transfer characteristics of the SRAM cell with the SNM equals to the length of the side of the smallest square located between the two curves as demonstrated in Figure 1.2(b). There, A and B represent the outputs of the cross-coupled inverters of the 6-T SRAM cell (see Figure 1.2(a)). $V_{\mathrm{A}}(V_{\mathrm{B}})$ plots the output voltage resulting from a sweep of a voltage in B from 0 to V_{dd}. $V_{\mathrm{B}}(V_{\mathrm{A}})$ is the voltage in B when V_{A} sweeps from 0 to V_{dd} (i.e. from 0 V to 0.8 V in our shown example in Figure 1.2(b)).

1.2.2 Read Access Time

Read Access Time (RAT) of an SRAM cell represents the required time until the SRAM provides its stored data on its bit lines (i.e. BL and $\overline{BL}$). For instance, during a read operation both BL and $\overline{BL}$ are pre-charged to V_{dd} and the SRAM pulls one of the bit lines down to Ground (GND).

An important concept here is the threshold of the sense amplifier of the SRAM cell. In the case of a threshold of $\frac{V_{dd}}{2}$, for instance, the SRAM sense amplifier will be activated as soon as the $BL \leq \frac{V_{dd}}{2}$ and then it amplifies the difference in potential between the bit lines (i.e. $V(BL) - V(\overline{BL}) = \frac{V_{dd}}{2} - V_{dd} = -\frac{V_{dd}}{2}$) to full V_{dd} which forces BL to the GND potential and $\overline{BL}$ to the V_{dd} potential.

It is worthy to note that the aforementioned explanation of RAT is for the case of reading a stored logic '0'. However, for the opposite case of reading a stored logic '1', the explanation would be analogous with $\overline{BL}$ instead of BL.

1.2.3 Critical Charge

The Critical Charge (Q_{crit}) of an SRAM cell is the minimal amount of charges which needs to be induced within the SRAM cell to corrupt its stored data. Similar to the SNM that represents the SRAM resiliency against *intrinsic* noise, Q_{crit} represents the SRAM resiliency against *extrinsic* particles-induced noise (i.e. against the deposition of kinetic energy of a particle when it strikes one of the SRAM cell transistors).

1.3 Aging Effects

Continuous shrinking in feature sizes results in both elevated electric field strength as well as current density which make transistors in the nano-CMOS era suffer more from degradations induced by aging phenomena. Of these phenomena, NBTI, PBTI and HCID have become the most dominant in impeding reliable transistor operation over its lifetime [51, 52]. Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) operates through switching the electric fields on and off which poses a stress for the materials within the transistor leading to the formation of defects there [53]. Such defects are imperfections within the lattice of the material, i.e. unsatisfied bonds due to missing atoms. Ultimately, aging-induced defects degrade the electrical characteristics of the transistor (e.g., V_{TH}, Carrier Mobility (μ), Drain Current (I_D), etc.).

While understanding the physical processes of aging phenomena is not entirely required at the system level, there is indeed still a substantial need to analyze their ability to induce degradations in order to accurately estimate reliability – *this holds even more when multiple aging phenomena interdepend, i.e. when they interact with each other.* As a matter of fact, these phenomena simultaneously occur in the gate dielectric of a transistor and thus the physical processes behind them interact with each other. Therefore, estimating correctly the overall impact of aging phenomena on the electrical characteristics of transistors (e.g., V_{TH}, μ, etc.) necessitates investigating aging phenomena *simultaneously*, as the focus of this thesis, and not *individually* as state-of-the-art does (detailed discussion will be later presented in Chapter 3).

1.3.1 Aging phenomena

NBTI/PBTI: It is the defect generation due to the electric field over the gate dielectric (i.e. the Si−SiO_2 interface) when a PMOS/NMOS transistor is operated, as Figure 1.4(c) shows. These defects cause charge buildup within the gate dielectric and thereby they weaken the electric field resulting in degrading the electrical characteristics of the transistor over time [52].

HCID: It is the defect generation near the drain due to *hot carriers* – whose high kinetic energies are sufficiently significant to form electron-hole pairs through impact of ionization – that are injected in undesirable areas of the transistor. As a matter of fact, the strong electric field across the channel of a transistor is the key source behind such

hot carriers [54]. It is worthy to note that *hot* does not refer to transistor's temperature but to the ability to tunnel out of the semiconductor material. Over time, HCID causes charge buildup near the drain of a transistor (see Figure 1.4(d)) and thus similarly to BTI results in degrading the electrical characteristics of transistors [55].

Unlike NBTI/PBTI which is more pronounced in PMOS/NMOS transistors and can be recovered, the recovery in HCID is negligible and its mechanism degrades both NMOS as well as PMOS transistors. When operating a transistor both of the vertical and lateral electric fields may be generated (see Figure 1.4(d)) and thus both BTI and HCID phenomena *simultaneously* occur. The induced defects by both of them jointly degrade the electrical characteristics of the transistor which, in turn may cause malfunctions within the on-chip system during its lifetime.

In general, the overall degradations in the electrical characteristics of transistors are mainly determined through the processes of defect generation/recovery and electrical activation/deactivation of defects akin to the the electric fields, either across the channel or over the gate dielectric. Such processes are *driven* by both operating conditions such as temperature and stress/relaxation time ration as well as circuit design parameters such as the operating voltage (V_{dd}). Additionally, the employed materials in manufacturing the transistors (e.g., SiON/poly−Si, SiO_2/HfO_2/TiN, etc.) play an important role in defining the strength of occurring aging phenomena (i.e. the amount of generated defects due to BTI and HCID over time).

1.3.2 Impact of Technology Scaling

Before the nano-CMOS era, semiconductor fabrication used to employ the same scaling factor of both supply voltage (S_V) and transistor length (S_L) in order to maintain that the electric fields across the channel remain almost constant [56] after the scaling to the next technology node, as shown in Eq 1.1. This is known as Dennard's scaling [22]. However, the technological pressures on device designers to aggressively reduce transistor geometries towards integrating more transistors per chip run counter to the need of maintaining a similar scaling factor. This is mainly due to the fact that V_{dd} scaling is reaching its limits as it would be below or near V_{TH}. As a matter of fact, operating transistors in such regions result in running the circuits at significantly lower performance, which may not be applicable in many of nowadays on-chip systems.

In other words, technology scaling in the nano-CMOS era has changed the rule of scaling from ($S_L = S_V$) to ($S_L < S_V$) which, in turn, made the electric field across the channel becomes much stronger/higher as shown in Eq 1.1 and illustrated in Figure 1.3.

$$E_{new}(Channel) = \frac{V_{dd} \cdot S_V}{L \cdot S_L}; \tag{1.1}$$

$$S_L = S_V \Rightarrow E_{new}(Channel) = E_{old}(Channel) S_L < S_V \Rightarrow E_{new}(Channel) > E_{old}(Channel)$$

Similar to the electric field across the channel, the electric field over the gate also depends on the supply voltage and thus it will also increase due to the aforementioned reason: $E_{new}(Gate) > E_{old}(Gate)$.

As earlier explained, the induced defects by the aging phenomena depend on the electric fields, either across the channel (i.e. HCID) or over the gate (i.e. BTI). Therefore, with higher fields akin to scaling in the nano-CMOS era, aging phenomena become stronger and more severe resulting in higher degradations in the transistors electrical characteristics.

In short, with each new technology node, insufficient voltage scaling leads to magnifying the deleterious impact of aging and thus jeopardizing more the reliability on-chip systems. This, in turn, illustrates the inevitable need of developing new techniques to estimate and increase the reliability of critical components of embedded on-chip systems (e.g., register files) with respect to aging effects as the focus of this thesis.

Finally, as technology nodes were being reduced to below 45 nm, the insulating quality of the gate dielectric, additionally, became much less and thus leakage currents due to tunneling significantly increased. This was because continuous scaling in the transistors geometries made the used layers thinner and thinner. To solve this problem, manufactures replaced silicon dioxide, that has been conventionally employed to form the gate dielectric, with a material that has a higher dielectric constant K like hafnium-based dielectric layer in Intel [24], as shown in Figures(1.4(a) and 1.4(b)). While the employment of such new materials (i.e. high-K materials) was necessary to maintain the desired electrical characteristics of transistors, the susceptibility of transistors to aging effects became, unfortunately, much higher [57, 58] due the new employed materials (e.g., HfO_2 exhibits less resiliency to aging effects than SiO_2).

As a matter of fact, scaling in conjunction with high-K materials has made aging phenomena, that have often been assumed to be negligible (e.g., PBTI in NMOS), become

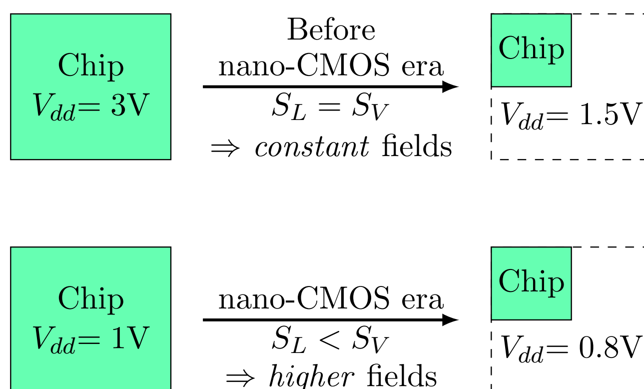
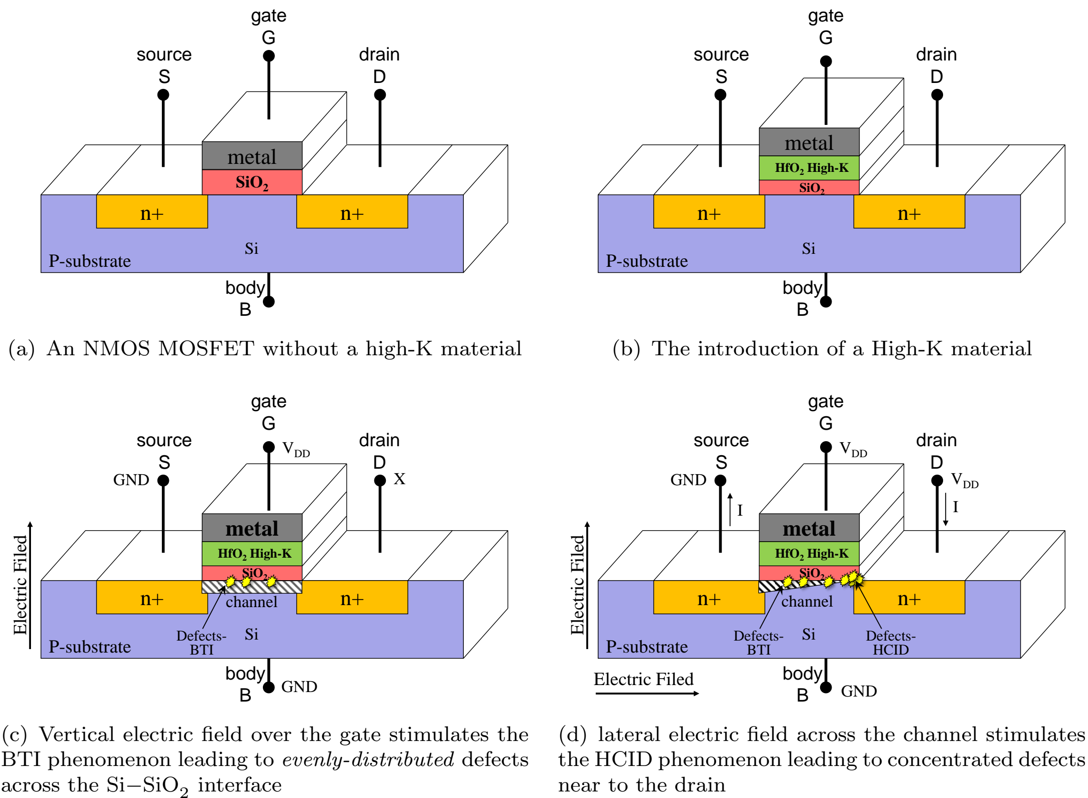


FIGURE 1.3: Technology scaling results in constant electric fields when both area and V_{dd} scale with the same factor (e.g., $S_L = S_V = 0.5 \Rightarrow E_{new}(Channel) = E_{old}(Channel)$). However, scaling within the nano-CMOS era results in higher fields as area and V_{dd} are not scaled anymore with the same factor (e.g., $S_L = 0.5$, $S_V = 0.8$ $\Rightarrow S_L < S_V \Rightarrow E_{new}(Channel) > E_{old}(Channel) \Rightarrow$ more induced defects by aging and thus higher reliability degradations)

noticeable [57]. The measurements in [57] established that the employment of the new high-K material results in considerably stronger degradation regarding PBTI and a similar behavior regarding NBTI.



(a) An NMOS MOSFET without a high-K material

(b) The introduction of a High-K material

(c) Vertical electric field over the gate stimulates the BTI phenomenon leading to *evenly-distributed* defects across the $Si-SiO_2$ interface

(d) lateral electric field across the channel stimulates the HCID phenomenon leading to concentrated defects near to the drain

FIGURE 1.4: Impact of scaling on a transistor

1.4 Soft Errors

They may be generated by radiation and they are recognized as a key external source of failures in on-chip systems. When particles like neutrons strike the silicon substrate, the induced radiation from the collisions of them with nuclei in the substrate liberates a shower of charged particles which then leave an ionization trail [59]. If the resulting deposited charge is sufficient and depending on the *spatial/temporal* susceptibility of the circuit (i.e where/when the particle strikes), the incident particle-induced current spike, which manifests itself as an extrinsic noise, results in a soft error.

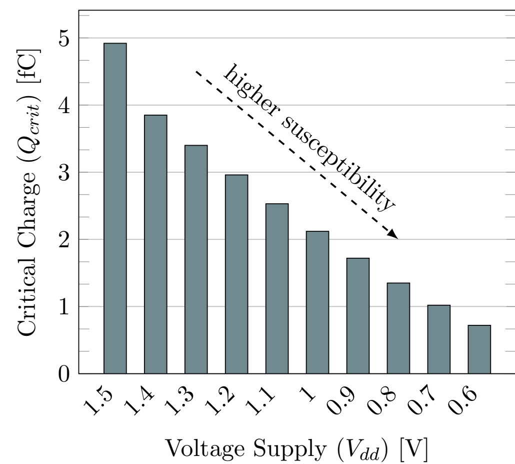


FIGURE 1.5: Impact of voltage scaling on increasing the susceptibility to soft errors

In the past, making IC operating outside the earth's atmosphere to withstand soft errors was primarily a reliability concern [60]. However, through the relentless pursuit of scaling down along with lowering the operating voltage, soft errors have also emerged as an important threat to circuits even in terrestrial applications [61] and Soft Error Rate (SER) is dominated by low-energy neutrons [62].

We clarify in Figure 1.5 the impact of voltage scaling on Q_{crit} of a 22 nm SRAM cell. As it can be noticed, Q_{crit} noticeably decreases with lower voltage which, indeed, increases the SRAM susceptibility to soft errors. Additionally, we also examine the corresponding

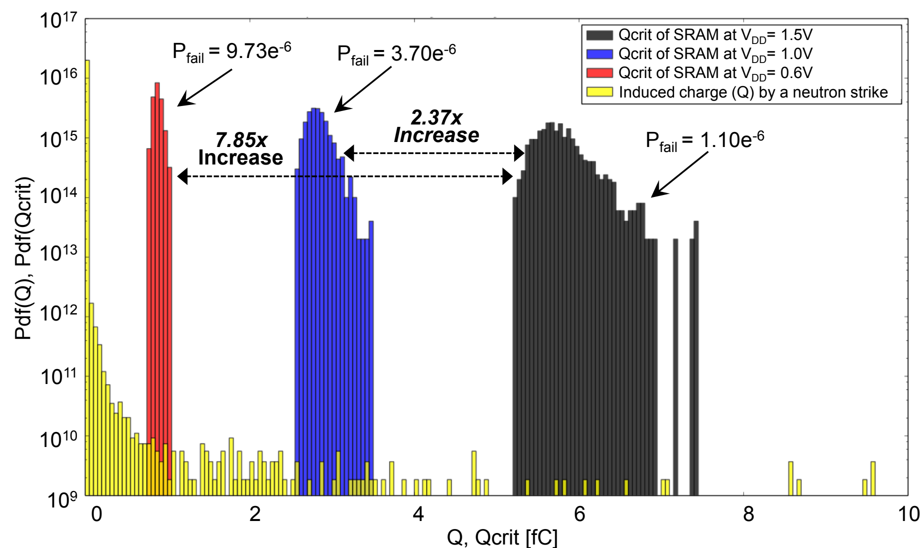


FIGURE 1.6: The SRAM susceptibility to radiation due to different operating voltage showing how lower V_{dd} significantly increases the probability of failure (P_{fail}) in 22 nm SRAM cells due to soft errors. Therefore, increasing the resiliency to soft errors in low-power embedded on-chip systems is indispensable

probability of failure (P_{fail}) in around 1000 SRAM cells[1] due to different operating voltages. Details regarding the reliability estimation to conduct the aforementioned results and other results within this chapter will later be presented in Chapter 3.

As it can be noticed from Figure 1.5, scaling the V_{dd} from 1.5 V to 1.0 V and 0.6 V reduces the Q_{crit} of SRAM by 1.32x and 5.83x, respectively. This, in turn, can be translated to an increase in the SRAM P_{fail} by 2.4x and 7.8x, respectively, as Figure 1.6 shows.

This clarifies the importance of increasing the reliability of SRAM-based CPU components, such as register files, against soft errors – especially in embedded on-chip systems where the need for low-power designs and thus scaling V_{dd} is dominant.

Finally, we present in Figure 1.7 a general overview of the different abstraction levels that need to be tackled in the scope of increasing the register file reliability. As a matter of fact, running workloads on top of the embedded on-chip system (i.e at the system level) *derive* the reliability degradations that start from the physical level and then propagate all the was up to the system level, where they manifest themselves as failures. Therefore, the *interdependencies* between the system and physical levels do matter and they need to be considered when estimating and/or increasing the reliability of on-chip system are targeted.

[1] A normal distribution with a standard deviation of $\sigma = 1.6$ has been considered to model the impact of manufacturing variability on the transistors geometries. The simulations of the 6-T SRAM cells have been performed under the assumption that each SRAM cell constantly stores the logic value '0'.

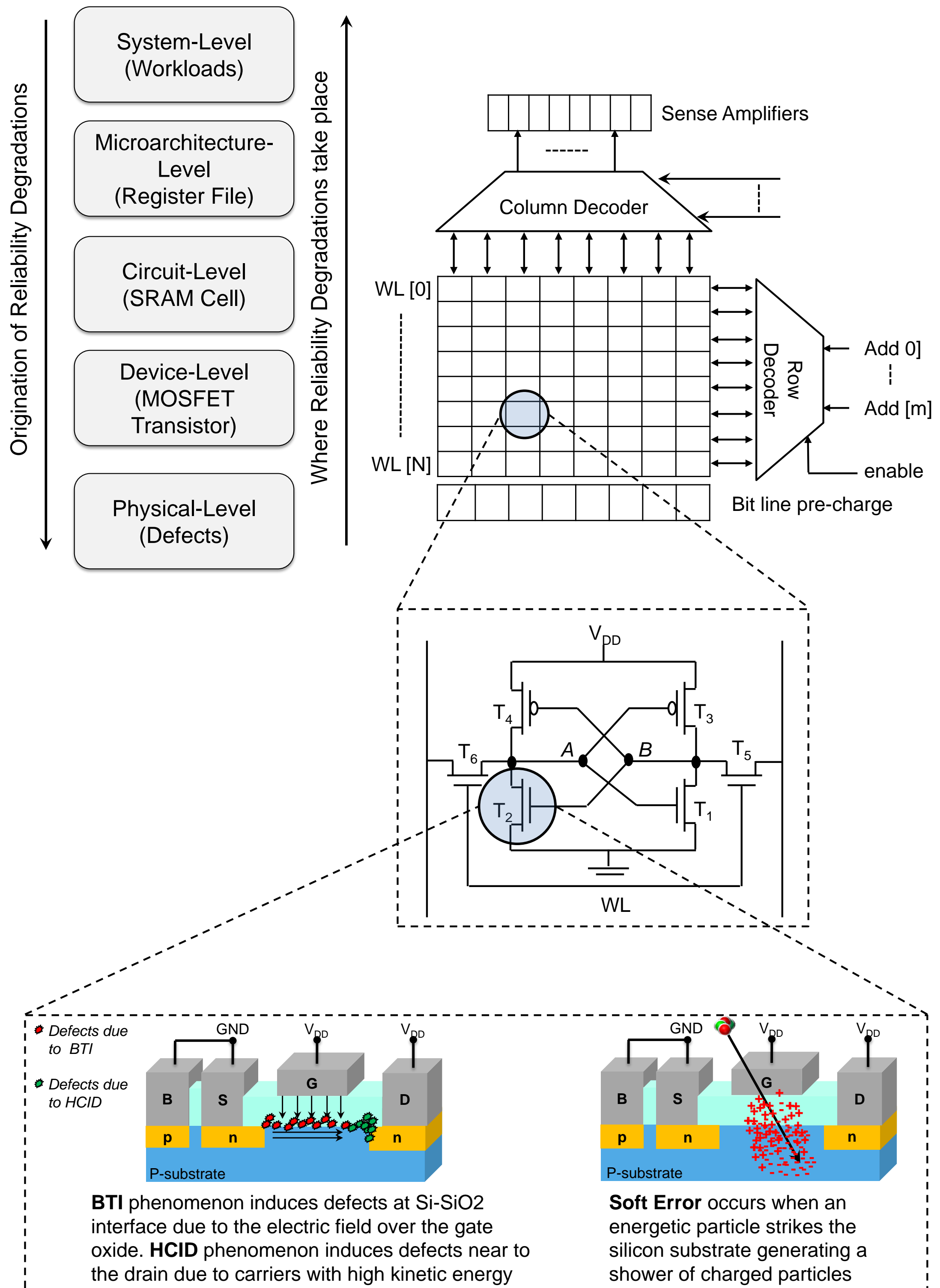


FIGURE 1.7: Overview of the register file reliability with respect to aging effects and soft errors

1.5 On Temperature-related Reliability

Advances in technology scaling in the nano-CMOS era have steadily increased on-chip power densities which are the key reason for unsustainable on-chip temperatures. The trend of technology scaling, with supply voltage scaling reaching its limits, leads to a discontinuation of Dennard Scaling [23], as motivated in Figure 1.3. This causes so-called of *dark silicon* where it mandates that only a subset of cores within an on-chip system can be powered at full performance during the same time. Otherwise, the thermal constraints of chip (e.g., Thermal Design Power (TDP)) cannot be fulfilled [63]. Furthermore, leakage power is directly related to the junction temperature. Hence, elevated temperatures rapidly lead to higher leakage power consumption.

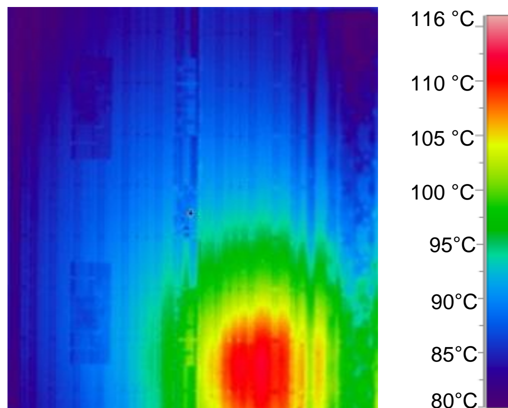


FIGURE 1.8: IR image of a chip that violates thermal constrains resulting in *short-term* and *long-term* effects with respect to reliability

As a matter of fact, elevated on-chip temperatures not only increase the cooling costs but they additionally jeopardize the chip's reliability akin to degradations in the circuit's key metrics, i.e. *short-terms* effects such as delay increase as well as *long-terms* effects such as aging-induced V_{TH} increase.

1.5.1 *Short-terms* Effects of Temperature on Reliability

In Figure 1.9, we conducted around 1000 simulations for 22 nm 6-T SRAM operating at V_{dd}=1.2 V for the purpose of analyzing the impact of temperature increase on the SRAM reliability. As it can be seen in Figure 1.9(a), the temperature rise shifts the SNM distribution to the left side resulting in lower resiliency against noise. A similar behavior can also be observed in Figure 1.9(b) which shows how a temperature increase noticeably makes the SRAMs more susceptible to radiation due to decreasing their Q_{crit}. In Figure 1.9(c), the temperature rise shifts the RAT distribution to the right side making SRAMs slower and thus more susceptible to failures due to timing violations.

For further understanding of the impact of temperature on the susceptibility to soft errors, we present in Figure 1.10 the probability of failure due to different temperatures. As it can be noticed there, elevated temperatures such as $60°C$ and $125°C$ increase the probability of failure (P_{fail}) with 1.12x and 3.55x, respectively, compared to the low temperature of $25°C$.

It is noteworthy, here, that even though operating the chip at lower voltages effectively reduces its temperature, voltage scaling concurrently makes the difference between V_{dd} and V_{TH} considerably smaller resulting in higher probability of failure, as it has been clarified in Figure 1.6 with respect to soft errors. More importantly, such an increase

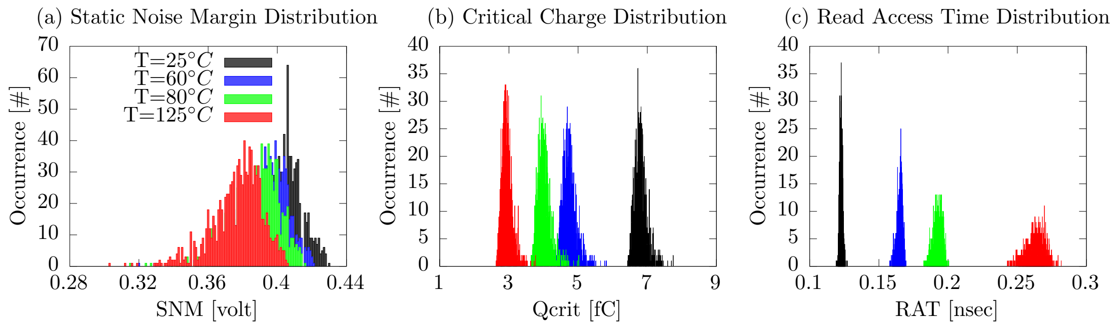


FIGURE 1.9: Impact of *short-term* effects of temperature on degrading the key reliability metrics of SRAM cells. The same voltage of 1.2 V has been used for the three different scenarios for a fair comparison. The shifts in SNM and Q_{crit} histograms (a and b) towards left indicate less reliability. Whereas, shifts in RAT histogram (c) towards right indicate less reliability

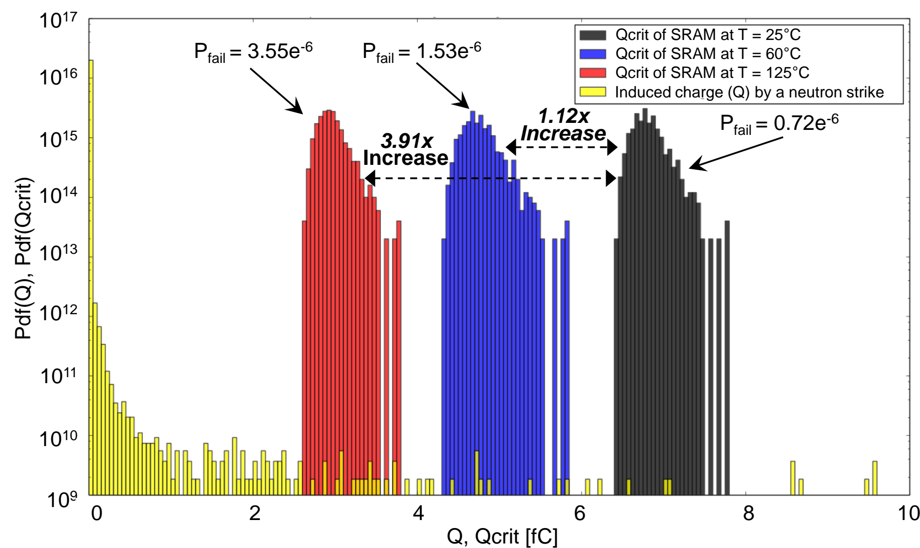


FIGURE 1.10: Impact of elevated temperatures on increasing the susceptibility of SRAM cells to soft errors. The same voltage of 1.2 V has been used for the three different scenarios for a fair comparison. Higher temperatures noticeably increase the probability of failure due to soft errors

in the probability of failure can compensate the potential gain, in terms of reliability improvement, that might come from lowering the temperature.

In a sense voltage scaling is a double-edged sword: While on the one hand a lower voltage reduces the on-chip temperatures and thus increases the reliability (see Figures (1.9 and 1.10)), it magnifies, on the other hand, the circuits susceptibility to failures due to, for instance, soft errors (see Figures (1.5 and 1.6)). To elaborate the aforementioned conflict, we present in Figure 1.11 the SER for different temperatures along with different voltages. As it can be observed from the dashed line, even if scaling the V_{dd} from 1.2 V to 1.0 V could provide a temperature reduction from 125°C to 80°C, the increase in susceptibility to soft errors, due to the lower V_{dd}, compensates the positive impact that the lower temperature has on SER.

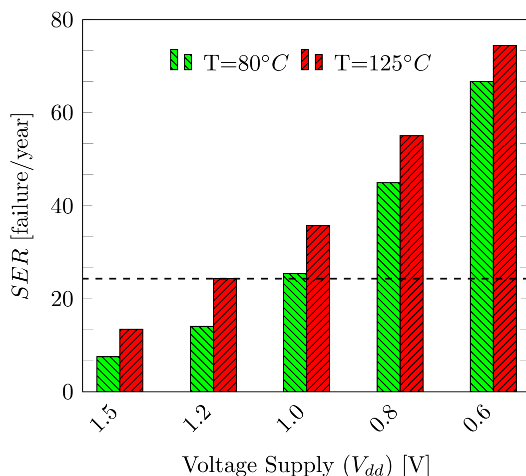


FIGURE 1.11: Evaluating the impact of different operating conditions of operating voltage (V_{dd}) and temperature (T) on the soft-error rate (SER) of 22 nm SRAM cells showing how voltage scaling is a double-edged sword with respect to reliability

In other words, operating conditions of (T=125°C, V_{dd}=1.2 V) and (T=80°C, V_{dd}=1.0 V) have similar impact with respect to the SER. This demonstrates how voltage scaling, which is widely used for managing on-chips temperatures, is *a double-edged sword with respect to reliability.*

1.5.2 *Long-terms* Effects of Temperature on Reliability

Last but not least, elevated on-chip temperatures, additionally, stimulate aging phenomena and thus magnifying their deleterious effects on the on-chip systems reliability (i.e. *long-term* effects on reliability). This can be observed in Figure 1.12 that demonstrates how higher temperatures contribute to higher aging-induced degradations in terms of the electrical characteristics of a 22 nm PMOS transistor (i.e. V_{TH} and μ) within a 6-T SRAM cell operating at V_{dd}=1.2 V.

Importantly, the formation of the channel of a MOSFET transistor beneath the gate dielectric entirely depends on both V_{TH} and μ. Thus, higher aging-induced degradations in both of them due to elevated temperatures result in degrading the transistor's drain current I_D as well, which is considered the key electrical characteristic in MOSFETs. This can be also observed from the following equation that approximately models the drain current I_D in MOSFET [64]:

$$I_D = \mu \cdot \frac{C_{ox} \cdot W}{2 \cdot L}(V_{GS} - V_{TH})^2 \cdot (1 + \Phi V_{DS}) \tag{1.2}$$

Where, W is the gate width, L is the gate length, C_{ox} is the gate oxide capacitance per unit area and Φ is the channel-length modulation parameter.

Furthermore, according to the previous equation, the MOSFET Transconductance (g_m), which represents how easy the drain current to flow, can be expressed (during the saturation mode of MOSFET) as follows [64]:

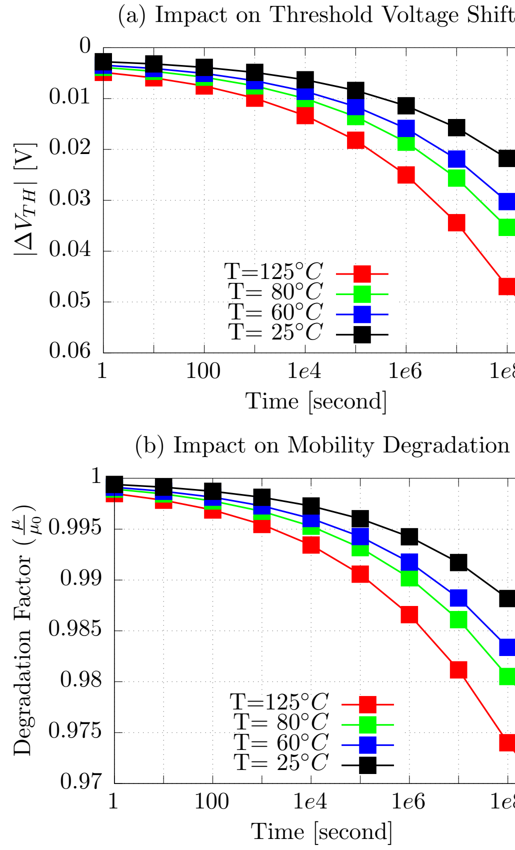


FIGURE 1.12: Impact of *long-term* effects of elevated temperature on degrading the electrical properties of transistors due to aging phenomena

$$g_m = \frac{2I_D}{V_{GS} - V_{TH}} = \mu \cdot C_{ox} \cdot \frac{W}{L}(V_{GS} - V_{TH}) \cdot (1 + \Phi V_{DS}) \tag{1.3}$$

As it can be noticed, degradations in both V_{TH} and μ (i.e. larger threshold voltage increase akin to ΔV_{TH} and smaller mobility) due to temperature-induced stimulated aging effects degrade the MOSFET transconductance (g_m) as a result.

It can be concluded that aging phenomena degrade the key electrical transistor's characteristics (i.e. threshold voltage, carrier mobility, drain current and transconductance). Additionally, elevated on-chip temperatures simulate more aging effects and thus they lead to magnifying the aforementioned induced degradations.

It is worthy to note that the time span within the experiments in Figures (1.9, 1.10 and 1.11) is just one second in order to present the impact of temperature standalone without aging phenomena in action. Whereas, the time span within the experiments in Figure 1.12 has been been selected as 10 years to show the impact of aging phenomena on the electrical characteristics of transistors during a typical/standard lifetime.

In summary, elevated temperatures jeopardize the chip's reliability due to their *short-term* and *long-term* effects. Therefore, there is an essential need to develop novel techniques that enable researchers to accurately investigate the thermal characteristics of embedded on-chip systems in order to correctly estimate their reliability as well as grasping how aging phenomena may ultimately fail them.

1.6 Thesis Contributions

The key objective of this thesis is to increase the reliability of embedded on-chip systems with respect to aging effects, soft errors and temperature-dependent behavior. First, it proposes an abstracted, yet sufficiently accurate reliability estimation that combines, from the physical to system level, the effects of *multiple simultaneous* aging phenomena and their interdependencies. Then, the reliability of the register file is investigated with the purpose of proposing novel techniques that increase register file resiliency against aging effects as well as soft errors. This is due to that fact that both of them are recognized as first-class candidates to jeopardize the entire on-chip system reliability in the nano-CMOS era as motivated.

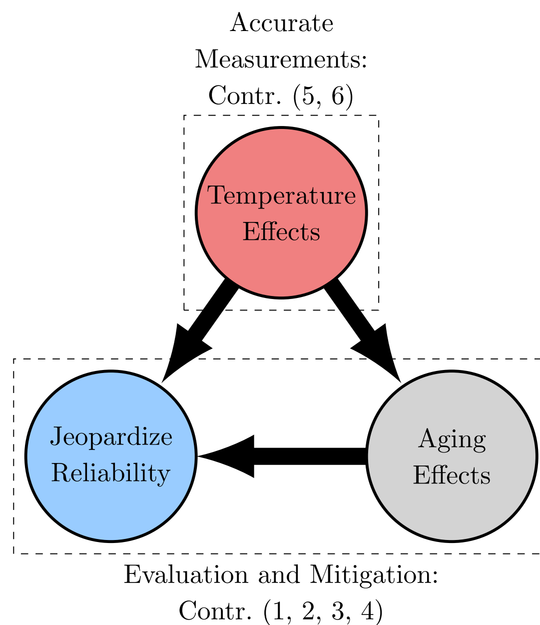


FIGURE 1.13: A high-level overview of the contributions within this thesis

This thesis also presents a novel in-house IR camera-based thermal measurement setup that enables designers to accurately investigate the thermal characteristics of their on-chip systems. The prime concern of the built setup is to accurately investigation the thermal characteristics of on-chip systems towards accurately estimating their reliability as well as to support the design-time exploration by providing designers with an accurate baseline which the effectiveness of their developed thermal management techniques on the chip's thermal behavior can be compared against. The setup has been employed to analyze both Field-Programmable Gate Array (FPGA) and Application-Specific Integrated Circuit (ASIC)-based embedded on-chip systems.

In particular, the novel contributions within this thesis are as follows and the different abstraction levels, that are involved, are presented in Figure 1.14 along with a general overview in Figure 1.13:

1. **Reliability Estimation:** It combines, from the physical to system level, the effects of multiple aging phenomena occurring simultaneously based on their interdependencies and shows how considering a sole individual phenomenon results in a non-negligible reliability underestimation. It also abstracts the various degradations induced by aging (i.e. V_{TH}, μ, SNM, RAT etc.) towards a probabilistic fault analysis which has a more meaningful interpretation of reliability. This is unlike the majority of state-of-the-art that employs other quantification metrics (e.g., SNM-decrease and V_{TH}-increase) which are hard for interpreting the overall reliability degradation – especially at the system level.

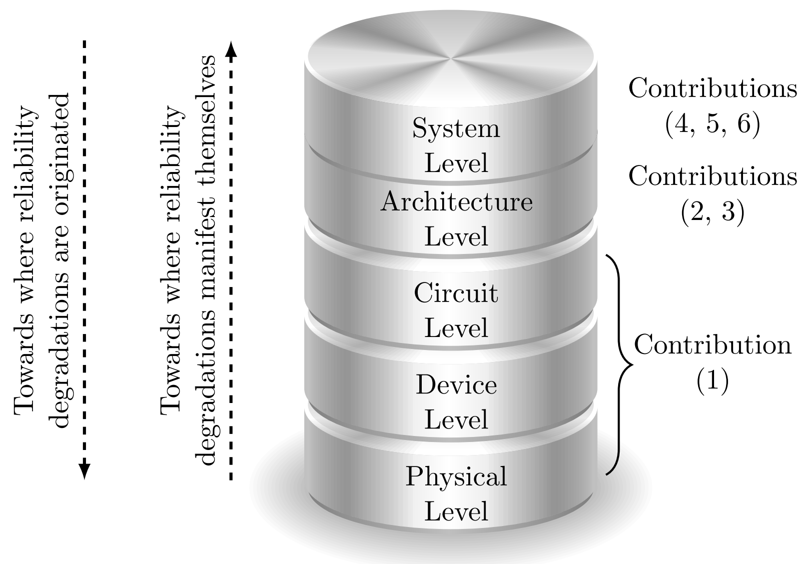


FIGURE 1.14: The involved abstraction levels within the contributions of this thesis

2. **The Novel Register-Internal Stress Balancing (RISB) Technique:** It proposes to *selectively* increase the resilience of individual registers of a register file against aging effects. It balances the applied aging stress to the transistors of cross-coupled inverters of SRAM cells within the register file such that both SRAM sides are under stress for approximately the same amount of time during operation – thereby it mitigates the deleterious effects of aging.

3. **The Novel Register-Embedded Self-Immunity (RESI) Technique:** It reduces the vulnerability of the register file not only against single bit upsets (SBUs) but also against multiple bit upsets (MBUs) by exploiting the unused bits of a register to embed the required Error Correction Code (ECC) – minimizing the need for extra bits. The RESI technique can additionally result in mitigating the aging stress in register file SRAM cells. Due to consuming less power per area, RESI also operates at a lower temperature compared to fully protecting the register file against SBUs only.

4. **Investigating Aging Effects under *Conforming Workloads*:** Estimating accurately the overall impact of aging on a register file within an on-chip system necessitates taking the running workloads on top of that system into account as their activities *drive* aging effects. As a matter of fact, this challenge is exacerbated when considering *conforming workloads* (i.e. several applications executed in parallel along with an operating system) that may be run on top of an embedded on-chip system because conventional simulation-based techniques may not be suitable to conduct the required analysis due to the unaffordable computational intensity that comes from simulating such very complex workloads. For that purpose, a novel real-time hardware-based technique has been implemented in an FPGA platform to extract the aging stress within the register file cells of the LEON3 processor. It enabled us to investigate aging effects that may be induced during the entire execution of several parallel applications along with an operating

system. This broadens the scope of estimating reliability from looking at a specific window of execution time of workloads of individual applications running on bare metal[2] towards looking at the entire execution of *conforming workloads*.

5. **The Thermal Investigation of FPGA-based Processors:** The analysis of the thermal behavior of commonly used FPGA-based processors (i.e. Microblaze, LEON3 and PowerPC) demonstrates that the thermal hotspot is located on the memory interface (e.g., DDR controller) due to its thermal properties (e.g. access frequency, operating voltage, capacitances). Therefore, a model linking the cache configurations with the FPGA chip's peak temperature for design-time exploration has been proposed.

6. **The Thermal Investigation of ASIC-based Processors through the Novel Rear-side Thermoelectric-based IR Thermography (RAMA) Technique:** For ASIC-based processors, an IR-transparent cooling needs to be provided to prevent the chip during measurements from overheating. To this end, the majority of state-of-the-art employs a coolant liquid, that is designed for IR spectroscopy. The problem is that several aspects like thermal convection may interfere with the measured IR radiation resulting in equivocal IR images. Thus, they decrease the accuracy in a way that leads to incorrectly estimating reliability. To solve this prominent and known problem, we introduce a novel setup for IR-transparent cooling that cools the chip from the rear side allowing the camera to *perspicuously* capture the IR emissions as no additional layer in between can impede the radiation. First, we evaluate our setup using an Intel dual-core processor operating at 1.8 GHz and show that the peak temperature of the chip can be configured to be equivalent to the unmodified (i.e. original) heat sink-based cooling. Then, we employ our built setup to analyze the thermal characteristics of state-of-the-art Intel 22 nm *octa-core* processor operating at 2.4 GHz and investigate the spatial thermal gradients, known as major causes for degrading reliability. We demonstrate how state-of-the-art inaccurate thermal analysis results in incorrectly estimating reliability. The built setup is the most accurate, least intrusive one that has been both proposed and actually applied to state-of-the-art multi-cores.

[2]This is necessary in simulation-based techniques to keep simulation times reasonable.

1.7 Outline

Before the contributions of this thesis are presented in detail, Chapter 2 gives an overview of how state-of-the-art estimates reliability as well as it discusses state-of-the-art techniques in mitigating the aging effects and soft errors in register files. It also provides a background of how stored values within SRAMs influence the induced aging stress in their transistors as well as it explains the concept of register file vulnerability that is often applied to estimate the reliability of the register files with respect to soft errors.

Chapter 3 presents our proposed reliability estimation that considers the simultaneous occurrence of aging phenomena towards providing a probabilistic fault analysis that abstracts the overall impact of aging effects on the register file reliability at the system level.

Chapter 4 analyzes the induced aging stress in the register file cells demonstrating aging stress in different registers needs to be tackled using different strategies corresponding to their access patterns. Then, our RISB technique is presented along with the potential overhead.

Chapter 5 explains our light-weight RESI technique to increase the resiliency of the register file against soft errors. Section 5.1.1 shows how the technique can be utilized to counteract the SBUs in register files. Whereas, Section 5.1.2 shows the case of MBUs.

Chapter 6 details the experiments results to evaluate the aforementioned proposed technique along with a comparisons to state-of-the-art. After evaluating in Section 6.3 the impact of our RISB technique in balancing the stress of aging in the register file SRAM cells, the register file vulnerability against soft errors and the fault system coverage after implementing RESI technique are discussed in Section 6.2. Additionally, we present also our hardware platform for investigating the reliability of the register file of embedded on-chip systems under *conforming workloads.*

Chapter 7 discusses the shortcomings of the conventional thermal analysis methods and motivates the need for an IR thermal camera-based measurement setup. Then, it analyzes the thermal behavior of FPGA-based embedded processors along with the impact of a CPU's cache on the peak temperature of chip. Last but not least, Section 7.3 illustrates the implementation of RAMA technique along with the evaluation of its effectiveness on providing lucid thermal images as well as comparing the reliability estimating based IR images captured by our built setup with state-of-the-art.

Finally, Chapter 8 concludes this thesis and gives an outlook.

Chapter 2

Background and Related Work

Over the last decade, rapid technology scaling and evolution, within the nano-CMOS era, brought many new advances in diverse facets of IC manufacturing. These new advances have steadily reduced the resiliency of modern chips against multiple aspects (e.g., aging phenomena and soft errors) and therefore increased their susceptibility to various kinds of failures that can be caused by the induced reliability degradations.

In the following, we discuss state-of-the-art methodologies in estimating reliability. Then, we explain how aging stress degrades SRAM cells and how it can be mitigated along with a discussion regarding the impact of *balanced* as well as *unbalanced* aging stress on the reliability of SRAM cells. Next, we present state-of-the-art aging mitigation techniques in SRAM-based register files. Afterwards, we present state-of-the-art techniques in increasing the resiliency of register files against soft errors along with the concept of Register File Vulnerability (RVF). Finally, we present how the thermal characteristics of on-chip systems are typically analyzed as well as we explain state-of-the-art techniques in building IR measurements-based setups for multi-cores processors.

2.1 Reliability Estimation akin to Aging Effects

Aging Effects Analysis:
As reliability becomes a more imminent challenge, researchers increasingly focus on understanding and modeling the physical processes behind NBTI/PBTI [14] and HCID [16] aging mechanisms. When combining the effects of multiple aging mechanisms is targeted, state-of-the-art methodologies (as also summarized in Figure 2.2) can be categorized into the following three categories:

1) System-level (e.g. [65, 66]): Where the failure rate of each aging mechanism (λf) is *individually* calculated and, then the overall reliability is estimated based on the Sum-Of-Failures-Rate (SOFR) rule within the Reliability-Aware MicroProcessor (RAMP) model [67], which is an architectural model for long-term processor reliability estimation.

The key problem behind this methodology is the assumption that each failure mechanism independently proceeds, is not valid anymore because recent observations through

measurements [16, 68] established that BTI aging mechanism *simultaneously* occurs along with HCID. Additionally, [51] showed that HCID models have intrinsic BTI components and thus treating these mechanisms *separately* may lead to overestimating the overall aging-induced degradations due the twofold consideration of BTI (which itself is significant), as may be expressed in the following equation.

$$\frac{1}{MTTF_{Total}} = \overbrace{\frac{1}{MTTF_{BTI}}}^{\text{BTI Model}} + \overbrace{\frac{1}{MTTF_{HCID}} + \frac{1}{MTTF_{BTI}}}^{\text{HCID Model}} \rightarrow \frac{2}{MTTF_{BTI}} + \frac{1}{MTTF_{HCID}}$$

Moreover, estimating the reliability degradations relying on Mean-Time-To-Failure (MTTF) equations (which is for the sake of simplicity and to keep the computational time minimum) can result in either ignoring (e.g., [69]) or oversimplifying (e.g., [70]) modeling the recovery mechanism of BTI leading to underestimating or overestimating, respectively, the recovery (i.e. self-healing) impact on reliability when the aging stress (i.e. voltage stress) is ceased. Figure 2.1 illustrates, for instance, the impact on overestimating the aging-induced degradation (i.e. ΔV_{TH}) when the recovery mechanism of NBTI within a 22 nm PMOS transistor is not taken into account.

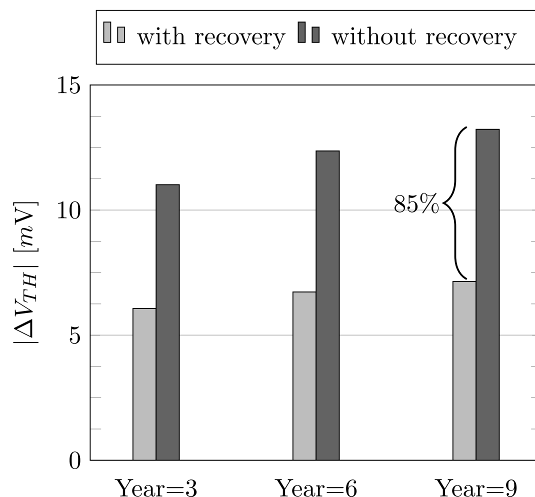


Figure 2.1: Impact of not considering the recovery mechanism of BTI on overestimating aging-induced degradation

2) Circuit-level (e.g., [71, 72]): There, mainly the dominant aging mechanism in each transistor is considered to estimate the overall reliability degradation of the entire analyzed circuit. In practice, it considers solely NBTI in PMOS and solely HCID in NMOS. This is due to the assumption that other aging mechanisms like HCID in PMOS and PBTI in NMOS are negligible. This assumption was reasonable in the past with ≥ 45 nm technology nodes but the recent introduction of high-K materials along with technology scaling, that causes a shorter MOSFET channel, (as found in 22 nm) has strengthened (i.e. increase number of aging-induced defects) the HCID mechanism in PMOS and the PBTI mechanism in NMOS [51] (see Section 1.3.2). Therefore, such a methodology can lead to underestimating the induced degradations as it will be investigated in Section 6.

Another methodology in [73] proposed to integrate multiple failure-equivalent circuits, which model multiple aging mechanisms, within the analyzed circuit. While SPICE can model the interrelations between these equivalent circuits (e.g., the overall ΔV_{TH} due to N aging mechanisms, which can shift V_{TH}, will be represented as $\sum_i \Delta V_{THi}$), the assumption that aging mechanisms are independent still applies because each individual aging mechanism is represented with its own equivalent circuit, whereas in fact these aging mechanisms simultaneously occur at the physical level (which SPICE cannot model)

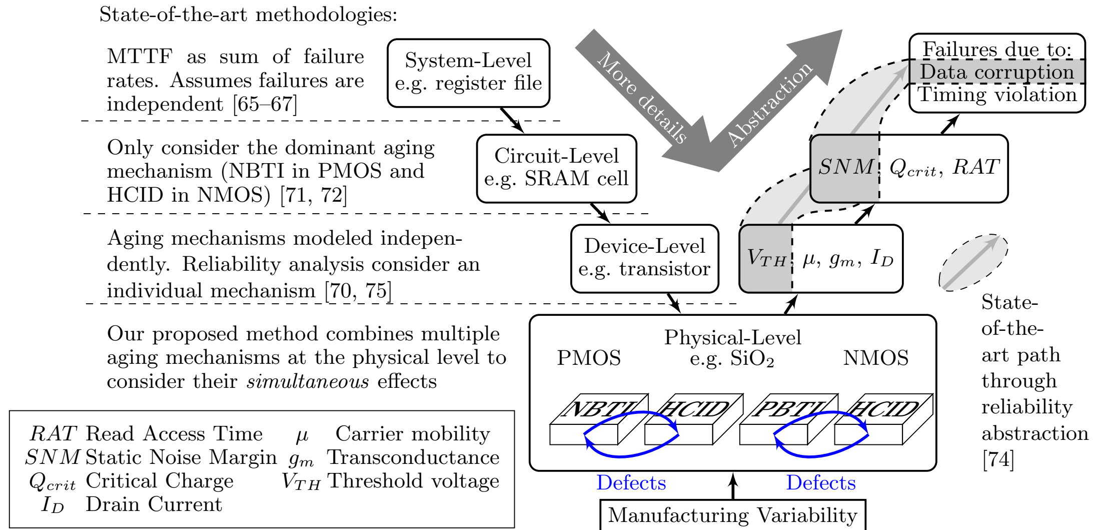


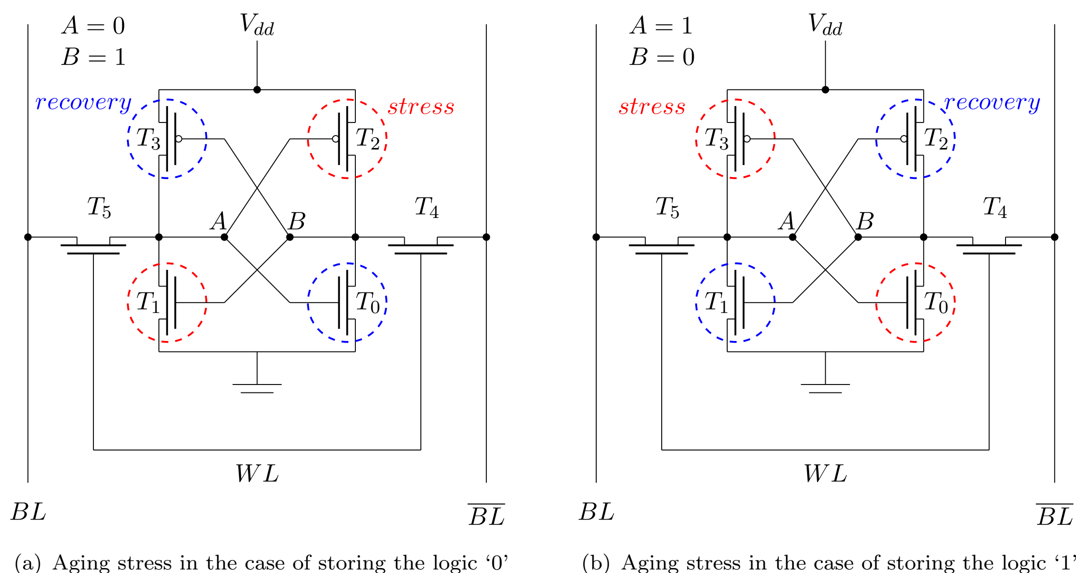
FIGURE 2.2: Abstracting aging effects

and they should be represented as one comprehensive failure-equivalent circuit rather than multiple ones to avoid overestimating the overall aging-induced degradations.

3) Device-level: The Berkeley Reliability Tool (BERT) [75] and similarly the RelXpert [70] from Cadence (which is based on BERT) combines different aging simulators into a modular reliability framework. Each simulator module individually models a single aging mechanism. While it can provide a rough guidance for design space exploration, it assumes, for the sake of enabling the modularity, that aging mechanisms are independent and thus existing interdependencies are missed. A Similar drawback exists in [76] where in-depth low-level simulation of SRAM libraries is performed. The latter also requires detailed knowledge of transistor parameters limiting the applicability at the system level.

Aging-induced Failure Analysis:
Regarding exploring the impact of aging-induce degradations on on-chip systems, [77] showed how aging increases the susceptibility to timing violations, [74] illustrated how the susceptibility to noise can also be increased and [78] reported the relation between the increase susceptibility to radiation and aging degradation. While these works focus only on one particular source of failure at a time when an individual aging mechanism (i.e. NBTI solely) is considered, they omit other sources of failures – *especially when multiple aging mechanisms simultaneously occur.*

Distinguishing from existing work: We propose within this thesis (details in Chapter 3) to combine the effects of *multiple simultaneous* aging mechanisms from the physical level to correctly analyze the induced degradations at the device/circuit level and to provide an abstracted, yet sufficiently accurate reliability estimation at the system level summarizing how aging-induced defects in the transistors gate dielectric will ultimately increase the probability of failures of the entire system due to both sources of failure (i.e. data corruption as well as timing violations), as Figure 2.2 demonstrates.



(a) Aging stress in the case of storing the logic '0' (b) Aging stress in the case of storing the logic '1'

FIGURE 2.3: Aging stress within a 6-T SRAM cell

2.2 Aging Effects Mitigation

2.2.1 Aging Stress Balancing

Let us revisit again the standard 6-T SRAM cell: As explained in Section 1.2, it contains two inverters (see Figure 1.2(a)) for storing complementary bit values. This means that while a logic value is stored within an SRAM cell, the PMOS transistor of the one inverter along with the NMOS transistor of the second inverter will constantly be in the *stress phase*, while other transistors will be in the *recovery phase*. This situation is sustained until the stored value is flipped. Then, the transistors within both inverters switch their roles with regard to the aforementioned stress. Figures 2.3(a) and 2.3(b) clarify how the aging stress is applied to SRAM transistors (T_0,T_1,T_2 and T_3) during the storage of the logic value '0' and '1', respectively.

As earlier discussed in Section 1.2.1, one of the key reliability metrics in the SRAM domain is its SNM as it describes its resiliency against noise and thus against failures due to data corruption. As a matter of fact, when aging alters the electrical characteristics of SRAM transistors (e.g., increasing the V_{TH} of transistors, as Figure 1.12 illustrated), the SNM becomes smaller (see Figure 2.4) which, as a result, degrades the entire SRAM reliability. Please note that the very high shown degradation within Figure 2.4 is due to the high employed ΔV_{TH}

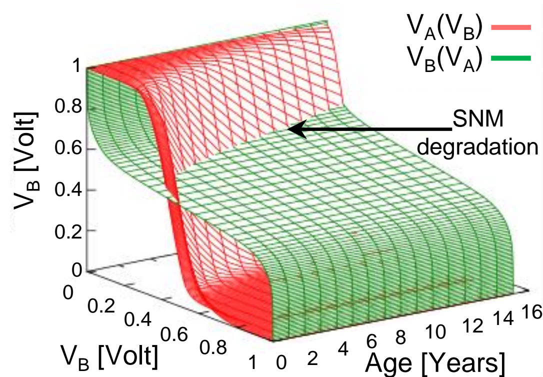


FIGURE 2.4: Impact of V_{TH} increase in PMOS transistors within a 6-T SRAM on the static noise margin (SNM) [5, 13]

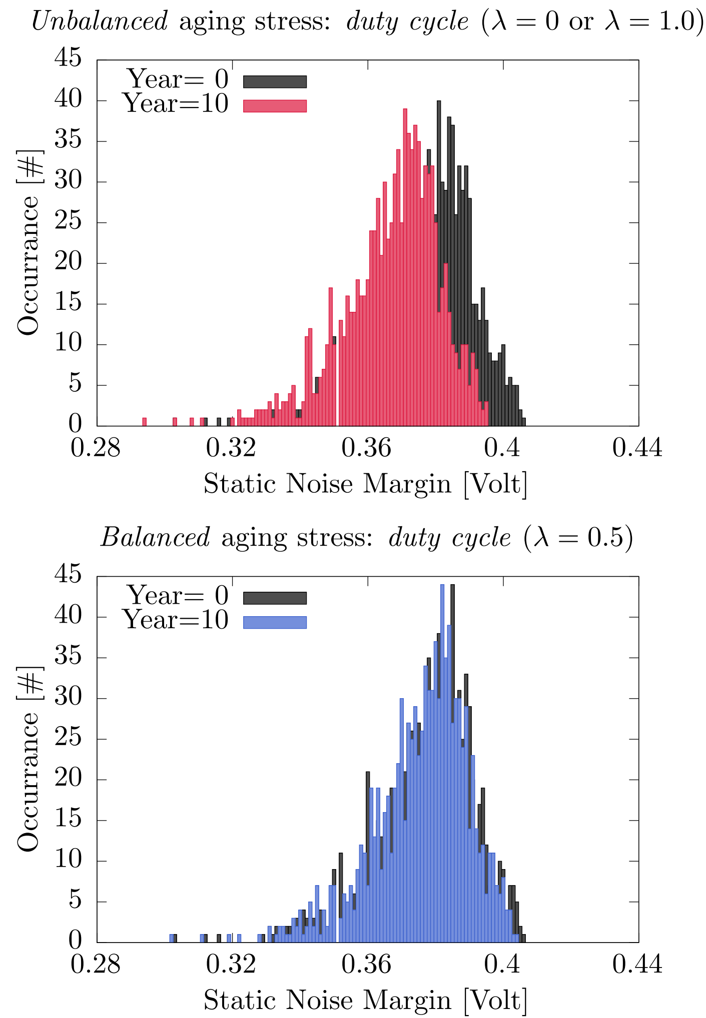


FIGURE 2.5: Impact of aging on SNM degradation for the *balanced* and *unbalanced* aging stress cases. As shown, *unbalanced* aging stress results in shifting the SNM distribution more towards the left side which, in turn, makes the SRAMs resiliency against noise less leading to a higher probability of data corruption-induced failure

(i.e. more than what aging can actually in-
duce) which is assumed to amplify the tendencies for a better clarity. However, the observation (i.e. SNM becomes smaller when V_{TH} increases) still applies at normal ΔV_{TH} as in the following will be presented based on Figure 2.5.

Assuming that the probability that the logic value '0' is stored in an SRAM cell is *duty cycle* (λ), then the probability of that the PMOS/NMOS transistors of the first and second inverters (i.e. T_1, T_2) will be under stress is λ. Whereas, the probability of other transistors (i.e. T_0, T_3) to be under stress is $(1 - \lambda)$. When λ is 1, both T_1 and T_2 transistors are constantly under stress, whereas, T_0 and T_3 transistors are constantly under recovery. Thus, T_1 and T_2 will age much fast than T_0 and T_3 transistors. On the other hand, λ=0 indicates that both T_1 and T_2 transistors are constantly under recovery, whereas, T_0 and T_3 transistors are constantly under stress. Thus, T_0 and T_3 will age much fast than T_1 and T_2 transistors. As a result, when λ is exactly 0.5, the *overall* aging effects on the entire SRAM cell will be at the lowest possible level because all

transistors will be *evenly* stressed and thus all transistors will similarly age (i.e. have similar aging-induced degradations). We call such case of ($\lambda = 0.5$) with a *balanced* aging stress and the other cases of ($\lambda \neq 0.5$) with an *unbalanced* aging stress.

The aging-induced degradations become even more severe when accounting for transistor variability. Figure 2.5 shows the effect of manufacturing variability on the SNM along with demonstrating the impact of aging stress, for a lifetime of 10 years, on shifting the distribution of SNM for both *balanced* (i.e. $\lambda = 0.5$) and *unbalanced* (e.g., $\lambda = 0$) stress cases. As it can be noticed, *unbalanced* aging stress considerably results in shifting the SNM distribution towards the left side more than the *balanced* case. This, in turn, increases the probability that an SRAM cell has a low SNM value resulting in higher probability of failures due to data corruption (i.e. the SRAM cell is not able anymore to suppress the noise).

In summary, mitigating the aging effects in SRAM cells necessitates balancing the aging stress (i.e. making λ for each SRAM cell as close as possible to 0.5). In other words, the key principle behind increasing the reliability of an SRAM-based component (e.g. a register file of an embedded on-chip system which is our focus within this thesis) is developing a technique that is capable of balancing the aging stress within its SRAM cells over its lifetime.

2.2.2 State-of-the-art Techniques

A study on reliability issues related to modeling and analyzing NBTI effects in SRAM cells with a discussion on the SNM degradation has been introduced in [79] showing that NBTI can degrade the SNM of an SRAM cell over time – especially regarding the stability during a read operation. Abella et al. [80] discussed the importance of developing new techniques that overcome the NBTI effects in new microprocessors and proposed a generic strategy to protect different blocks within a CPU against NBTI, relying on the idle time of processor resources. Li et al. [81] proposed a technique to cope with the NBTI-induced degradations by exploiting the idle cycles in order to relax the stressed PMOS transistors. However, such an approach is specific for out-of-order architectures and cannot work for the case of in-order CPUs.

A comparison between the effects of NBTI and PBTI on the V_{TH} of PMOS and NMOS transistor, respectively presented in [82]. It has been shown that both NBTI and PBTI (noticeably) degrade the stability during *read* and (marginally) during *write* operations.

The *Recovery Boosting* technique [83] proposed a low-level technique that modifies the design of SRAM cells by adding an inverter that raises the node voltages of the cell, putting both PMOS transistors into the *recovery phase*. Such a circuit-level approach requires a modification of the hardware infrastructure of the SRAM cell itself and has other side effects on the SRAM characteristics since the area footprint grows etc.

The authors in [84] studied the NBTI-induced lifetime degradation in SRAM cells. Their introduced results illustrated that the NBTI impact is at the lowest level when aging stress is *evenly* distributed between the involved PMOS transistors (i.e.when $\lambda = 0.5$)

which is consistent with our simulations shown in Figure 2.5. This work proposed the Bit Level Rotation (*BR*) technique to balance the NBTI stress within the SRAM cells of a register file and it works as follows: at each write operation, the register is rotated using a circular shift, moving the LSB (bit0) by one position. When the number of rotations reaches the bit-width of the register, bit0 has used every cell in the register to store its value. The benefit of this technique is that the stored value in each bit in the register will be repeatedly changed, reducing the overall time that the PMOS transistors spend in the *stress phase* leading to mitigate the reliability degradation induced by NBTI. However, the effectiveness of this technique declines when the stored value contains a large number of sequential zeros or ones (which is often the case as it will be later observed from our analysis). A second scheme within that work, called Register Rotation, uses a repetitive shift operations to map the *zero register* to different registers leading to ensure that all the register file rows are utilized to store the *zero register* after a complete rotation. This, in turn, results in distributing the NBTI stress to all the registers but can only be implemented in CPUs that support register file mapping unit. However, the *zero register* is a special register which, in practice, is often implemented as "hard-wired" SRAM cells and thus it is not affected by NBTI.

Authors in [85] proposed to exploit the narrow-width values and modify the physical register file by creating register banks with different sized transistors providing various levels of NBTI tolerance. By changing the register renaming policy, an NBTI-aware register allocation can be performed with around 20% area overhead. While the aforementioned work can achieve a good NBTI mitigation, it requires a renaming unit which makes it specific for out-of-order CPUs and cannot work for in-order CPUs. [86] proposes to flip the whole register data, using an XOR component, whenever a register is read, and writes the new value back to that entry to mitigate the NBTI stress. This may lead to an increase of the total number of write accesses in addition to the power consumption from frequently accessing the XOR component. Such a technique tackles the NBTI stress in all registers using one strategy and which fails to take the varying access patterns of different registers into account.

Work in [87] is capable of effectively mitigating the NBTI stress throug a power gating-based scheme but only in the upper half of a register file in out-of-order CPUs. The NBTI stress evaluation is done by calculating the averaged *duty cycle* between both register file halves which does not correctly represent the overall impact of aging effects in the register file because both halves may have *unbalanced* aging stress (i.e. $mean(\lambda_{upper}) = 0$, $mean(\lambda_{lower}) = 1$), and thus aging stress within the entire register file is *unbalanced*, but through calculating the averaged *duty cycle* of both halves, we find that $mean\lambda_{total} = 0.5$ which indicates to the case of *balanced* aging stress. Whereas in fact, the aging stress is *unbalanced* because all SRAM cells within the register file have *unbalanced* aging stress (i.e. all *duty cycle* values are either '0' or '1' and non of the SRAM cells has a *balanced* aging stress). Compared to this work, our proposed technique RISB within this thesis, as it will be later in Chapters (4 and 6) presented, is able to balance both halves of the register file. Moreover, we study the distribution of aging stress across all bits inside the entire register file to avoid incorrectly estimating reliability of the register file which may lead to incorrectly evaluating the effectiveness of our proposed technique.

Distinguishing from existing work: Unlike the current aging mitigation techniques for register files, where the aging effects are mitigated in all registers using the same strategy, our RISB proposes (details in Chapter 4) to *selectively* increase the register file resiliency aging effects. Based on their classification as *frequent* or *infrequent*, aging stress in different registers is balanced using the corresponding suitable strategy. Additionally, we look at multiple simultaneous aging mechanisms (i.e. NBTI/PBTI and HCID) rather than solely NBTI as state-of-the-art techniques often consider.

2.3 Soft Error Mitigation

Conventional Single Error Correction (SEC) codes (like Hamming codes [88]), that are often used to protect susceptible CPU components against soft errors, are able to correct just a single bit upset. Unfortunately, the MBUs rate has started to sharply increase due to technology scaling [89, 90]. Thus, codes targeting only single errors become less effective. Therefore, a larger number of redundant bits can be used to build higher order protection codes that are able to correct more than one bit upset simultaneously such as the Reed Solomon [91]. The cost and complexity growth depends on the type of code used, but they are always more difficult to handle than SEC codes [92]. Traditionally, Triple Modular Redundancy (TMR) [93] has been used to cope with the MBUs but at a significant hardware cost. This may not be tolerable in many of today's embedded systems. K. Walther et al. [94] proposed to use the cross parity check as a method for correcting up to 5-bit upsets in multiple registers. The proposed approach comes at around 100% overhead in terms of register file area.

To analyze vulnerability, both the Architecturally Correct Execution (ACE) and UnACE times need to be extracted for each single bit within the register file. ACE represents the vulnerable intervals of execution time where the bit is susceptible to soft errors. Whereas, ACE represents the fraction of time where a soft error occurring in the bit will have no deleterious impact on the output, as it will not spread throughout the computational system. Then, the register file vulnerability can be modeled based on the Architectural Vulnerability Factor (AVF) concept [95, 96] as follows:

$$AVF(\text{RegisterFile}) = \frac{\sum_i ACE(B_i)}{\sum_i ACE(B_i) + \sum_i UnACE(B_i)} \tag{2.1}$$

There, ACE(Bi) and UnACE(Bi) are the ACE and UnACE time of the Bi bit, respectively.

In general, bits in a non-protected register are considered invulnerable against soft errors when the next access will be a 'write' operation because any soft error in that register will be harmless due to the overwriting. On the other hand, it is considered vulnerable at any cycle if the next access to that register will be a 'read' operation because an occurred bit upset can spread to and harm other microarchitectural components as it is clarified in Figure 2.6.

As a matter of fact, a vulnerable interval becomes invulnerable if the applied protection technique is capable of correcting the bit upset occurring during that interval and, as a result, it prevents errors from propagating to other CPU components. When a protection technique applied to the register file requires extra bits (e.g. control bits, parity bits, etc.), both the vulnerable and invulnerable fractions of time (i.e. ACE and UnACE, respectively) need also to be extracted for each extra bit to take their impact also into account.

The Shield architecture [97] determines whether to protect the register value or not by predicting its lifetime using extra logic. The register values with a larger lifetime need to

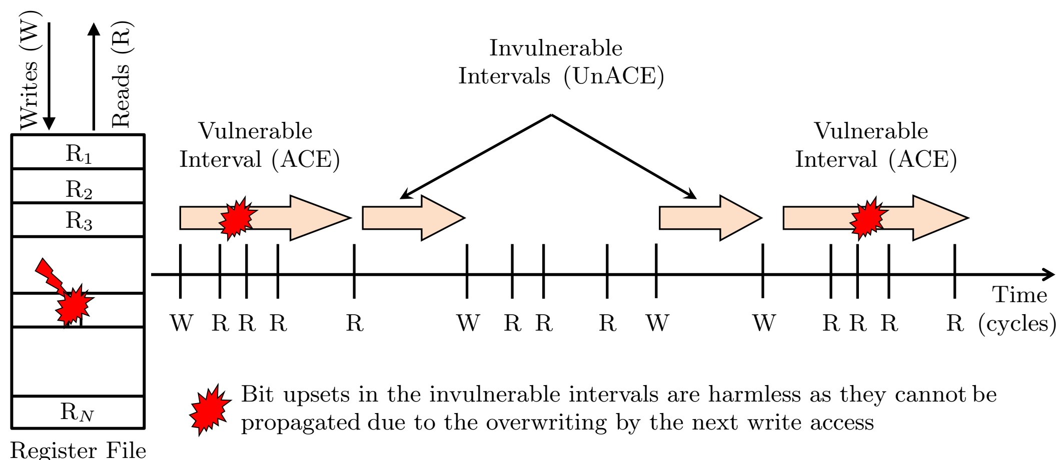


FIGURE 2.6: Vulnerable and invulnerable intervals during accessing a non-protected register, where soft errors in invulnerable intervals have no deleterious effect on the other microarchitectural components within the on-chip system

be protected as they stay longer in the register file and are exposed more to soft errors. It is designed for processors with out-of-order execution and it cannot correct MBUs as it is based on the SEC code. Moreover, the energy cost stemming from the runtime prediction can be high [98], which may not be tolerable in embedded on-chip systems.

A cost-efficient technique called 'In-Register Replication' (*IRR*) [99] exploits the register values that require less than or equal to 16 bits to be stored in 32-bit architectures. These values can be replicated within the same register to improve the register's resiliency against errors. A work close to the *IRR* technique has been presented by J. Hu [100] studying a 64-bit out-of-order microarchitecture and similarly proposing to duplicate, within the one register, register values that need 32 bits or less to be coded. Such techniques fail, however, when adjacent MBUs, e.g. induced by a strike from an energetic particle along with other strikes induced by the other secondary generated particles, simultaneously affect both the lower and upper halves of a register. Comparison results with this approach (as will be discussed later in the evaluation section) show that our RESI technique, proposed within this thesis, achieves a higher vulnerability reduction as well as a better system fault coverage under various fault injection scenarios.

To avoid area and power penalties that come from protecting all the registers, authors in [101] proposed a partial protection scheme. Instead of fully protecting the register file (i.e. all of the registers), their technique only protects the most vulnerable registers. The underlying protection mechanism can be based on either SEC code or replication. The effectiveness of this technique relies on the number of protected registers.

Lee et al. [98] presented a compilation technique to reduce the vulnerability of the register file (on average 37%) by writing the most vulnerable register values into a small protected part in the memory by adding extra load/store instructions to the code. The advantage of their technique is that it comes with no extra hardware cost (except the protected part of memory), but the achieved vulnerability reduction is relatively low and the technique increases the code size (up to 24%).

Fazeli et al. [96] introduced an MBUs protection technique for register files. It caches the vulnerable register values in a small cache using the average number of 'read' operations to decide which cache entry should be replaced. The achieved vulnerability reduction is 90% while the technique occupies 18% more area and consumes 25% less power compared to fully protecting the register file against SBUs only. Compared to this work, our RESI technique achieves a better vulnerability reduction along with a better cost saving as will be later demonstrated.

Distinguishing from existing work: Our RESI technique (details in Chapter 5) tackles the challenge of increasing the register file resiliency against SBUs as well as MBUs with minimum impact on overhead. The key conflict when protecting a register file is that – while maximizing register file resiliency against soft errors – the achieved vulnerability reduction (either with full or partial protection schemes) increases both area and power overheads.

2.4 On-Chip Thermal Investigation

In addition to ASIC-based processors, where analysis of thermal behavior has always been an important concern (e.g., [102, 103]), we also target within this thesis the thermal investigation of FPGA-based systems as there has been an increasing focus on them in recent years [104] due to their high level of functionality, flexibility, and advantageous low-volume production costs.

2.4.1 FBGA-based On-Chip Systems

Towards achieving higher performance, manufacturers continually increase the logic density in FPGAs making technologies sizes of 28 nm (Xilinx Vitrex7, Altera Stratix V) common. This results in high on-chip power densities and thus elevated temperatures similar to ASICs.

An early study on the manner in which the FPGA fabric heats up was conducted by Sundararajan et al. [105]. Relying on the results obtained from the HS3D thermal simulator they established the amount of heat generated by the different types of resources that can be instantiated in an FPGA design. As this study is based on temperature simulation the accuracy is limited as will be demonstrated in Section 7.2. Huang et al. [18] went so far as to validate their HotSpot thermal simulation tool with a 0.13 μm FPGA platform. They proposed to spread many Ring Oscillator (*RO*)s across the FPGA die to measure the temperature of different operating blocks. Thermal variations of up to 0.7°C were measured, and the simulated and measured values correlated with errors within 0.2°C. The limited range of these results leads us to suggest that this correlation is the result of the heating trend and are not sufficient to validate the HotSpot thermal simulator.

Long before FPGA self-heating became a key concern, Boemo et al. [106] proposed *RO*s as temperature sensors in reconfigurable logic. An array of such sensors was later used to observe the thermal behavior of a 0.22 mum FPGA [107] configured with two *soft* (i.e. synthesizable) processors. This showed that a change in processing throughput resulted in a change in temperature though these variations were very small (within −0.8°C and +0.7°C from the mean temperature). Similar work was performed several years later by Zick et al. [108] who also configured an array of *RO*s on a modern 65 nm FPGA to analyze process variation and to track the aging of the chip's fabric. While this paper had less to do with temperature, much consideration was given to *RO*s design, together with an evaluation of their bleak feature as temperature sensors.

Happe et al. [109] researched the precise design of micro-heaters and their transient effect on temperature based on the components they are made up from (LookUp Tables (LUTs) and Flip-Flops (FFs)) at different frequencies in order to explore the potential maximum generated heat in FPGA platforms. To estimate the thermal map, a grid of calibrated *RO*s-based theraml sensors has been utilized.

Instead, thermal measurements of chips can be obtained using the available thermal diode sensors. Unfortunately, the amount of these sensors is restricted because they are area and power hungry [110]. This limitation may result in a failure to capture the peak chip temperature, particularly when the thermal hot spot is located far away from the placement of the sensor, which is fixed during design time, as it will be clarified in Section 7.2.

In Chapter 7.2, we discuss the limitations of the aforementioned conventional methods of thermal analysis in order to illustrate the need for using an IR camera-based thermal measurement setup to accurately explore the thermal characteristics of chips at design time. Employing a thermal camera for analyzing FPGA temperatures has been proposed by Cochran et al. [110]. This work presented a methodology to convert the captured IR images to estimated power patterns and demonstrated the low-pass filter effect the die has on temperature.

Distinguishing from existing work: Neither of these papers investigated the temperature of FPGA-based embedded on-chip systems to investigate the responsible parts for the chip's temperature increase, despite the fact that nowadays they are considered one of the key usages of FPGAs. We demonstrate in Section 7.2.2, the dominant role of the cache configuration on the thermal behavior of these systems and the relation between different cache parameters and the memory interface accesses. Though the power impact of cache on a system has long been studied [111], little is known of its thermal impact, especially when targeting FPGA platforms.

2.4.2 ASIC-based On-Chip Systems

Building such an IR thermography-based measurement setup inevitably requires removing the cooling and packaging from the chip in order to expose its silicon wafer and thus allow the IR radiations emitted from the chip to reach the camera's lens. Unlike FPGA-based processors that can still properly operate after exposing its bare silicon, due to their relatively low operating frequencies (e.g., < 100 MHz in the LEON3), measuring the temperature of ASIC-based processors presents a more challenging problem as it necessitates building an alternative IR-transparent cooling to allow the IR radiation emitted from the chip to reach the thermal camera and concurrently counteract the very rapid increase in temperature due to the excessive on-chip power densities.

To this end, state-of-the-art techniques [46, 102, 112] use a liquid-based cooling setup that applies a layer of IR-transparent liquid on top of the measured chip. Through a continuous stream of cooled liquid, the chip temperature can be controlled and thus the setup prevents it from damage. The problem is that several aspects like thermal convection may interfere with the measured IR radiation resulting in equivocal IR images. Thus, they decrease the accuracy in a way that leads to incorrectly estimating reliability.

To demonstrate the aforementioned problem, we present in Section 7.3 a comparison between *equivocal* IR images – when a thin layer of IR oil is added on top of chip during measurements – and *lucid* IR images that have been captured without the addition of any

layer on top of the measured chip. Additionally, we investigate how inaccurate thermal analysis leads to inaccurately estimating the key reliability metrics such as SNM, RAT and Q_{crit}.

Distinguishing from existing work: In addition to the complexity that comes with building state-of-the-art in IR thermal measurements, having additional layers on top of the measured chip is disadvantageous because of significant interference with the IR radiation which, in turn, cause the IR images to lose its lucidity resulting in inaccurately estimating the on-chip system reliability. Solving this prominent problem, we introduce a novel technique (RAMA) for IR-transparent cooling that cools the chip from the rear side allowing the camera to *perspicuously* capture the IR emissions as no additional layer in between impedes the radiation.

2.5 Summary

Considering that the register file is one of the crucial parts of almost any microprocessor, increasing the its reliability with respect to aging effects and soft errors is a prerequisite in the nano-CMOS era. Therefore, developing new techniques for that purpose along with a minimum impact on the cost (i.e. area, power overheads) is desirable for embedded on-chip systems – especially for those that are designed to operate under tight constraints.

On the other hand, technology scaling has made temperature concerns one of the major challenges that faces designers due to their negative impact on reliability. In order to investigate the thermal characteristics of modern chips, real-time thermal measurements obtained from an IR camera are substantially needed to provide an accurate thermal analysis that can be employed in correctly estimating the on-chip system reliability.

Chapter 3

Reliability Estimation Through the Interdependencies of Aging Mechanisms

With technology in deep nano scale, the susceptibility of transistors to various aging mechanisms such as NBTI, PBTI and HCID etc. is increasing. As a matter of fact, different aging mechanisms *simultaneously* occur in the gate dielectric of a transistor. In addition, scaling in conjunction with high-K materials has made aging mechanisms, that have often been assumed to be negligible (e.g., HCID in PMOS and PBTI in NMOS), become noticeable as it has been discussed in Section 1.3.2. Therefore, in this chapter we investigate the key challenge of providing designers with an abstracted, yet sufficiently accurate reliability estimation that combines, from the physical to system level, the effects of *multiple simultaneous* aging mechanisms through their interdependencies [4]. We show that the overall aging can be modeled as a superposition of the interdependent aging effects. Our presented methodology deviates by around 6% from recent industrial physical measurements. We conclude from our experiments that an isolated treatment of individual aging mechanisms is insufficient to devise effective mitigation strategies in current and upcoming technology nodes. We also demonstrate that estimating reliability due to an individual dominant aging mechanism together with solely considering a single kind of failures, as currently is a main focus of state-of-the-art (e.g., [74]), may result in 75% underestimation on average.

3.1 Motivation

The International Technology Roadmap for Semiconductors (ITRS) states that upcoming technology nodes introduce reliability challenges at an increased pace compared to the last decade [32] because devices below 45 nm are increasingly susceptible to multiple aging mechanisms. This is primarily due to the higher vertical/horizontal electric fields within scaled MOSFETs along with the employment of the new high-K material in forming the transistor's dielectric. We therefore focus within this chapter on estimating the

reliability with respect to the deleterious impact of aging effects, that originate at the physical level (i.e. transistor dielectric), on the probability of failures, that may occur at the system level (i.e. where workloads/applications run).

Aging Effects: Shrinking feature sizes leads to higher electric field strengths, as well as higher current densities (see Section 1.3.2), which both accelerate device aging and thus increase degradation of transistor electrical characteristics which can ultimately turn into failures. NBTI, PBTI and HCID have become the most prominent aging mechanisms impeding reliable transistors. Their effects on aging are more significant than others including Time-Dependent-Dielectric Breakdown (TDDB) [113], even in current high-K transistors [114]. While understanding the physical processes of aging mechanisms is not entirely required at the system level, there is still a substantial need to analyze their impact on degradations to accurately estimate reliability – *this holds even more when multiple aging mechanisms interdepend i.e. when they interact with each other.*

Of the two forms of BTI, NBTI degrades PMOS transistors and PBTI degrades NMOS transistors, whereas HCID degrades both. Over time, aging-induced degradations ultimately cause transistor malfunctions and increase a circuit's susceptibility to failures. Such failures are mainly due to timing violations and data corruption caused by voltage noise or radiation. We focus within the paper on how simultaneous occurring aging mechanisms jointly increase the probability of these failures.

The Challenge of Combining Aging Effects: Recently introduced physics-based aging models such as [15, 16] describe the detailed underlying physical process behind aging mechanisms to interpret them. Additionally, measurements have shown that these processes simultaneously occur [16, 51, 68]. Unlike higher-level aging models (e.g., [115]), physics-based models are more accurate but complex as they are highly device-dependent and computationally intensive which is due to the large number of chemical bonds which need to be modeled along with their varying properties (e.g., the Si−H bonds affected by BTI exhibit a wide range of density variability due to locally higher breaking rates induced by the interaction with HCID). As these models aim to fully capture the actual underlying physical processes – at the atomic level – along with modeling them in-depth (e.g., interpreting aging in the order of $[\mu - m]sec$), their computationally intensive solutions are limited to a single transistor device – especially when aiming to study multiple mechanisms results in a significant increase in the complexity. This makes such solutions not feasible for designers at the system level dealing with tremendous number of transistors, that form an entire on-chip system or a microarchitectural component within it, in order to estimate the impact of aging on reliability during a typical chips lifetime (e.g., in the order of years).

Finally, manufacturing variability also plays an important role as it varies transistors characteristics which makes similar transistors be differently degraded by aging effects. Therefore, paying attention to it is inevitable when estimating reliability.

In summary: Analyzing failures due to isolated individual aging mechanism is insufficient in order to estimate the overall reliability because the interdependencies do matter.

Demonstrating this is our goal along with showing how the effects of multiple simultaneous mechanisms can be combined towards providing an abstracted, yet sufficiently accurate reliability estimation.

3.2 Problem Formulation

System designers aim to estimate the lifetime of their on-chip systems in order to determine the additional cost of sustaining reliable operation during runtime. Such an additional cost comes through over-designing the circuits and/or employing aging mitigation techniques.

The challenge is that there are several interdependencies between aging mechanisms that need to be carefully tackled to correctly estimate the degradations in the electrical characteristics of transistors over time as well as their deleterious effects on reliability. Given various aging mechanisms $M = \{m_1, m_2, ...\}$, the set of initial transistor characteristics[1] $\mathfrak{P}_{t_0} = \{p_1(t_0), p_2(t_0), ...\}$ (e.g., V_{TH}, etc.), environment parameters $E = \{\varepsilon_1(t), \varepsilon_2(t), ...\}$ (e.g. temperature, voltage noise, etc.), and a stress condition $S(t)$, aging can be expressed as a function $\mathcal{A}_{\mathfrak{P}^j} : M^n \times \mathfrak{P}^{|\mathfrak{P}|} \times E^{|E|} \times S \rightarrow \mathfrak{P}^j$, where $\mathfrak{P}^j$ is the set of j affected transistor characteristics degraded by n mechanisms. Assuming that the time dependencies for E and S are given, the time dependency for $p_i \in \mathfrak{P}$ can be expressed as

$$p_i(t) = \int_{t_0}^{t} \mathcal{A}_{p_i}(m_1, ..., m_n, \mathfrak{P}, E, S)(\hat{t})\ d\hat{t} \tag{3.1}$$

Due to this recursive dependency of $\mathfrak{P}$ on transistor characteristics, $\mathfrak{P}_1, \mathfrak{P}_2 = \mathfrak{P}_{t_k}$ at time $t_k > 0$ for $\mathcal{A}_{\mathfrak{P}}$ considering two non-empty subsets $M_1, M_2 \subset M$, respectively, are generally only equal ($\mathfrak{P}_1 = \mathfrak{P}_2$) if $M_1 = M_2$.
Modeling aging mechanisms separately results in a different estimation of transistor characteristics degradation than when modeling them simultaneously. For a given circuit state $\mathfrak{S}$ and behavior $\mathfrak{B}$, transistor characteristics can be *abstracted to a failure probability* and, by extension, this probability can be expressed through $M_i \subset M$:

$$P_f = P_{\mathfrak{S},\mathfrak{B},E,\mathcal{A}_{\mathfrak{P}}}(M_i) \tag{3.2}$$

Analog to the parameters, $\forall M_i, M_j \subset M; M_{i,j} \neq \{\emptyset\}$, $P_{\mathfrak{S},\mathfrak{B},E,\mathcal{A}_{\mathfrak{P}}}(M_i) = P_{\mathfrak{S},\mathfrak{B},E,\mathcal{A}_{\mathfrak{P}}}(M_j) \Leftrightarrow M_i = M_j$, at time $t_k > 0$. A key problem is that it is impossible for system-level designers to conceive how degradation of various $p_i(t)$ over time will interdepend to ultimately degrade the entire system's reliability, i.e. increasing the $P_f(Total)$.

[1]Initial values of characteristics at $t = t_0$ may vary from transistor to transistor due to manufacturing variability.

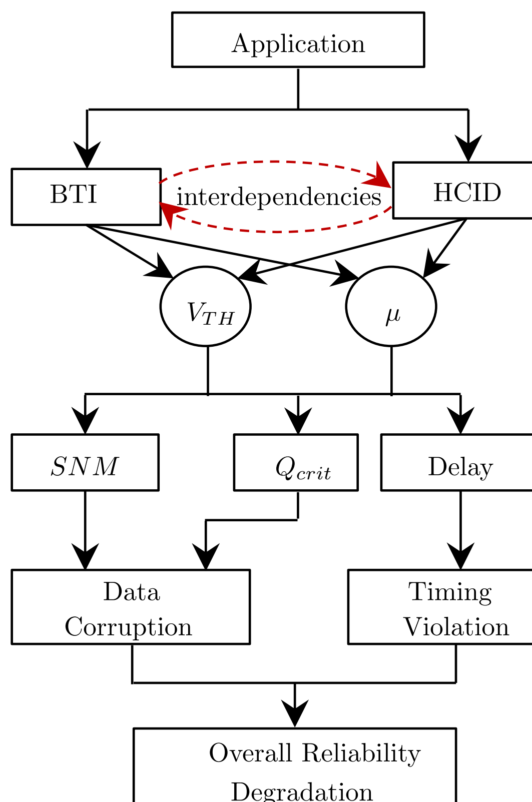


FIGURE 3.1: Our flow for superposing multiple aging effects

The key contributions within the proposed reliability estimation are:
(1) Combining from the physical to system level, the effects of multiple aging mechanisms occurring *simultaneously* based on their interdependencies and showing how considering a sole individual mechanism results in a non-negligible reliability underestimation.
(2) Abstracting the various degradations induced by aging (see Figure 2.2) along with radiation towards a probabilistic fault analysis which has a more meaningful interpretation of reliability, unlike state-of-the-art that employs other quantification metrics e.g., SNM, ΔV_{TH}, etc. which are hard for interpreting the overall reliability degradation of the entire on-chip system.

3.3 Degradations Modeling

In this section, we illustrate our proposed methodology of combing the effects of multiple aging mechanisms showing that the degradation of transistor characteristics is a superposition of multiple interdependent aging effects. It is worthy to note that while the defects induced by BTI and HCID at a given time (and given transistor characteristics) can be considered independent due to their different location in the dielectric (see Figure 3.1), the overall degradation of V_{TH} relies on the total number of induced defects. *Over time, this will lead to interdependency between BTI and HCID since the amount of defects induced by each is recursively dependent on V_{TH}.*

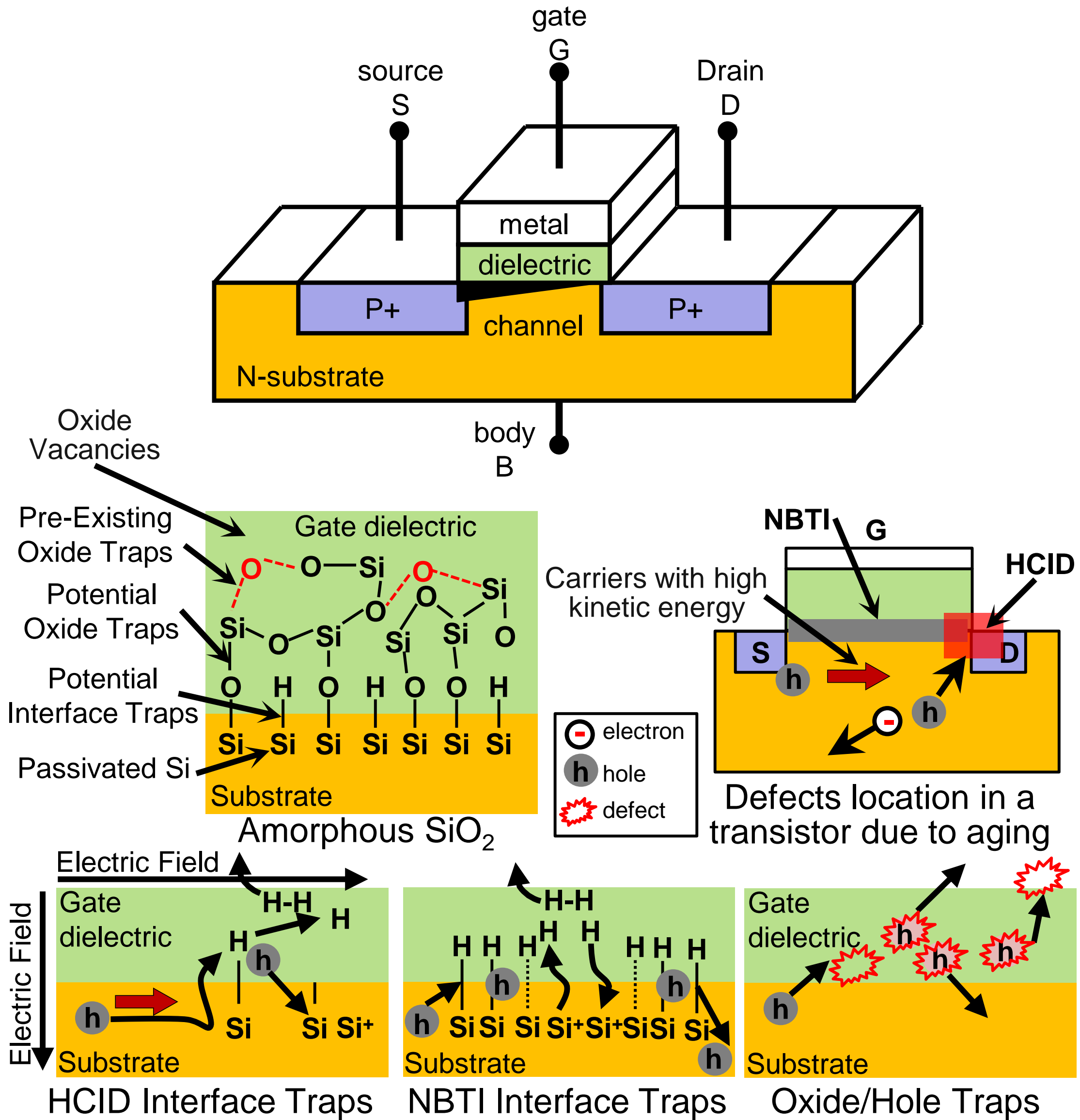


FIGURE 3.2: Key aging defects in a PMOS transistor

Finally, we present our abstraction of different kinds of failures caused by aging along with our implementation.

3.3.1 Defects due to Aging

BTI is mainly caused by continuous trap generation in the Si$-$SiO$_2$ interface of a transistor, whereas HCID is caused by *hot carriers*[2]. These carriers are caused by the strong electric field across the transistor's channel causing kinetic energies sufficient to form electron-hole pairs through impact of ionization, which may be injected in undesirable areas.

[2]*Hot* does not relate to transistor's temperature but to the ability to tunnel through of the semiconductor material.

In the following, we consider a PMOS transistor as an example for the sake of explication. However, defects in NMOS transistors are analogously induced but with the corresponding opposite charges.

Interface Traps (N_{IT}): These are caused by dissociating the Si−H bonds in the Si−SiO_2 interface due to the non-epitaxial structure of amorphous SiO_2 on the crystalline silicon. Further details can be found in [116] and Chapter 9.1. When an electric field between gate and source is applied, holes can be captured by an Si−H bond and combined with one of the two available electrons there resulting in weakening the bond. This can activate the hydrogen to break the bond and diffuses away. Overall, the rates of dissociation/healing Si−H bonds and diffusion of molecular hydrogen mainly determine the total number of unsatisfied silicon atoms (N_{IT}) over time. BTI is responsible for one physical Si−H dissociation mechanism. Additionally, the Si−H bonds can also be dissociated because of accelerated carriers in the MOSFET channel when an electric field between the drain and source is applied. The *hot carriers* lose their energy due to Coulomb scattering leading to dissociating Si−H bonds. Therefore, HCID is responsible for the additional *interface trap* defects.

Self-Healing/Recovery of Si−H Bonds: Healing the dissociated Si−H bonds is the opposite process to its dissociating. Actually, the dissociated bonds may almost be recovered if a sufficient relaxation time is given. However, a full recovery is impossible because the H_2 may leave the gate dielectric after reaching the metal gate. In such a case, the corresponding dissociated Si−H bond may not be healed anymore as it will lack for a bonding atomic H. Generally, the majority of dissociated Si−H bonds (i.e. due to BTI and/or HCID phenomenon) may immediately after heal because its H partners are still available beside the unsatisfied silicon atoms Si^+. It is noteworthy that the ratio between bond dissociating k_f and bond healing k_r, (i.e. $\frac{k_f}{k_r}$), ultimately determines the generation of N_{IT}. Further details about the aforementioned processes of dissociating and healing Si−H bonds are explained in [116].

Deleterious Impact of *Interface Traps* on the MOSFET Characteristics: Since the *interface traps* are, at the end, unsatisfied silicon atoms Si^+, they, indeed, lead to undesired positive charges that may be accumulated under the SiO_2−Si interface. The latter results in less and less attracted carries during the formulation of the conducting channel of MOSFET because of the weakener electric filed. This, in turn, manifests itself as a shift in the MOSFET threshold voltage (V_{TH}) because the required voltage, to turn the transistor on, becomes higher in order to provide the same electric field that was originally (i.e. before the generation of aging-induced defects) necessary to form the MOSFET channel.

Additionally, the accumulated aging-induced charges at the SiO_2−Si interface also result in a reduction in the carrier mobility (μ) across the MOSFET channel. This is because that the channel is formed from holes[3] and therefore the positive induced defects/charges and the holes within the channel will scatter each other. This makes the channel has more

[3]This is valid for the case of PMOS transistors which we employ as an example during our explanation of the *interface traps* generation. However, for NMOS transistors, the channel consists of electrons, instead of holes, and defects will be analogously induced but with the corresponding opposite charges.

scattering making the carriers within the channel be obstructed during their movement. As explained earlier in Section 1.5.2, both degradations in V_{TH} and μ lead to other degradations in the MOSFET electrical characteristics such as the drain current (I_D) and transconductance (g_m) due to the interdependencies of them.

Oxide Traps: These are partially because of *Pre-Existing* defects in the amorphous SiO_2 of the gate dielectric material during manufacturing. Such defects are unsatisfied bonds due to oxide vacancies, which are not electrically active due to their neutral atoms. However, these act as Hole Traps (N_{HT}) and, thus, are positively charged if a hole is trapped. Due their spatial absence from the channel, they do not affect the transistor carrier mobility like *interface traps*. Importantly, the number of hole traps is limited to the number of the unsatisfied bonds (from manufacturing) contrary to *interface traps* which are continuously generated over time due to abundance of the Si−H bonds that can be dissociated. Beside the *Pre-Existing oxide traps*, other Oxide Traps (N_{OT}) can also be induced over time due to the slow and irreversible dissociation of Si−O bonds [117] which are stronger than Si−H bonds. This increases the number of available *oxide traps* resulting in increasing the saturation point of hole traps within the transistor. Despite the experimental measurements observing *oxide traps* when HCID is analyzed, it has been proven that these traps are only induced by BTI which simultaneously occurs [51]. The defects caused by NBTI and HCID are shown in Figure 3.2.

Deleterious Impact of *Oxide Traps* on the MOSFET Characteristics:

Activated *oxide traps* also result in undesired positive charges within the gate dielectric. Importantly, these charges, contrary to the *interface traps*, are not accumulated near to the $SiO_2 - Si$ interface but, instead, deep within the gate dielectric. Therefore, the deleterious impact of *oxide traps* on the transistor mobility is neglected because they cannot interact with the carriers within the MOSFET channel, unlike *interface traps*. However, *oxide traps* still can weaken the electric field over the gate dielectric due to their positive charges. This again manifests itself as an increase in the MOSFET threshold voltage (V_{TH}).

It is worthy to note that as more *oxide traps* are generated over time the probability to have a conducting path between the two sides of the gate dielectric becomes higher. This results in a gate-leakage current which may burn the transistor up if it is sufficiently high due to breaking the gate dielectric down. In such a case, we say that the TDDB phenomenon has occurred. However, with the employment of the high-K material *oxide traps* are barely generated. The latter along with the increase in the thickness of the dielectric (compared to the technology before the high-K material) makes having the deleterious impact of TDDB is seldom[4]. Thus, the TDDB phenomenon is not within the focus of this thesis.

[4]The probability of to have a beak down due to TDDB in the current technology node of 22nm is smaller than 10^{-11} [C1].

3.3.2 Superposition of Aging Effects

Over time, the induced defects at the physical level will degrade the following key electrical characteristics of the transistor.

1. **Threshold Voltage Shift (ΔV_{TH})**:

   The induced defects result in undesirable charges in the gate dielectric of a transistor weakening the electric field between the gate and bulk. Therefore, the degradation manifests itself as an increase of V_{TH}. N_{IT}, N_{HT} and N_{OT} defects, induced by aging mechanisms over time, will contribute to ΔV_{TH} and the role of each one in weakening the electric field depends on the number of present defects.

   As a matter of fact, each defect within the gate dielectric will contribute to the charge buildup with a single carrier and each carrier, in turn, contributes with its elementary charge q. Thus the threshold voltage shift (ΔV_{TH}) can be modeled through the relative shift of the gate capacitance $\frac{C_{defects}}{C_{ox}}$, which leads to the following equation according to the RD model:

$$\begin{aligned} \Delta V_{TH}(t) &= \frac{q}{C_{ox}} \cdot (\Delta N_{IT}(t) + \Delta N_{HT}(t) + \Delta N_{OT}(t)) \\ \text{with} \quad & \Delta N_{IT}(t) = \Delta N_{IT.BTI}(t) + \Delta N_{IT.HCID}(t) \end{aligned} \tag{3.3}$$

   To obtain the number of required number of defects induced by each aging mechanism, we modified the analytical solutions of the differential equations [14] that describe trapping mechanisms, with the factor d to take into consideration the recovery of *interface traps* (that occurs when the voltage stress ceases) based on [116, 118]. We also introduced HCID on top of BTI via the combination of their simultaneous occurring physical process (i.e. N_{IT} generation) based on [119, 120]:

Interface Traps:

$$\begin{aligned} \Delta N_{IT.BTI} &= A(V_{GS} - V_{TH} - \Delta V_{TH})^{\Gamma_{IT}} e^{\frac{-E_{AIT}}{kT}} \cdot t^{n_1} d^{n_1} \\ d &= \frac{\Lambda}{1+\sqrt{\frac{1-\Lambda}{2}}};\ E_{AIT} = \frac{2}{3}(E_{Akf} - E_{Akr}) + \frac{E_{ADH2}}{6} \\ \Delta N_{IT.HCID} &= B\left[t \cdot d \cdot \frac{I_{DS}}{W} \cdot e^{-\frac{\Phi_{IT,e}}{q\lambda_e E_m}}\right]^{n_2} \\ \text{with } E_m &= \frac{V_{DS} - V_{DSAT}}{\sqrt{\frac{\epsilon_{Si}}{\epsilon_{SiO_2}} \cdot t_{ox} \cdot x_j}} \text{ and } V_{DSAT} = \frac{(V_{GS} - V_{TH}) \cdot L \cdot E_{cr}}{V_{GS} - V_{TH} + L \cdot E_{cr}} \end{aligned}$$

Oxide Traps:

Stress Phase:

$$\begin{aligned} \Delta N_{HT} &= C(V_{GS} - V_{TH} - \Delta V_{TH})^{\Gamma_{HT}} \cdot e^{\frac{-E_{AHT}}{kT}} \\ & \cdot (1 - e^{-(\frac{t}{\tau})^{\beta_{HT}}}) \\ \Delta N_{OT} &= D(1 - e^{-(\frac{t}{n})^{\beta_{OT}}}) \\ n &= \eta(V_{GS} - V_{TH} - \Delta V_{TH})^{-\frac{\Gamma_{OT}}{\beta_{OT}}} \cdot e^{\frac{E_{AOT}}{kT\beta_{OT}}} \end{aligned}$$

Recovery Phase:

$$\Delta N_{HT} + \Delta N_{OT} = E(e^{-(\frac{t}{\tau_r})^{\beta_{HTR}}})$$

details in Chapter 9.1.1

As this is not our main scope, in-depth explanation of the employed physical aging models can be found in [14, 73, 116, 118, 120]. It is worthy to note that BTI-induced defects are evenly distributed across the $Si-SiO_2$ interface, where the electric field is homogeneous, contrary to the HCID due to defects which are concentrated near the drain (see Figure 3.2).

2. **Carrier Mobility (μ) Degradation**:

   As explained, the induced defects harmfully impact the mobility of carriers within the transistor channel, as the charged defects can interact with the carriers impeding their passage through the channel leading to degrading the transistor's carrier mobility. Because *hole/oxide traps* (N_{HT}/N_{OT}) are located *deep* inside the gate dielectric away from the channel, they have a negligible impact on the carrier mobility and, thus, only *interface traps* have to be considered here. Similarly as [121], we model the μ degradation as follows:

$$\mu(t) = \frac{\mu_0}{1 + \alpha \cdot \Delta N_{IT}(t)} \quad (3.4)$$
$$\text{with} \quad \Delta N_{IT}(t) = \Delta N_{IT.BTI}(t) + \Delta N_{IT.HCID}(t)$$

   Where μ_0 is the initial carrier mobility (i.e. $\mu_0 = \mu(t=0)$) and α is a device dependent parameter. This empirical relationship is still the basis for many mobility degradation models [73] and just α needs to be experimentally determined for every new technology node.

3. **Drain Current (I_D) Degradation**:

   As a matter of fact, both V_{TH} and μ degradations have a direct impact on the drain current which represents the key electrical characteristic of a MOSFET. The following equation exemplifies the existing relation between V_{TH}, μ and I_D:

$$I_D = \mu \cdot \frac{C_{ox} \cdot W}{2 \cdot L} (V_{GS} - V_{TH})^2 \cdot (1 + \Phi V_{DS}) \quad (3.5)$$

   In practice, we actually rely – in our aging analysis – on a more accurate/sophisticated model of I_D to tackle the aforementioned dependency, provided through the recent BSIM models (see Chapter 5 of [122]) that SPICE employ. Similarly, other MOSFET electrical characteristics such as its transconductance (g_m) (see Section 1.5.2 and Eq 1.3), that may be influenced due to the V_{TH} and μ degradations, can be considered during our aging analysis.

In Summary, the degradation of transistor characteristics is due to both BTI and HCID and the overall impact of aging is a superposition of their interdependent effects. State-of-the-art approaches [71, 74] look at V_{TH} solely when analyzing reliability. We show later in Section 3.6.2 (see Figure 3.12) why it is necessary to additionally take μ into account to avoid underestimating aging-induced degradations.

3.4 Reliability Abstraction

In the following, we present how we deal with the aging-induced degradations occurring at the device/circuit level to provide a *reliability abstraction* summarizing their impacts on the susceptibility to failures at the system level (see Figure 3.1). As an example, we apply our reliability abstraction to SRAMs due to their susceptibility to different reliability aspects (e.g., noise, radiation, delay, etc.) and due to their total chip area that may reach up to 70% [123]. Additionally, SRAMs are typically used to implement register files of microprocessors as earlier discussed (further details are in Sections (1.1 and 1.2)) which estimating as well as increasing its reliability is one of the main focuses of this thesis as explained in Chapter 1. It is noteworthy that our proposed reliability estimation is not restricted to a specific kind of circuits (e.g., SRAM cells) and it can also be applied to others such as computational units.

3.4.1 Data Corruption

Data in SRAM cells can be corrupted because of the voltage noise from neighboring circuits transferred over parasitic capacitances, supply voltage, etc. As earlier explained in Section 1.2, the *Static Noise Margin* (SNM) quantifies the resiliency of the SRAM against noise through the *butterfly* curve that describes the transfer characteristics of the cross-coupled inverters within the SRAM (see Figure 1.2). Thus, it provides a metric for data correctness and it is widely used in research for that purpose.

Aging-induced V_{TH} degradation will shift the *butterfly* curve shrinking the size of the square within and, thus, degrading the SNM of the aged SRAM, as shown in Figure 2.4. Indeed, SNM degradation reduces the SRAM resiliency resulting in an increased failure probability due to data corruption ($P_f(Data)$). Additionally, radiation can also corrupt the SRAM data when a particle deposits its energy through an SRAM resulting in an electrical current spike. The *transconductance* (g_m) of a transistor determines if the generated spike will induce charges above the Critical Charge (Q_{crit}) (i.e. the minimum amount of charge required to flip/corrupt stored data). As discussed in Section 1.5.2, both degradations in V_{TH} and μ result in degrading g_m (see Equation 1.3) and, therefore, aging increases the probability of failure due to soft errors ($P_f(Qcrit)$).

3.4.2 Timing Violations

An SRAM in a (synchronous) design can cause timing violations i.e. it fails to provide correct data in time. The capability of the SRAM (for a given sense amplifier) to drive its bitlines within timing constraints depends on the I_D of the SRAM transistors which is reduced by the degradation of ΔV_{TH} and μ. In other words, aging will result in a longer *read access time* (RAT) in the SRAM[5] increasing the failures due to timing violations ($P_f(Timing)$). Figure 3.1 summarizes how aging increases the system's susceptibility to failures.

[5]Unlike write access time which is improved by aging [124].

3.4.3 Interpreting Aging-induced Degradations to Failure Analysis

Figure 3.16 illustrates our implementation to abstract the impact of multiple simultaneous aging mechanisms on the probability of failures. As shown, the model of NBTI/PBTI together with HCID is employed to degrade the affected transistor characteristics based on the Predictive Technology Model (PTM) [125–127] and the BSIM [128] that is utilized to address the varied interdependencies between of the MOSFET characteristics (e.g., how ΔV_{TH} & μ influence I_D & g_m etc.). Then, device-level characteristics and the corresponding circuit-level metrics (i.e. SNM, RAT, and Q_{crit}) are computed.

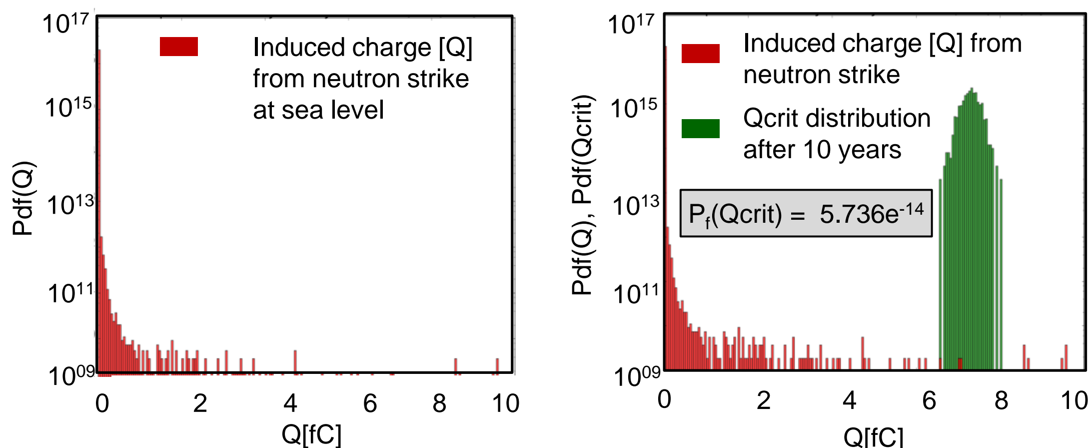


FIGURE 3.3: Probability density function (Pdf) of both Q and Q_{crit} to calculate the probability of failure when a particle strikes an SRAM cell $P_f(Qcrit)$

There is a variation in device/circuit-level metrics due to manufacturing variability, resulting in varying susceptibilities to failures. The developed manufacturing variability modeling was provided from semiconductor industry and corresponds to a normal distribution for the transistor dimensions.

The Passage of Particle Through Matter (Geant4) particle simulator [129] along with a distribution of energy/flux of neutrons [130] and SRAM layers information [131] have been employed to obtain the charges deposited in an SRAM [132]. Then, the deposited charges distribution in conjunction with the Q_{crit} distribution akin to aging after the targeted lifetime (e.g., 10 years) is used to compute the failure probability due to soft error $P_f(Qcrit)$ as shown in Figure 3.3 and later in Figure 3.7(a) but when aging phenomena are considered. In practice, $P_f(Qcrit)$ is calculated by folding the Probability Density Function (Pdf) of the particle strikes ($Pdf(Q)$) with the Pdf of the Q_{crit} ($Pdf(Q_{crit})$) as follows:

$$P_{fail}(Q_{crit}) = P(Q \geq Q_{crit}) = Pdf(Q) * Pdf(Q_{crit}) \tag{3.6}$$

It noteworthy that the Y-axis of Figure 3.3 shows the Pdf and therefore there is no unit there because these numbers are numeric values and have no meaning alone as we always

need to integrate the probability density function over an interval to get a probability. We can see well from the orders of magnitude how tremendously improbable is to get high critical charges. This is reasonable as we analyze soft errors at the sea level.

When analyzing failure probabilities due to noise or timing, it is important to consider the employed safety margins. These are chosen by the system designer to allow for variability within the design (either at the beginning due to manufacturing or later on due to aging-induced degradations). If the resulting SNM/RAT degradation exceeds the corresponding noise/timing safety margins, failures start to occur more frequently. In practice, after calculating the distribution of both SNM and RAT, the corresponding probability of failure $P_f(SNM)$ and $P_f(RAT)$, respectively, are calculated according the employed safety margins. In mathematical terms the broken SRAM cells, due to SNM degradation, are determined as follows:

$$SNM_{Threshold} = SNM_{Reference} - (SNM_{Reference} \cdot SM_{SNM}) \tag{3.7}$$

$$SNM_{sram}(sample) < SNM_{Threshold} \rightarrow \text{broken cell} \tag{3.8}$$

$$SNM_{sram}(\text{sample}) \geq SNM_{Threshold} \rightarrow \text{intact cell} \tag{3.9}$$

Where, SM_{SNM} is the safety margin of SNM.

Then, the $P_f(SNM)$ is calculated as:

$$P_{fail}(SNM) = \frac{\#\text{ broken cells}}{\#\text{ broken cells} + \#\text{ intact cells}} \tag{3.10}$$

On the other hard, broken SRAM cells, due to RAT degradation, are determined as follows:

$$RAT_{Threshold} = RAT_{Reference} + (RAT_{Reference} \cdot SM_{RAT}) \tag{3.11}$$

$$RAT_{sram}(sample) > RAT_{Threshold} \rightarrow \text{broken cell} \tag{3.12}$$

$$RAT_{sram}(sample) \leq RAT_{Threshold} \rightarrow \text{intact cell} \tag{3.13}$$

Where, SM_{RAT} is the safety margin of RAT.

Analogously to SNM, the probability of failure due to RAT degradation can be calculated as follows:

$$P_{fail}(RAT) = \frac{\#\text{ broken cells}}{\#\text{ broken cells} + \#\text{ intact cells}} \tag{3.14}$$

In our scenario (register file of a processor) SRAM cell often has half a clock cycle to deliver its data to the output lines. This is necessary for register files to support

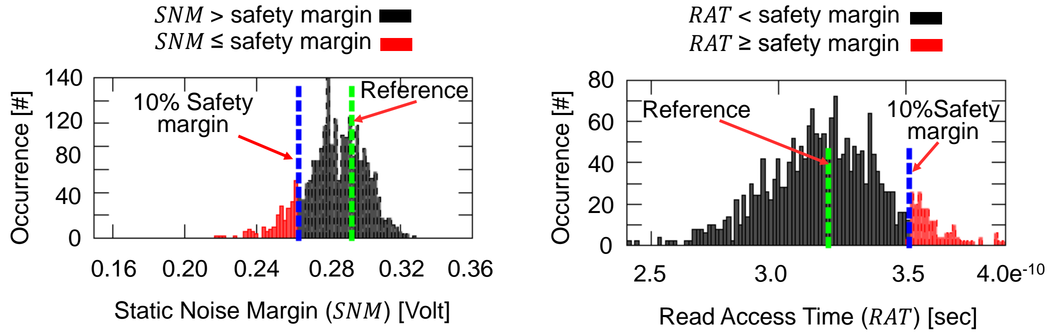


FIGURE 3.4: Employed safety margins define if the aging-induced degradations can be tolerated or they cause failures

multiple accesses per clock cycle. For instance, on the rising edge of clock, data is read and on the falling edge of a clock data is written. Therefore, $RAT_{sram}(sample) > \frac{T}{2} \rightarrow$ broken cell with $T =$ clock period.

Figure 3.4 presents an example in terms of data corruption failures and timing violations and how the selected safety margins of SNM and RAT can be applied to the aging-induced SNM and RAT distributions to calculate $P_f(SNM)$ and $P_f(RAT)$, respectively.

Then, the total failure probability is expressed as:

$$P_f(Total) = \overbrace{P_f(SNM) + P_f(Qcrit)}^{P_f(Data)} + \overbrace{P_f(RAT)}^{P_f(Timing)} \tag{3.15}$$
$$- \overbrace{P_f(SNM \cap Qcrit) - P_f(SNM \cap RAT) - P_f(RAT \cap Qcrit)}^{P_f(Data \cap Timing)}$$
$$+ P_f(SNM \cap Qcrit \cap RAT)$$

Finally, the aforementioned steps are analogously repeated for different operating conditions, e.g., temperatures ($T \in [25, 125]$), aging stresses ($\lambda \in [0, 1]$) and supply voltages ($V_{dd} \in [1.2, 0.7]$), to build a database that provides designers with fast lookup-based reliability estimation at the system level through a wide range of operating conditions.

3.4.4 Implementation Details of Our Reliability Estimation

To estimate RAT: The SPICE simulator[6] calculates how long reading the stored value '0' from the SRAM takes.

To estimate SNM: The butterfly curves of the SRAM cell is plotted based on SPICE simulations in order to get the upper and lower squares (boxes), as demonstrated in Figure 1.2(b). The stored value does not play any role here and only the electrical characteristics of the transistors within the SRAM cell determine the shape of the butterfly and thus the SNM value of the SRAM.

[6]The ngspice [133] has been employed in all presented results within this thesis.

To estimate the Q_{crit}: A current source at the data point of the SRAM cell, based on the presented model in [134], is inserted within the netlist of the SRAM cell as an equivalent circuit.

$$I(t) = \frac{Q}{t_{fall} - t_{rise}} \cdot [e^{-\frac{t}{t_{fall}}} - e^{-\frac{t}{t_{rise}}}] \tag{3.16}$$

An iterative approach based on the SPICE simulation, then, is employed to calculate minimal charge to corrupt the stored data within the SRAM cell. We start first from a minimal deposited charge Q and the SPICE simulation checks if the data stored at A (see Figure 1.2(a)) was corrupted or if the SRAM cell could tolerate the inserted current (i.e. deposited charge). If the data was still valid, the Q is increased and the process repeated. Once the data gets corrupted, the minimal charge to corrupt the data will be determined (i.e. Q_{crit}).

To model the impact of temperature: Temperature modeling is based on three components. The first one is the modeling of transistors through the last recent version of BSIM modeling from Berkeley [128]. The second component is calibrating the BSIM model to current high-K 22 nm technology with characteristics provided by the Predictive Technology Model (PTM). The last component is using BSIM with the SPICE front-end. Concisely, our temperature modeling is at the device level (i.e. through SPICE along with the employment of BSIM models) after obtained the required transistor characteristics from the 22nm PTM model. It is noteworthy that the temperature dependence model inside BSIM is detailed/sophisticated as the Chapter 13 of its manual describes [122].

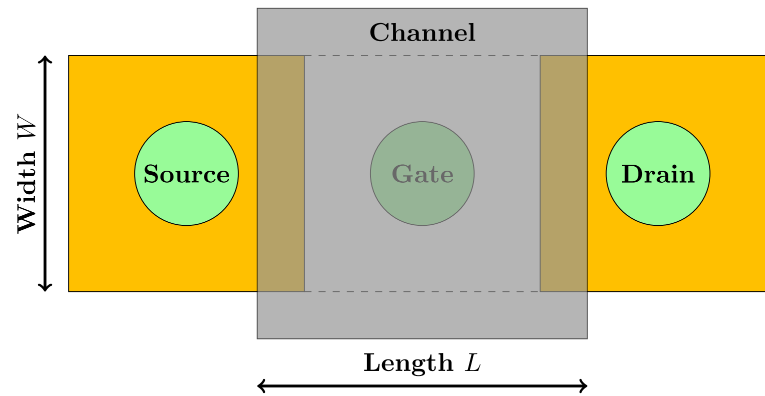
To model the impact of manufacturing variability: As motivated in Chapter 1, manufacturing variability is a variance in the transistor characteristics due to the manufacturing processes of chips. In the nano-CMOS era, the manufacturing process introduces considerable fluctuations in transistors geometries [135] and therefore their impact needs to be taken into account, when estimating reliability. In Figure 3.5(a), the width W and the length L of a transistor are shown. It is often that L is constant at a specific technology node, e.g. 22 nm, while, W might be chosen by the chips' designers.

We model the manufacturing variability as a fluctuation of the transistor geometries based on a normal distribution [7]. Then, the resulting effects such fluctuations on the electrical characteristics of MOSFET are addressed through the BSIM modeling.

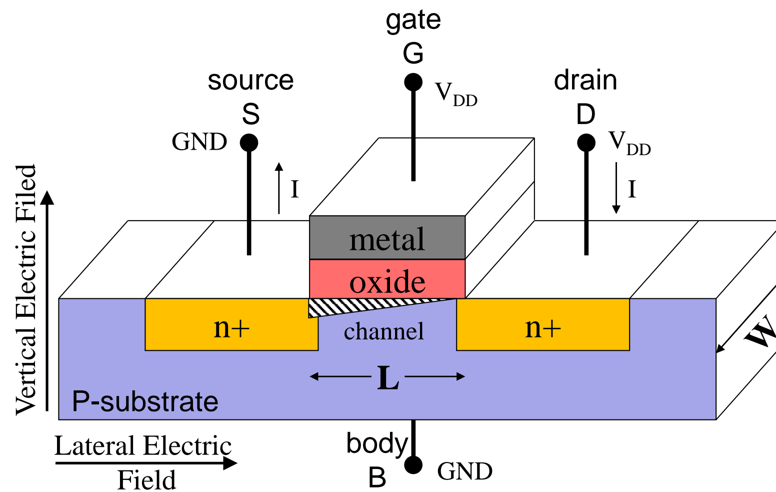
3.4.5 Failure Probability Estimation

Figure 3.6(a) shows, for the case of *duty cycle* $\lambda = 0$, the increase of $P_f(Total)$ depending on the chosen safety margins that determine if the induced degradations can be tolerated or not (see Section 3.4). As shown, $P_f(Total)$ exponentially decreases with higher safety margins which, in turn, directly increase the device's cost. For instance, a higher SNM safety margin to cope with aging-induced SNM degradation necessitates building more

[7] In the experimental results within this chapter and later in the evaluations within Chapter 6, we employ a standard deviation (σ) of 0.8 while modeling the effects of process variation.



(a) A top view of an nMOSFET



(b) A 3D view of an nMOSFET

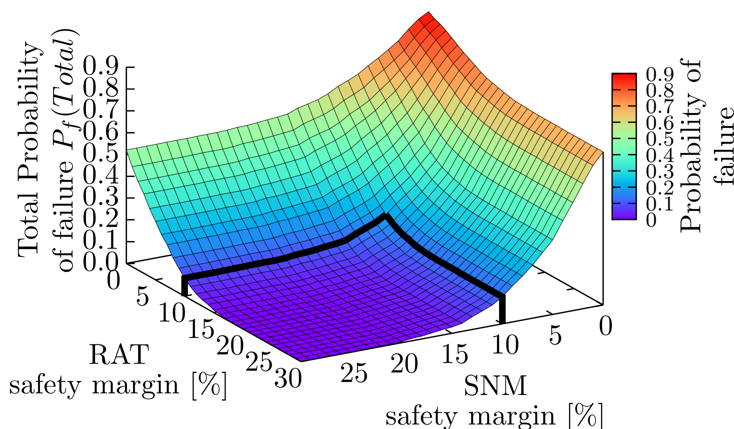
FIGURE 3.5: Transistor geometries

robust SRAM sense amplifiers, which can negatively affect the area/power budget. On the other hand, higher *RAT* safety margin leads to selling the device at lower frequency to avoid aging-induced timing violations during its lifetime. Therefore, such an analysis in Figure 3.6(a) can guide the designer to choose appropriate safety margins that maintain a reliable operation in the presence of aging. Figure 3.6(b) clarifies, for the case of 10% safety margins, that multiple simultaneous aging mechanisms can increase $P_f(Total)$ up to 42% over 10 years on top of the failures due to manufacturing variability.

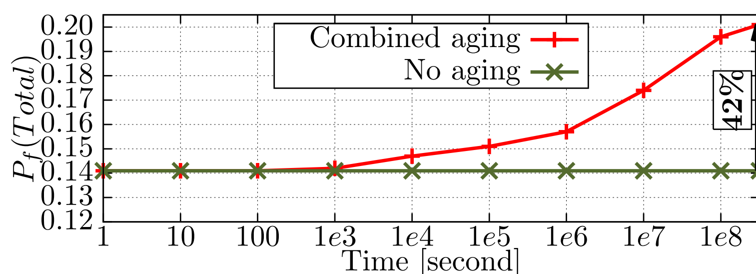
In Figure 3.7(a), we show the resulting degradation in Q_{crit} along with the distribution of electrical charge deposited when an energetic particle strikes the device. These distributions can be used to derive the probabilities of soft error during the device's lifetime in the presence of aging.

Due to just a small number of charges generated above the Q_{crit}, the failure probability is very low. This is mainly due to analyzing soft error under typical operation conditions i.e. neutrons at the sea level [8] together with $V_{DD} = 1.0\,\text{V}$ that results in low Q_{crit} shifts and, therefore, just a small change in soft error sensitivity. In Figure 3.7(b), these

[8] Analyzing soft error due to other kinds of particles or at higher altitudes, where higher fluxes are available, can result in higher $P_f(Qcrit)$.



(a) $P_f(Total)$ analysis under different safety margins



(b) Impact of aging in the presence of manufacturing variability

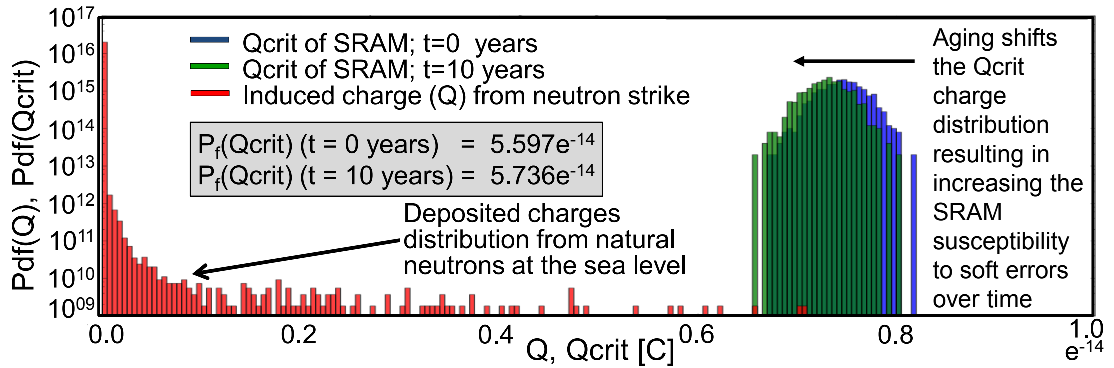
FIGURE 3.6: Failure analysis from our proposed implementation

probabilities are combined with a neutron flux of to compute the expected number of soft errors per year. As shown, aging can increase soft error rate by 2.4%.

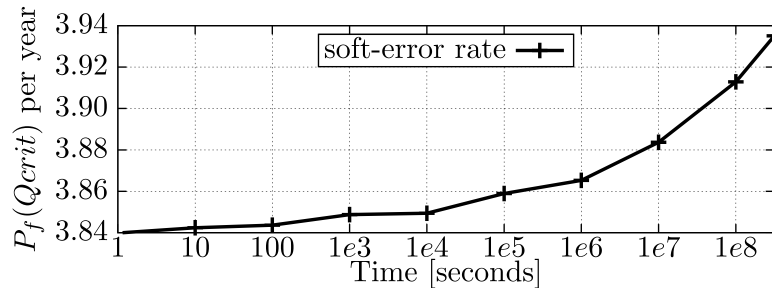
3.5 Validation

Since we do not have our own experimental data to validate our results, we rely on the published measurements in [14] to validate the V_{TH} degradation due to NBTI and PBTI. Then to validate the mobility degradation due to NBTI, we reply on the presented measurements in [15]. Finally, to validate the drain current degradation – which represents the key MOSFET characteristic – due to HCID as well as due to the combined effect of PBTI and HCID, we compare our results against state-of-the-art measurements from STMicroelectronics [16].

Figure 3.8 shows the ΔV_{TH} due to NBTI and PBTI. The presented device dependent parameters in [14] enabled us to match their experimental results accurately. This ensures that employed modeling of *interface traps*, *pre-existing oxide traps* and *generated oxide traps* provide proper results as the ΔV_{TH}, obtained from our model presented in Equation 3.3, has a good agreement with experimental measurement results.



(a) Empirical probability density functions (Pdfs)



(b) Effect of aging on the rate of radiation induced-soft errors

FIGURE 3.7: Soft error analysis in the presence of aging

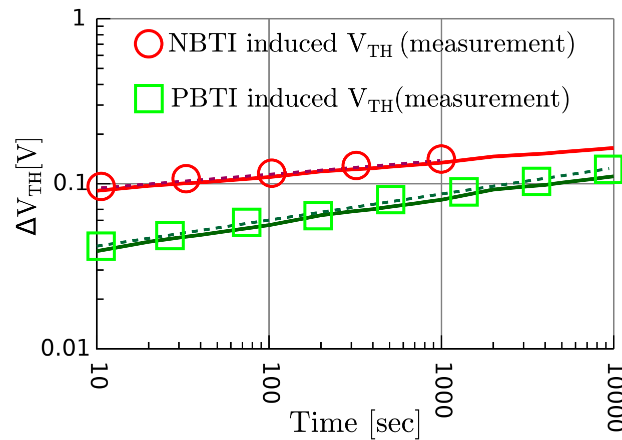


FIGURE 3.8: Validation of V_{TH} shift due to NBTI and PBTI against measurements published in [14]

In Figure 3.9 the mobility degradation validation is presented. The experimental result shows a slight deviation from the prediction by equation 3.4 deviation was needed to have later a good match in the drain current validation as the latter represents the key transistor characteristic, from the MOSFET operation perspective, and thus it has the highest priority in our work.

Finally, in Figure 3.10 the drain current degradation is demonstrated. This is important, as the I_D degradation is the ultimate impact of the different aging-induced degradations at the device level because it includes both V_{TH} as well as μ (see Equation 1.2). Our modeled prediction matches the experimental results for PBTI very well. Within the

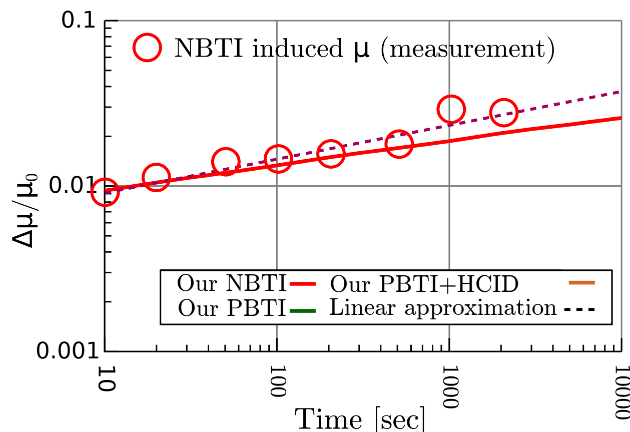


FIGURE 3.9: Validation of mobility (μ) against measurements published in [15]. A slight mismatch was needed to get a good matching in terms of drain current validation presented in Figure 3.10, as the latter has the highest priority due to its prime impact on the MOSFET operation

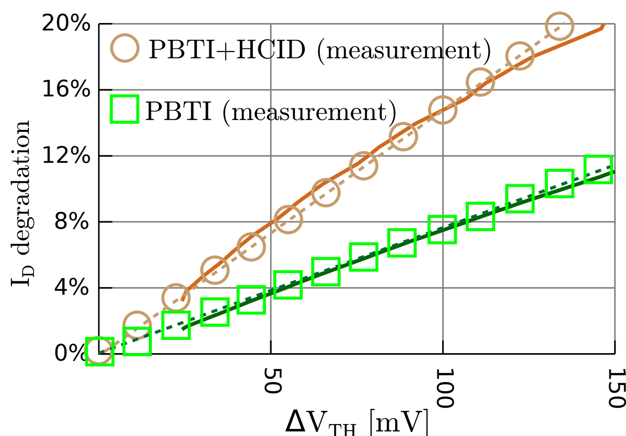


FIGURE 3.10: Validation of the combined aging model against industrial measurements [16]. PBTI alone in green boxes and PBTI *simultaneously* occurring with HCID in brown circles

same figure, the I_D degradation due the simultaneous occurrence of both PBTI and HCID is also presented. As it can be seen, our drain current degradation presents a close albeit not perfect match (i.e. there is around a 6% deviation). As our drain current degradation matches the experimental data well, it can be concluded that our proposed methodology presented in Section 3.3.2 (i.e. $N_{IT}/N_{OT}/N_{HT} \rightarrow V_{th}/\mu \rightarrow I_D$) properly combines the effects of multiple simultaneous aging phenomena.

3.6 Aging Effects Analysis

3.6.1 Transistor Electrical Characteristics

Figure 3.11 presents the corresponding V_{TH} and μ degradations over time, due to multiple aging mechanisms that occur either *separately* or *simultaneously* (the latter is the focus within the work), with respect to the induced defects under different cases.

As shown, the deleterious impact of PBTI on the transistor characteristics is quite small (but not negligible) in comparison to other aging mechanisms, even though the number of induced defects is higher (see Figure 3.11(a)). This is because these defects interact weaker with the carriers in the channel. As shown in Figure 3.11(b) NBTI initially shifts V_{TH} more than HCID because of the *pre-existing oxide traps* which come from manufacturing and, additionally, inducing *oxide traps* as well as *interface traps* (see Section 3.3.1).

Because the electrical activation of *hole traps* saturates over time [14], the BTI-induced ΔV_{TH} is dominated by *interface traps* in the long term. On the other hand, μ degradation only depends on charges in the proximity of the transistor's channel (i.e. *interface traps*). Thus, BTI-induced *oxide traps* play a weaker role here. Therefore, both BTI and HCID have a similar μ degradation over time (see Figure 3.11(c)). Shifts due to multiple simultaneous aging effects within Figure 3.11 are up to 6%.

It is worthy to note that the impact of simultaneous aging effects cannot be fully grasped at this abstraction level as motivated in section 2 and thus further evaluation at the system level is done later in Chapter 6.

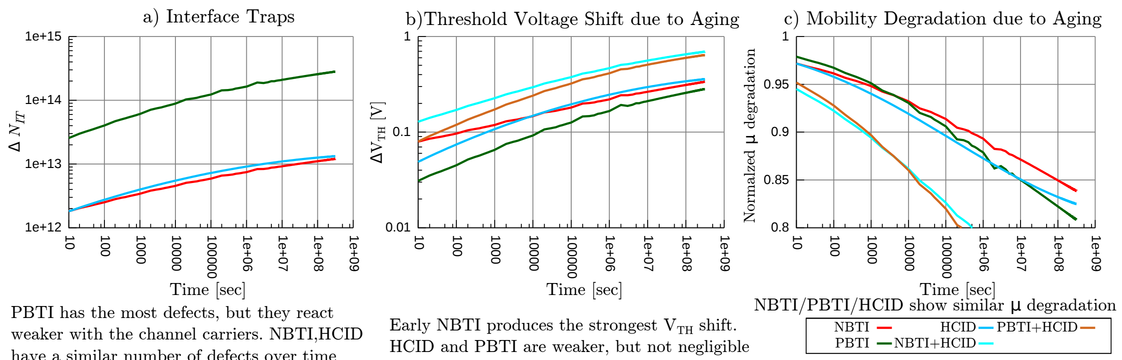


FIGURE 3.11: Transistor degradations due to BTI and HCID aging mechanisms *separately* / *simultaneously* considering

3.6.2 Impact of Considering Carrier Mobility

For a fair comparison, we select the work [72] as it follows similar goals. Additionally, other state-of-the-art often employ its concept (e.g., [71]) to estimate reliability when multiple aging mechanisms are targeted. As explained in Section 2, [71, 72] mainly consider the *dominant* aging mechanism in each transistor of the studied circuit. Figure 3.12 presents a *RAT* degradation[9] of an SRAM at $\lambda = 0.5$. It shows a noticeable deviation over time compared to our proposed *simultaneous* aging combination that ignore neither PBTI in NMOS nor HCID in PMOS. This establishes why it is vital to not rely only on the dominant aging mechanism when reliability analysis is performed to avoid underestimation. Additionally, we show how solely considering V_{TH} in analysis [71, 74] can significantly underestimate degradation. Therefore, examining μ together with V_{TH} as our implementation does (see Section 3.3.2), is indeed essential.

[9]Delay analysis has been chosen here for consistency's sake with [72]

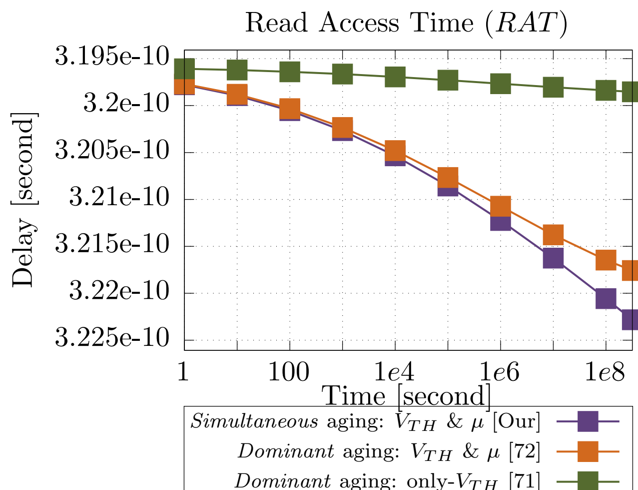


FIGURE 3.12: Comparison between our proposed combination of multiple *simultaneous* aging mechanisms and state-of-the-art

3.6.3 Abstracting Aging Effects at the System Level

As motivated in chapter 1, increasing the register file reliability is the key objective of this thesis. Therefore, we estimate within this Section the impact of aging on degrading the entire register file reliability. For that purpose, diverse applications from the MiBench benchmark suite [27] have been employed which exhibit varying characteristics enabling us to explore many different possible stress scenarios in the register file of the MIPS architecture. Its register file consists of 32 registers along with a bit-width of 32. However, our reliability estimation can easily be performed for other architectures, as it will be explored later in Section 6.4. We assume in the following estimation experiments safety margins of 10% for both SNM and RAT which represent a good compromise as Figure 3.6(a) illustrates.

Figure 3.13 illustrates the discrepancy that arises from considering solely an individual, *instead of multiple simultaneous*, aging mechanisms even through examining both kinds of failures. As it can be observed, considering NBTI as the most dominant mechanism and, thus, ignoring PBTI and HCID results in an underestimation of 7%, on average. Importantly, we present in Figure 3.14 the serious impact of considering NBTI as an individual dominant aging mechanism together with looking at only failures due to SNM degradation as state-of-the-art techniques (e.g., [74]) do when they estimate the reliability of register files. In such a case, the underestimation reaches, on average, 75% and up to 85%. Both Figures (3.13, 3.14) demonstrate that considering NBTI as the most dominant aging mechanism and, thus only taking it into account [74], is insufficient to estimate the overall reliability. This is even more evident when one failure source is alone examined leading to missing a non-negligible part of failures stemming from other degradations. Therefore, performing an accurate reliability estimation necessitates analyzing different failure sources that are induced by multiple simultaneous aging mechanisms.

Examples of the register file failure maps, obtained from our in-house reliability estimation (see Figure 3.16), are presented in Figure 3.15(a) for the *patricia* benchmark as well as the corresponding failure probability distribution in Figure 3.15(b). The samples with the higher probability of failures in Figure 3.15(b) correspond to a group of SRAM cells which suffer more from aging (also seen in the stress map of the register files SRAM cells, for the same benchmark, in Figure 3.16(left)).

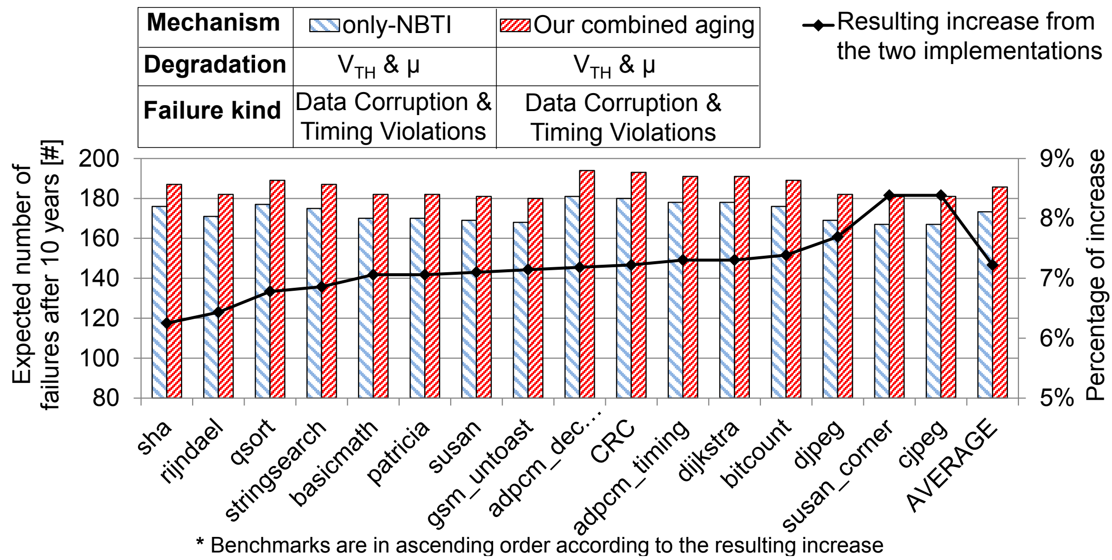


FIGURE 3.13: Register file reliability estimation comparison showing the underestimation when only an individual aging mechanism is considered even through examining both kinds of failures

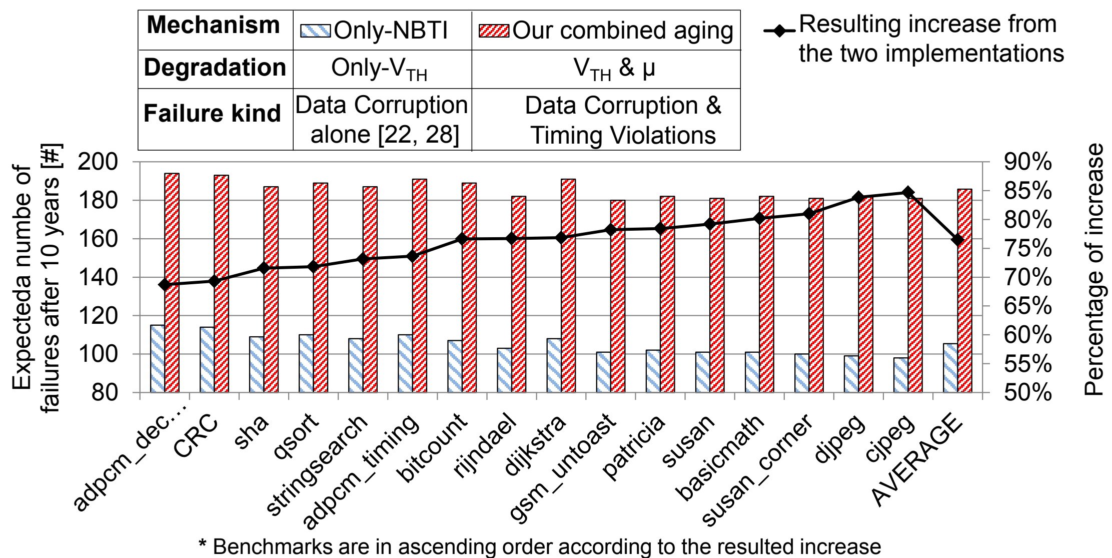
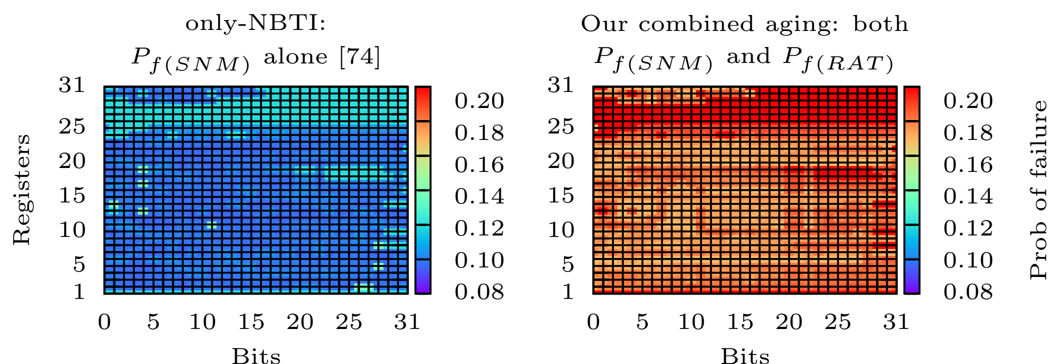


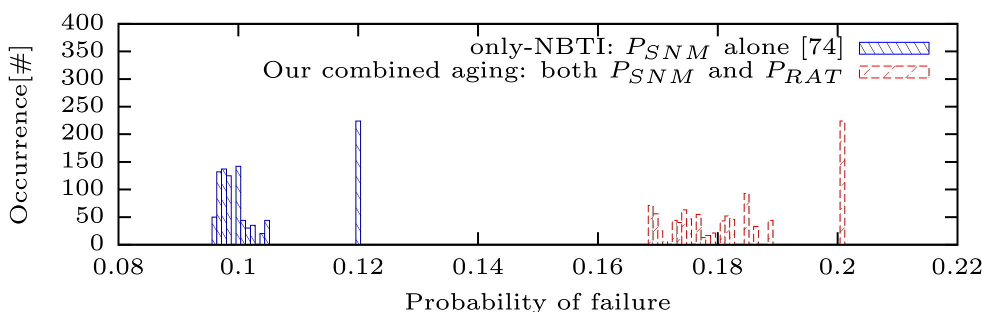
FIGURE 3.14: Register file reliability estimation comparison showing the underestimation when only an individual aging mechanism is considered together with examining only a single kind of failures

All in all, the above analysis can be used by the designer to obtain an abstracted, yet accurate reliability estimation of how defects, induced by multiple simultaneous aging

mechanisms, at the physical level can ultimately increase the probability of failure at the system level. Moreover, it can be used to find a compromise between cost of the chip and its reliability by exploring different design choices (i.e. safety margins).



(a) Example of the register file failure map



(b) Probability of failure distributions

FIGURE 3.15: Failure analysis of the register file SRAM cells

3.6.4 Limitations:

Attaining the reliability abstraction in terms of probability of failure through the modeling of physical defects comes at higher computational time compared to concepts discussed in 2 where the interdependencies of aging mechanisms are not taken into account. Having a meaningful interpretation of reliability at the system level together with examining different kinds of failures compensate for this, as long the underlying transistor characteristics (e.g., dimensions, manufacturing variability, etc.) do not exhibit a wide variance across the design, which would require multiple abstractions. Likewise, the approach benefits from regularity in the design (i.e. the reuse of circuits). It is worthy to note that when database of failure probabilities (see Section 3.4) is obtained, fast lookup-based reliability estimations can be performed as long as the properties of circuit (i.e. different technology node, schematic, etc.) remain the same. Our implementation is currently limited to the most three dominant aging mechanisms (NBTI, PBTI and HCID). The interdependencies with, e.g. TDDB, which is less pronounced, as motivated in Section 1 and discussed in Section 3.3.1, necessitates following analogously our process presented in Section 3.3.2 and Figure 3.1.

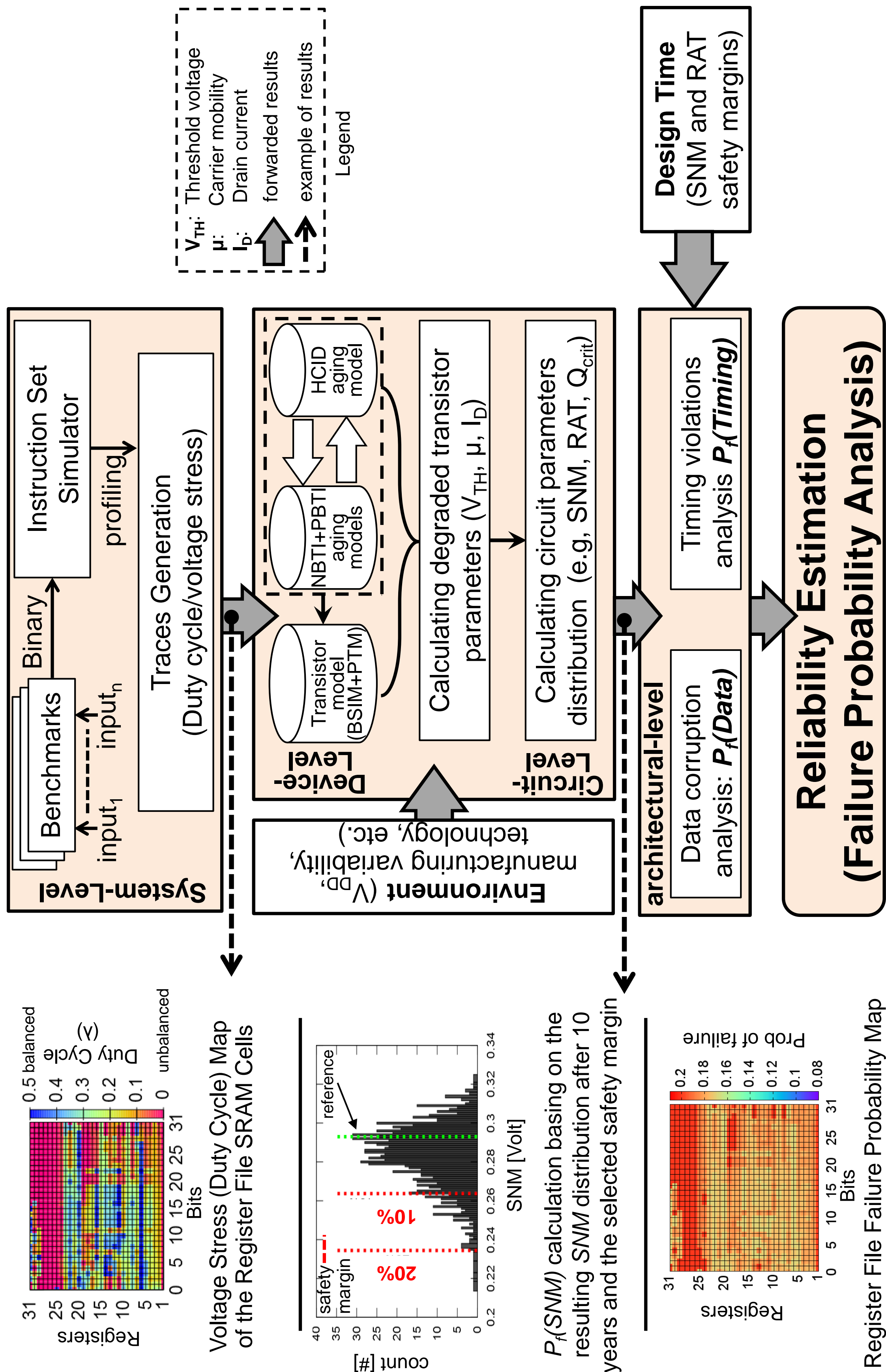


FIGURE 3.16: Our proposed in-house implementation to bridge the gap between application-induced stress at the system level and induced defects at the physical level together with abstracting the corresponding induced failures to estimate the overall reliability

Chapter 4

Increasing the Reliability of Register Files against Aging Effects

In this chapter, we propose a new means to mitigate the aging effects in SRAM-based register files, which are particularly vulnerable to aging due to being under a continuous stress for prolonged intervals. Based on the presented analysis, we show that aging stress in different registers needs to be tackled using different strategies corresponding to their access patterns. To this end, we propose to selectively increase the resiliency of individual registers against aging effects.

Our technique Register-Internal Stress Balancing (RISB) [5] balances the aging stress of the transistors within the cross-coupled inverters of an SRAM cell such that all transistors suffer from stress for approximately the same amount of time during operation (i.e. having similar aging-induced degradations) – thereby minimizing the deleterious effects of aging, as earlier explained in Section 2.2.1. In other words, our RISB technique balances the aging stress of SRAM cells within the register file towards making their *duty cycle* (λ) value as close as possible to 0.5 which mitigates impact that aging has on SRAM cells (see Figure 2.5). We present implementations in both hardware and in software of RISB along with the incurred overhead. Through a wide range of applications, we show that our technique increases the register file reliability with respect to aging effects by 26% on average. This is 19% better than current state-of-the-art as it will be demonstrated later in Chapter 6.

4.1 Exploration of Aging Effects in Register Files

In this section, we present an aging stress analysis of different registers of the register file and also of different bits inside the one register, to closely understand the aging problem in the register file. Then, we employ this study to raise key observations which form the foundation of our RISB technique.

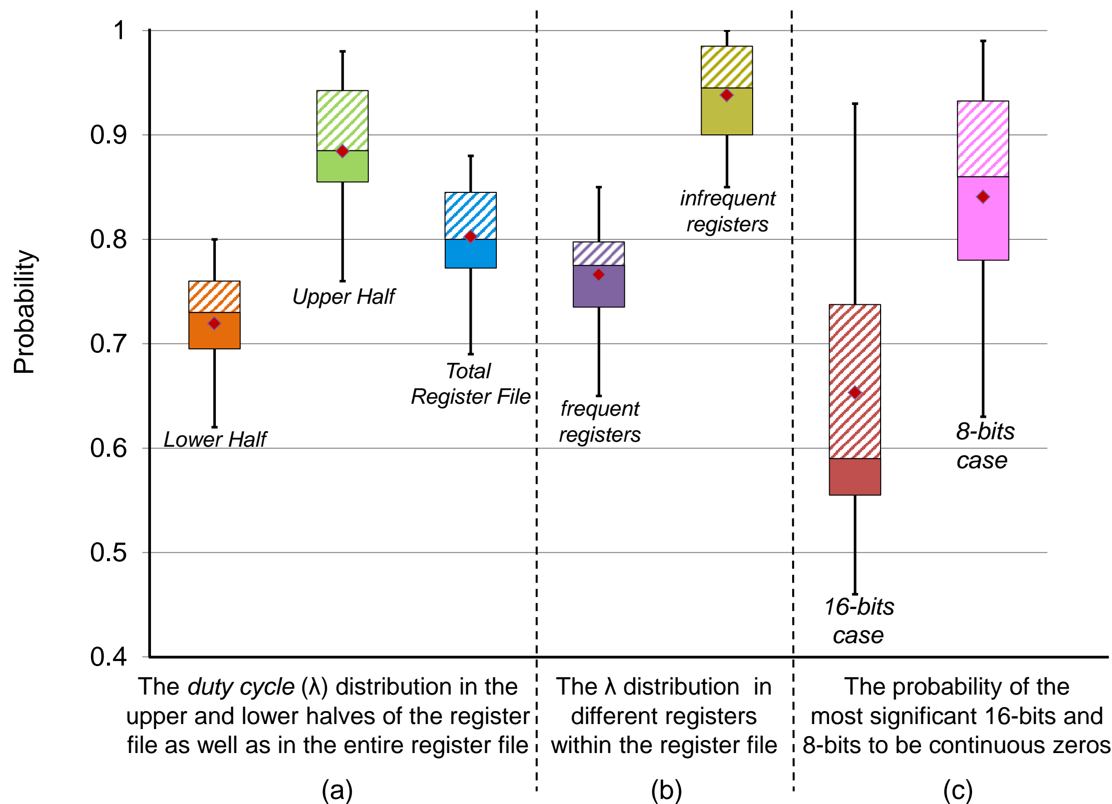


FIGURE 4.1: Register file statistics results gathered throughout all 16 benchmarks show the aging stress analysis (a and b) and the probability of the most significant *16-bits* and *8-bits* of a written value to be contiguous zeros (c)

Key Experimental Observations: In order to analyze the induced aging stress within a register file of a typical RISC architecture [37], we examine the 32-bit MIPS architecture. However, our technique can be adapted to other architectures as it will be discussed in Section 4.3.3. To take various possible stress scenarios on the register file into account, a wide range of applications from the MiBench and MediaBench benchmark suites [26, 27], compiled for the MIPS architecture, have been investigated (for further details see Section 6.3). All applications have been individually analyzed, i.e. without interruptions from other tasks or an operating system. This was necessary in order to be able to achieve an unfalsified analysis for each application.

Other works such [86] and [87] only focus on the relation between the mean stress in SRAM-based registers and aging degradation. However, when examining registers, the mean *duty cycle* λ alone is no longer able to quantify the aging-induced stress. The distribution of λ among the register bits must also be considered, since disregarding it can result in a wrong estimation in some cases. For instance, if an 8-bit register contains the constant value "11110000", then the average *duty cycle* for the total 8 bits will be 0.5, giving the impression that this register is under well-balanced stress, whereas instead this particular register will quickly age and may even fail after some time due to aging because half of its transistors (i.e. the PMOS transistor of one of the two cross-coupled inverters along with the NMOS transistor from the second inverter within the

SRAM cell) were under continuous stress for the register's entire lifetime, as it has been explained in Section 2.2.1.

For this we augment the *duty cycle* with the supporting metric, the Activity Bits Probability (ABP) of register r is defined as: $ABP_r = \{\langle\lambda\rangle_r, \sigma_{\lambda,r}\}$ where $\langle\lambda\rangle_r$ is the mean *duty cycle*, and σ_λ its standard deviation inside the register r. Extending this concept to the entire register file ABP_F, we define

$$ABP_F = \left\{ \langle ABP_r \rangle_F, \sqrt{\frac{\sum_r(\sigma_r^2 + \langle\lambda\rangle_r^2)}{N} - \langle ABP_r \rangle_F^2} \right\}$$

where N is the size of the register file, i.e. the total number of registers.

Figure 4.1(a) shows the distribution of aging stress within the register file SRAM cells for different (in this case 16) applications. As it can be observed, the *duty cycle* of register file SRAM cells, in all the analyzed applications, is within a critical level ($\lambda \in [0.69, 0.88]$, mean $\lambda = 0.8$) which indicates high aging effects on the register file, because the aging-induced degradation within an SRAM cell becomes higher when the aging stress is not *evenly* distributed among the SRAM cell transistors (i.e. when the aging stress is far from the well-balance case which occurs at $\lambda = 0.5$), as discussed in Section 2.2.1.

To obtain a closer view of the register file, we study the aging stress inside the bits of each register to recognize which bits are more susceptible to aging. Figure 4.1(a) also shows that the unbalanced aging stress is pronounced within the *upper half* of a register – defined as the half of the register where the most significant bits are stored – more than the *lower half*. As it can be noticed, the *upper half* is under higher aging stress ($\lambda \in [0.75, 0.98]$, mean $\lambda = 0.88$) and therefore the aging effects between the two halves are not balanced. In other words, the *upper half* of a register will age much faster the *lower half* which, as a result, degrades the reliability of the entire register file.

Towards further investigation, we analyze the aging stress in the individual registers to study their sensitivity with respect to aging effects. Our experiments in this direction raised a key observation: that registers which contain the same value for prolonged intervals during execution time are largely responsible for worsening the overall aging stress in the entire register file. The main reason behind this fact is that not updating the stored value in the SRAM cells of a register over long time intervals results in unbalanced stress the transistors. In other words, some transistors will be in the *stress phase* for a very long time and only in the *recovery phase* for a short time.

As a matter of fact, this observation leads to a categorization of the registers within a register file into two main groups: *infrequent* and *frequent*. The *infrequent* category contains the registers that are *infrequently* written during execution time[1] and the other group contains the registers which are *frequently* written and hence the stored value there is repeatedly changed. The aging stress distribution in both the *infrequent* and *frequent* categories is presented in Figure 4.1(b). As shown, the SRAM cells within the

[1]We consider a register to be *infrequent*, throughout our analyzed applications, if it is rarely accessed by read/write operations. Particularly, when the percentage of write/read operations occurring in that register is less than 0.1% of the total write/read operations in the whole register file.

infrequent category are under a higher aging stress than SRAM cells within the *frequent* category due to seldom updating of the stored values, leading to higher degradations due to aging effects.

Last but not least, it is noteworthy that the *zero register* presents a special case in our analysis. In hardware, this register cannot be written to and therefore it is often implemented using “hard-wired” SRAM cells through *tied-to-gnd* or *tied-to-Vdd* standard cells. Because of that, we do not apply any aging mitigation technique to the *zero register* and we consider that it is not affected by aging effects across all experiments.

In summary, we observe through our analysis that not all registers are equally affected by aging effects (i.e. some registers suffer from more aging stress than others) and the aging stress across the register bits itself is not equally balanced. Moreover, the overall aging analysis reflects the high and unbalanced aging stress on the register file such that mitigating aging-induced stress effects is a desirable goal.

4.2 Our Proposed Technique RISB: Register-Internal Stress Balancing

Based on the key observations obtained from our experiments (see Section 4.1), we propose to separately address the aging effects in *frequent* and *infrequent* registers. Firstly, we explain how our technique balances the aging stress in an entire *frequent* register along with showing the required implementation to achieve that. Later, we discuss possible implementations to also balance the aging stress within the *infrequent* registers. Figure 4.2 shows the flow diagram of our technique detailed in the rest of this section.

Frequent Registers: as it can be observed from Figure 4.1(a), the *upper half* of a register often suffers more from aging stress than the *lower half*. We also found in our study that this observation becomes more highlighted in the *frequent* registers. For that, we propose to relax these bits by finding a way to reduce the imbalance of the *duty cycle* of the individual bits of the register, and thereby minimize the aging effects.

To understand why the aging stress in the *upper half* is higher and in order to analyze which bits therein should be relaxed, we investigate the probability for the case that the most significant bits in the *upper half* are contiguous zeros/ones. Based on our statistics, we found that the probability of the *upper half* to be continues zeros plays the most important role. The presented results in Figure 4.1(c) illustrate that, when the 32-bit MIPS architecture is studied, the most significant 8 bits are contiguous zeros in 84% of the values written into the register file. Similarly, the most significant 16 bits are also contiguous zeros in 66% of all written values, on average across all benchmarks.

We propose a two step technique for balancing the aging strss within the SRAM cells of *frequent* registers of a register file. It is applied every time the register is written to.

- **Relaxing:** resulting from the observation that the most significant bits often contain the same value with little entropy, e.g. contiguous leading zeros, it is possible to reduce the number of information bits and instead write values into the *upper* bits of the register that result in balancing the aging stress on the SRAM transistors. Since the leading bits are generally zeros, this typically implies writing '1' into the upper bits – in order to compensate the induced degradations by aging during the storage of '0' – and storing the information that the bits have been replaced in binary flags (as later clarified in the implementation section).
- **Swapping:** while *relaxing* allows for balancing of aging-induced stress in the *upper* bits of a register, it is not suited to balance the stress *spatially* among all bits of the register. To facilitate this, we periodically swap the values stored in the *upper* and *lower* halves to balance the stress between them and to balance the aging effects distribution among all the entire bits of a *frequent* register.

Infrequent Registers: in order to achieve a better balancing and to further mitigating the aging effects, we propose to periodically toggle the *infrequent* registers, as they are largely responsible for the deviation of the aging stress of the register file SRAM cells from the well-balanced case where the impact of aging is minimal. Profiling information of an application can guide us to identify *infrequent* registers. Then, toggling them can be done either in *software* or *hardware*, as will be explained.

4.3 Implementation

In this section, we separately discuss the different implementations of our technique to tackle the aging degradation in the *frequent* and *infrequent* registers. It is worthy to note that aging stress in the *frequent* registers is always tackled in the hardware. Whereas in the *infrequent* registers, it can be mitigated in either **hard-** or **software**.

4.3.1 Frequent Registers

Figure 4.3 explains how our technique works in detail during read and write operations. Three extra flag bits (*S-bit*, *M-bit* and *R-bit*) are associated with each *frequent register*. The first flag, the *S-bit*, is used to distinguish between the *swapping* case and *non-swapping* case. The second flag bit (*M-bit*), is needed to indicate if the written value is being *manipulated* through applying our technique or not. Where it is considered *manipulated* when the most significant 16 or 8 bits of the written value are contiguous zeros. Finally, and only in the case of a *manipulated* value, the third flag bit (*R-bit*), is used to tell which subpart in the *upper half* has being *relaxed* (i.e. inverted to ones).

When *R-bit* is '1', it indicates to that the full *upper half* has entirely been relaxed. Otherwise, when it is '0', it indicates that only the most significant 8 bits have been relaxed. These flag bits are later on used to correctly read the stored value of that register whenever an instruction performs a read operation. Initially, we clear all flag

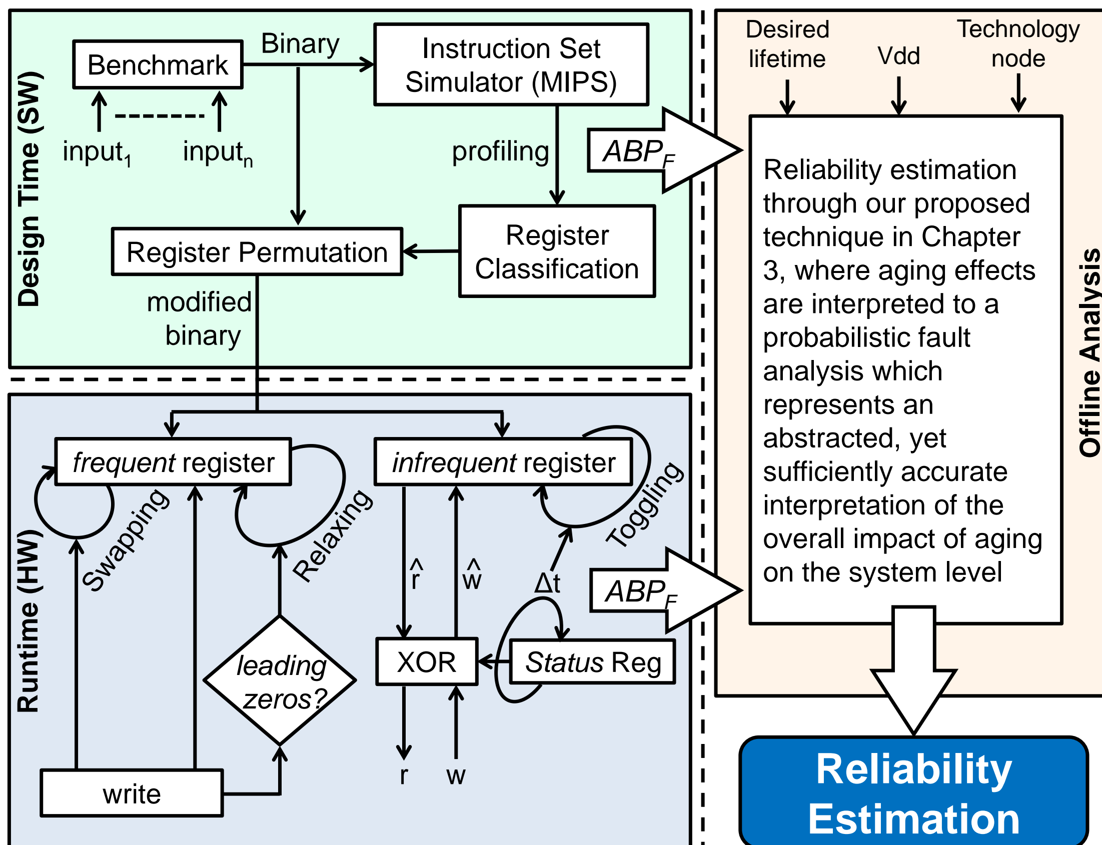


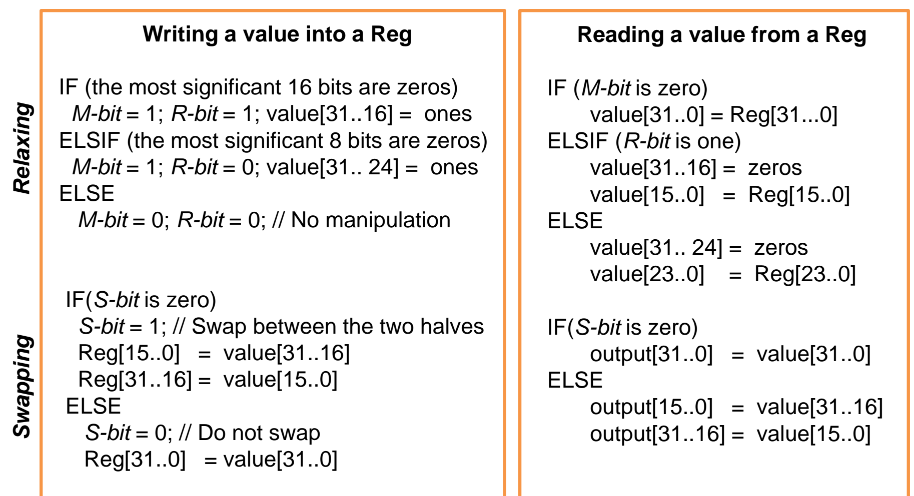
Figure 4.2: Flow diagram of our proposed technique. To estimate the register file reliability before and after applying an aging mitigation technique, we employ our presented work in Chapter 3

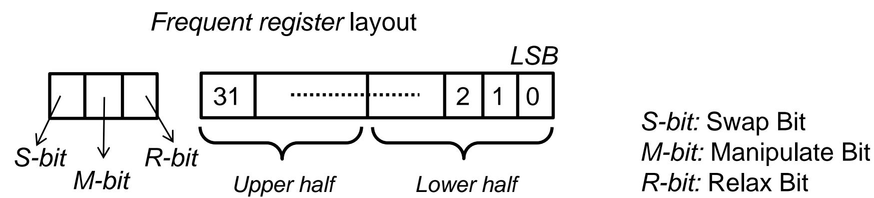
bits to indicate the absence of any *relaxing* or *swapping*. The *swapping* between the two halves is done only in the case where the *S-bit* is cleared (i.e. no *swapping* was done in the last write operation for that particular register). We then swap between the two halves and set the *S-bit* to '1' indicating the *swapping* case.

4.3.2 Infrequent Registers

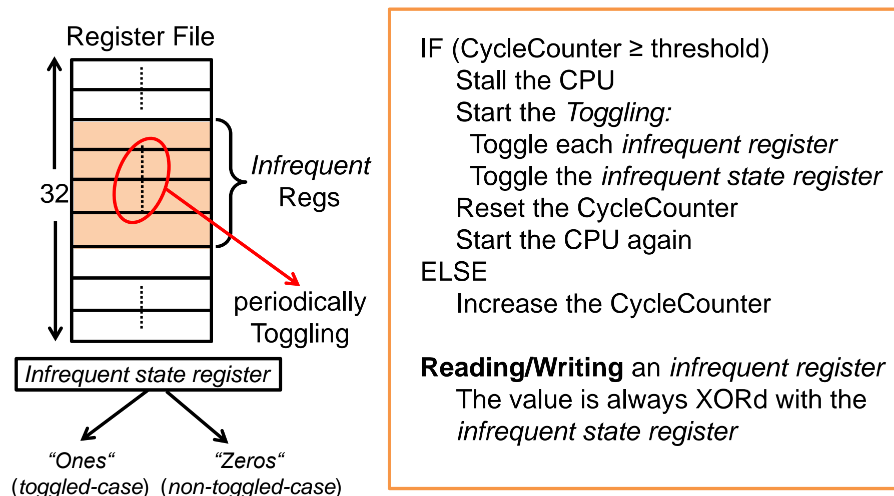
In the following we differentiate between **hardware** and **software** implementations of our technique for *infrequent* registers category and describe separately each one.

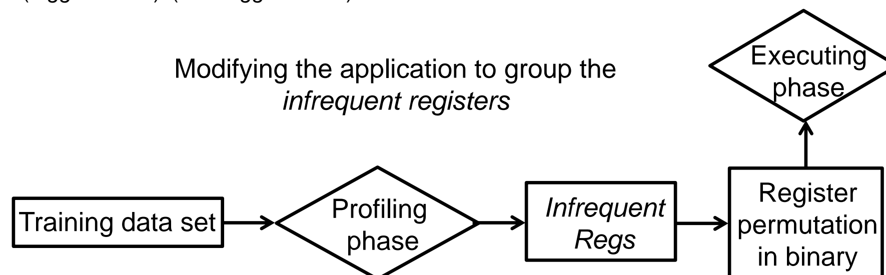
Through **software**, *infrequent register* toggling can be accomplished by inserting extra `xor` assembly instructions that periodically toggle the *infrequent* registers. Whether or not the register is in a toggled state needs to be determined when reading and writing from/to the register by adding extra assembly instructions that first decode the value of the register before it is used. A simple case for this is the `move` instruction which loads or stores the value of the register. Since there is no direct hardware implementation of the `move` instruction, the GNU assembler replaces it with `add`. Thus, for instance `move $13,5` which loads the value 5 into register 13 is replaced with `add $13,$0,5` which has the same effect. Here, the toggling can be implemented in software, by





(a) Mechanism balancing the aging stress in the *frequent* registers of a register file





(b) Mechanism balancing the aging stress in the *infrequent* registers of a register file

FIGURE 4.3: Mechanism of selectively balancing the aging stress in the *frequent* registers and *infrequent* registers of a register file according to our proposed technique

replacing the `move` instruction with `xor $13,$x,5` if the register value needs to be toggled where `$x` is a register containing all ones. In the case of the `move` command, the toggling can thus be implemented with no additional overhead in instruction count. Other instructions, however, need to be replaced by multiple instructions that first perform the `xor` separately.

In general, there are two scenarios in the software implementation: either there is a register which is unused during application execution or not. If such a register exists, it can be exploited to store the 32-bit *infrequent state register*. Assuming this register is `$x`, a default implementation for a general instruction `instr $i` for the *infrequent* register `$i` using in-register decoding is as follows:

```
xor   $i, $x, $i
instr $i
xor   $i, $x, $i
```

Thus adding two extra instructions in the general case (unlike the special case of, e.g. the `move` instruction which takes no additional cycles). Since *infrequent register* accesses comprise less than 0.1% of the total read/write accesses (typically around 100 accesses per application), this overhead is negligible. The greater overhead arises when the *infrequent* registers are toggled, as this occurs more often and involves the *infrequent state register* as well as all n *infrequent* registers `$i1 ... $in`:

```
nor $x,  $x,  $zero
nor $i1, $i1, $zero
...
nor $in, $in, $zero
```

This toggling is performed every interval Δt which is targeted to be around each 100k cycles, adding an overhead of $a_b \cdot (n+1)$ cycles at each iteration, with a_b being the overhead for each bitwise logic operation, e.g. through the `nor` and `xor` instructions. In our simulator, a_b is equal to one cycle. The overall performance overhead for an application with total execution time E is thus:

$$\frac{E}{\Delta t} \cdot a_b \cdot (n+1) + 2 \cdot a_b \cdot \sum_{k=1}^{n} A_k \quad cycles \tag{4.1}$$

where A_k is the number of accesses to register `$ik`, $k \in [1, n]$.

If there is no extra register available to store the *infrequent state register* however, there are two possibilities. The *first possibility* is to use immediate values[2], transforming the instructions to:

```
xori  $i, $i, 0xFFFF
rol   $i, $i, 16
```

[2]MIPS immediate instructions operate on 16 bits.

```
xori  $i, $i, 0xFFFF
rol   $i, $i, 16
instr $i
xori  $i, $i, 0xFFFF
rol   $i, $i, 16
xori  $i, $i, 0xFFFF
rol   $i, $i, 16
```

if `$i` is toggled, and simply `inst $i` if not. In this case the compiler must keep track of whether or not the register value has been toggled. The worst performance overhead is $8 \cdot a_b \cdot \sum_{k=1}^{n} A_k$. This method, however, has the major shortcoming that it cannot be used inside a loop since the compiler needs to know the toggle state, and it is therefore not usable for many applications. To allow this, it is necessary to store the *infrequent state* in memory and use one of the *non-infrequent registers* as a temporary register `$t`:

```
sw    $t, [temp_addr]
lw    $t, [infreq_state_addr]
xor   $i, $t, $i
instr $i
xor   $i, $t, $i
lw    $t, [temp_addr]
```

The aforementioned methods, which represents the *second possibility*, adds $3 \cdot (a_l \cdot 2) + 2 \cdot a_b$ cycles where a_l is the overhead of a load or store instruction. `sw` and `lw` both require two store or load operations, respectively, since MIPS only allows storing/loading 16 bits at a time. In our simulator, this overhead was 14 cycles.

Toggling the n *infrequent* registers can be performed as follows:

```
sw  $t,   [temp_addr]
lw  $t,   [infreq_state_addr]
nor $t,   $t,  $zero
nor $i1, $i1, $zero
...
nor $in, $in, $zero
sw  $t,   [infreq_state_addr]
lw  $t,   [temp_addr]
```

which adds $4 \cdot (a_l \cdot 2) + (n+1) \cdot a_b$ cycles ($= 17 + n$ cycles in our simulation, where $a_b = 1$ and $a_l = 2$ cycles). The overall performance overhead is:

$$\frac{E}{\Delta t} \cdot (8 \cdot a_l + (n+1) \cdot a_b) + (6 \cdot a_l + 2 \cdot a_b) \cdot \sum_{k=1}^{n} A_k \quad cycles \tag{4.2}$$

The main challenge in implementing the software approach is finding a suitable place in the code to perform toggling. At present, this requires manually profiling each application and inserting the corresponding toggle instructions into the assembly code. In practice, we were able to find a free register to use as the *infrequent state register* in

almost all benchmarks. This is due to the fact that the GNU assembler often did not make use of the *assembler temporary* register `$1`.

The advantage of this approach is that it does not modify the hardware and therefore has no area overhead and can be used on existing hardware. An additional benefit of the software implementation is that it can be applied for multiple applications when multitasking is used since it does not physically change the register file. In such a case, each application must store its own separate register state in memory. On the other hand, the software approach comes at the cost of performance degradation due to the extra cycles that are required for the additional inserted assembly instructions and possible memory accesses (e.g., to store the register state). Since only *infrequent* registers are affected. However, this overhead remains low due to their small access count, our average measured overhead being a 3.7% increase in execution cycles.

The **hardware** implementation can be done by gathering the *infrequent* registers in a specific part of the register file. Then, periodically toggling the registers in that part over the runtime (see Figure 4.3). A counter is needed that counts the number of cycles until it reaches the predetermined threshold at which point toggling is performed. Additionally, the CPU needs to be stalled during toggling to ensure that the value of the register remains consistent before and after toggling as well as to avoid any potential conflicts from register accesses while toggling. During a read/write operation to the register, the value is always XORd with a dedicated *infrequent state register* which contains either all zeros (non-toggled) or all ones (register value toggled) depending on the toggle state of the register.

The hardware implementation requires the location of the *infrequent* registers to be known a priori. To this end, we use a post-compilation strategy that modifies the binary of an application. Practically, we permutate the registers to guarantee that the *infrequent* registers are always located in the dedicated part in the register file. For instance, exchanging the registers x and y means replacing all occurrence of register x with y and similarly replacing those of y with x.

Interval (cycle)	*Measured Effect*	*adpcm encoder*	*adpcm decoder*	*qsort*
100	*Exec. cycle overhead*	80%	84%	59%
	Total $\lambda \in 0.5 \pm 0.05$	68%	71%	53%
100K	*Exec. cycle overhead*	0.016%	0.017%	0.012%
	Total $\lambda \in 0.5 \pm 0.05$	58%	58%	43%
5M	*Exec. cycle overhead*	$< 10^{-4}$%	$< 10^{-4}$%	10^{-4}%
	Total $\lambda \in 0.5 \pm 0.05$	55%	23%	18%

Table 4.1: The potential effects of the chosen interval to toggle the *infrequent* registers on the execution cycles and percentage of aging-balanced register file SRAM cells

The threshold of toggling the *infrequent* registers is chosen as 100,000 cycles in our implementation. Based on our experiments this interval presents a trade-off between the execution overhead and achieved balancing in terms of aging stress. Table 4.1 demonstrates the potential impact of aforementioned trade-off. As it can be noticed, larger intervals will result in less aging-balanced SRAM cells per application (i.e. the percentage of SRAM *duty cycle* that is close to 0.5 is less) but they have smaller overhead in

terms of execution cycles. Smaller intervals, on the other hand, would be accompanied by a noticeable performance loss but with more aging-balanced SRAM cells.

4.3.3 Other Microrchitectures

Although not our focus in this work, our proposed technique can also be applied in other microprocessor architectures such as the out-of-order processors [38]. There, the renaming unit can assign *infrequent* registers to predetermined physical registers where the aging-induced stress is relaxed by our introduced *hardware* solution. Indeed, for longer bit-width registers (e.g., 64-bit), our analysis to determine the high-stressed bits in a register should be repeated similarly (see Figure 4.1). Actually, dividing the registers into *infrequent* and *frequent* categories, as we propose, exists to some extent, but lessens the more physical registers there are. The divide disappears once there are so many registers that all of them are in effect only infrequently written, meaning all physical registers are not used most of the time.

Chapter 5

Increasing the Reliability of Register Files against Soft Errors

Over the last decade, and in spite of the increasingly advanced architectures, the technology scaling has raised the threats akin to soft errors to become one of the key external sources for jeopardizing the reliability of on-chip systems in the nano-CMOS era, as motivated in Chapter 1. Soft errors caused by charged particles are dangerous primarily in high atmospheric, where heavy energetic particles are available [136]. However, trends in today's nano-scale technologies such as aggressive shrinking in transistors feature sizes along with lower operating voltages have made low-energy particles, which are more superabundant than high-energy particles, also cause appropriate charge to provoke a soft error in the terrestrial applications. Therefore, there is a prevailing prediction that soft errors will become a cause of an inadmissible error rate problem in the near future even in earthbound applications [137]. Despite the fact that the overall area footprint of the register file is rather small, the register file is accessed more frequently than any other microarchitectural component [97, 138], as earlier discussed in Chapter 1. Thus, corrupted data in any register, if not taken care of, may rapidly propagate throughout the other components of processor, leading to drastic reliability problems [138].

In other words, the rise in soft errors in critical components of shrinking architectures has led to a prominent need to cope with their deleterious effects in the register files. Additionally, SRAM cells of register files can also be simultaneously threaten by aging effects as it has been shown in Sections (1.3, 3 and 4). Therefore, there is an aggravated need for new techniques to increase the register file resiliency against soft errors and simultaneously mitigate the aging effects. On the other hand, due to the fact that the register file achieves an elevated average temperature [45, 46], analyzing the potential impact of any register file protection technique on the consumed power is essential to maintain a minimum influence on the temperature increase.

In this chapter, we address the aforementioned challenge in order to develop a lightweight technique called Register-Embedded Self-Immunity (RESI) [1, 7] to reduce the vulnerability of register files not only against SBUs but also against MBUs by 97% (up

to 100%) resulting in a high system fault coverage under various scenarios. The technique is based on our observation that some register bits are not continuously used to represent a value stored in a register. Thus, we propose to exploit the unused bits in order to improve the register file immunity against soft errors. We demonstrate how our proposed technique can additionally result in mitigating the aging stress in register file SRAM cells. We also tackle the problem of the cost overhead that typically comes as a negative side effect when the register file is protected against MBUs. Compared to protecting the register file against SBUs only, our technique operates at lower temperature, consumes less power and still occupies a similar area footprint. The achieved result is 63% better compared to state-of-the-art in register file protection. Finally, to compare and quantify the effect of our technique, we observe its impact on the processor's temperature using an IR thermal camera and show that, due to consuming less power per area, our technique also operates at a lower temperature compared to protecting the register file against SBUs only, as it will be later discussed along with other experimental results in Chapter 6.2.

5.1 Our Proposed Register-Embedded Self-Immunity Technique: RESI

In order to effectively utilize chip area, we propose the idea of exploiting register values that do not require all bits for value representation, i.e. values that are generally coded using only the lower bits of the register. The upper *unused* bits of that register can be exploited to increase the register's immunity against soft errors by embedding the required ECC, minimizing the need for extra bits. In the following, we first apply this idea to develop a solution to protect the register values against SBUs then we additionally address MBUs. Finally, we present the required microarchitectures for both solutions along with a discussion regarding the effectiveness of our RESI technique.

5.1.1 Single Bit Upsets

The required number of parity bits when the Hamming SEC code is used to protect an w-bit data word against SBUs is $p = \log_2 w + 1$. Thus, the entire code word will be $w + \log_2 w + 1$ bits and any single error occurring either in the original data word or in the SEC code itself can be recovered [93]. Assuming that l is the number of the bits required to represent a register value, then according to our proposed idea, the bit-width n of the register file should cover both l and the corresponding SEC code of that value. Therefore, the desired value of l is the maximum value which guarantees this. In other words, l is the maximum value that holds the following condition ($l+\log_2 l+1 \leq n$). For instance, when studying 32-bit architectures, where each register can represent a 32-bit value, the desired value of l will be 26[1]. Thus, we could exploit register values that only

[1]Our technique can easily be adapted to other architectures. For the 64-bit architectures, the desired value of l will be 57.

require the lower 26 bits of a register to be represented by storing the corresponding SEC code in the upper *unused* six bits.

We call such values *manipulatable* values. On the other hand, we call register values which cannot embed their SEC codes (as they need over l bits to be represented) *non-manipulatable* values. Figure 5.1(a) shows the percentage of register value usage gathered from 10 different applications from the MiBench Benchmark Suite [27] compiled for the MIPS architecture. As it can be noticed, most of the register values are *manipulatable*. In other words, on average the upper six bits of 88% of the stored data in the register file are '0's. Thus, we may exploit these *unused* bits to store the required SEC code and thus increase the register's resiliency against SBUs without the need for additional bits.

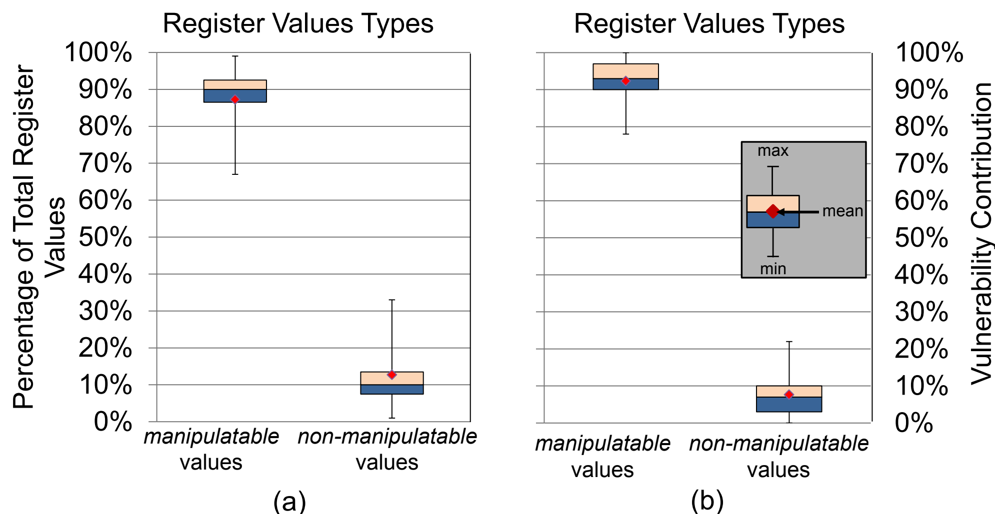
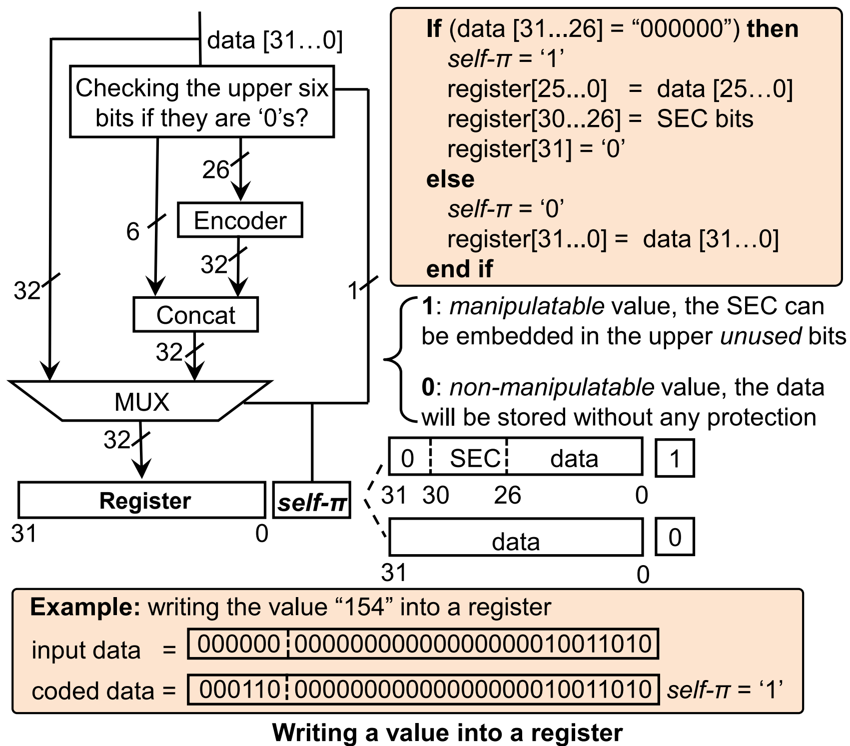


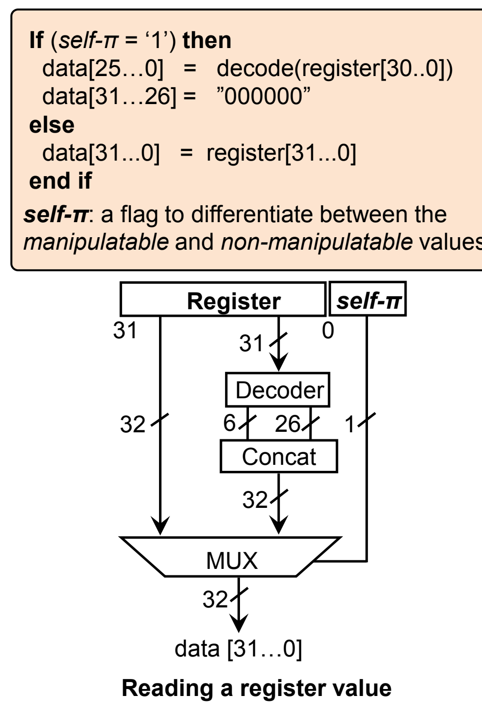
FIGURE 5.1: Register file statistics gathered throughout various 10 applications show the distribution of the *manipulatable*/*non-manipulatable* register values and the corresponding fraction of vulnerable intervals

Additionally, the contribution of *manipulatable* register values to the total vulnerable intervals is much more than the contribution of *non-manipulatable* register values. It reaches, on average, 93% as reported in Figure 5.1(b) which presents the fraction of register values in vulnerable intervals for the studied benchmarks. Therefore, embedding the SEC codes in the *manipulatable* values can lead to a considerable reduction of the overall register file vulnerability – as it will be later evaluated in Section 6.2 – since protecting the register values during the corresponding vulnerable intervals results in converting a large fraction of vulnerable intervals into invulnerable intervals because soft errors during these particular intervals will be harmless.

To distinguish the *manipulatable* register values from the *non-manipulatable* register values, a 1-bit flag (called self-π bit) needs to be associated with each register. Initially, all flag bits are cleared to indicate the absence of any protection. Whenever an instruction writes a value to a register, it checks whether the upper six bits of that value are '0's or not. If they are (*manipulatable* register value case), the corresponding self-π bit is set to '1' indicating the existence of register protection. The SEC code is generated and stored in the upper bits of the register. Hence, the data value and its SEC code are



(a) Microarchitectural support for writing a value into a register



(b) Microarchitectural support for reading a register value

FIGURE 5.2: Microarchitectural support for writing and reading a register value when our RESI protection technique against SBUs is applied

stored together in that register. In the second case (*non-manipulatable* register value), the self-π bit is set to '0' and the value is written into the register without encoding.

In 'read' operations, the self-π bit is used to differentiate between the *manipulatable* and *non-manipulatable* values. In the first case, the value and its SEC code are stored together in that register and consequently the read value needs to be decoded. In the second case, the stored value is not protected and hence does not need to be decoded. The required microarchitectures to support the 'read'/'write' operations are illustrated in Figure 5.2.

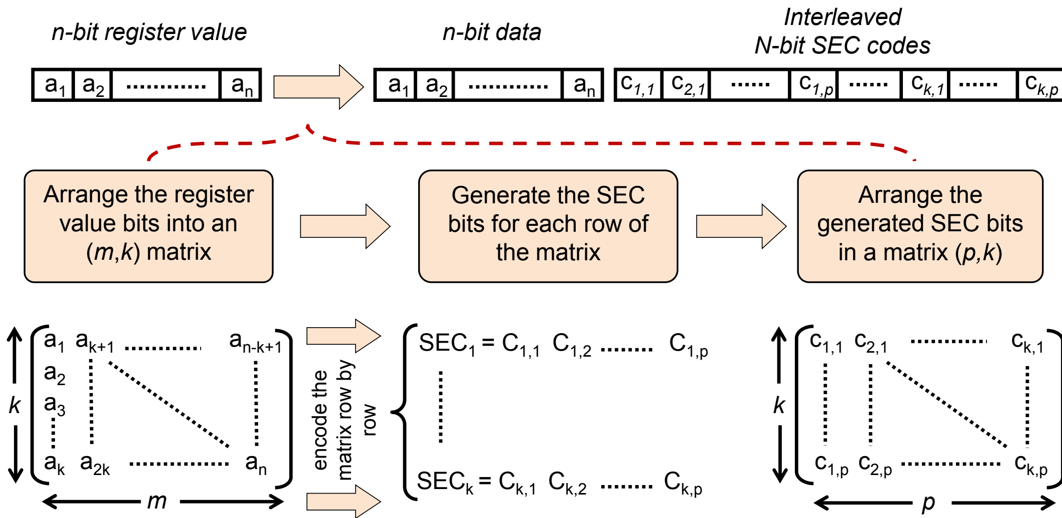
5.1.2 Multiple Bit Upsets

In this section, we extend our idea to tackle the vulnerability against MBUs in adjacent bits of a register as it is higher than in random bits. Therefore, we divide (based on the interleaving concept [139]) the register value into small words and protect each of them against SBUs, instead of generating the SEC code for the entire register value as one word (as is the case in the conventional protection schemes). This will increase the resiliency of the register against MBUs because an incident particle will only generate an error when it causes more than one upset in the same divided word. Let us assume that n is the bit-width of a register and k is the desired number of bits protected against upsets, we propose to divide the register value into k words where each word consists of $m = n/k$ bits. Hence, the generated SEC code of each divided word will consist of p-bits where $p = \log_2 m + 1$. Consequently, the total number of extra bits required to store the generated SEC codes will be $N = p * k$. Interleaving makes the adjacent bits of a register value always belong to different SEC codes as clarified in Figure 5.3.

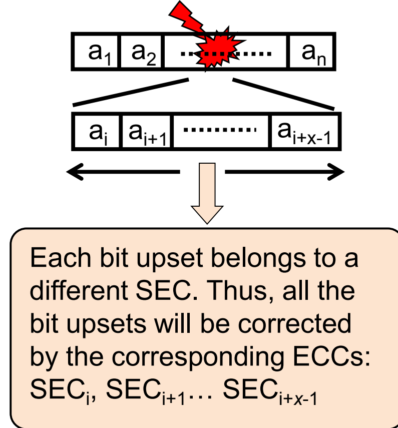
Practically, we arrange the n-bit register value into a matrix $M(m, k)$ (see Figure 5.3(a)) and encode this matrix row by row using a Hamming SEC code. Since an MBU may also affect multiple adjacent bits in the stored SEC codes which would result in losing the ability of recovering the corrupted register values, we also arrange the bits of the individual generated SEC codes to alternate between each other to guarantee that adjacent bits always belong to different SEC words. The example in Figure 5.3(b) shows that when a particle generates upsets in x adjacent bits (either in the data bits or in the bits of the SEC codes), where $x \leq k$, the corresponding SEC codes will detect and correct all of these upsets. However, the main drawback of the aforementioned method is that the area overhead (imposed by storing the generated SEC bits and for the needed ECC encoders/decoders) quickly increases. Similarly, the total power overhead (consumed when accessing the ECC encoders/decoders) may also increase.

5.1.2.1 Reducing Overheads

To solve the problem of the high potential overheads, we also exploit the upper bits of a register value to store a part of the generated SEC words similarly to how we protected the register file against SBUs in the previous section. However, instead of employing the *unused bits*, we define the Available Bits (AvB) of a register value as the number of upper bits of that register value that are not actually used to represent the register value and thus they are *available* for storing bits of the SEC codes. To increase the amount of protected register values, we also include all leading bits with minimal information



(a) The required mechanism of our RESI technique in the case of MBUs



(b) Example of how stored SEC codes, when our RESI is applied, detect and correct the occurred bit upsets

FIGURE 5.3: Our proposed RESI technique in the case of MBUs for encoding a written value and how the register bits need to be arranged

entropy in the AvB, i.e. values with leading '1's in addition to values with leading '0's, whereas the previous term definition (*unused bits*) covered only the leading '0's case. To investigate how often the generated register values require the full bit-width of a register, Figure 5.4 reports the AvB distribution for different applications. As seen, on average 97% and up to 100% of the generated register values have 8 bits *available* ($AvB = 8$) and the percentage of the register values that have less than 8 is almost negligible. In other words, the written register value seldom requires more than 24 bits to be represented. Based on this property, we mitigate the high overheads that come from the presented mechanism in Figure 5.3(a).

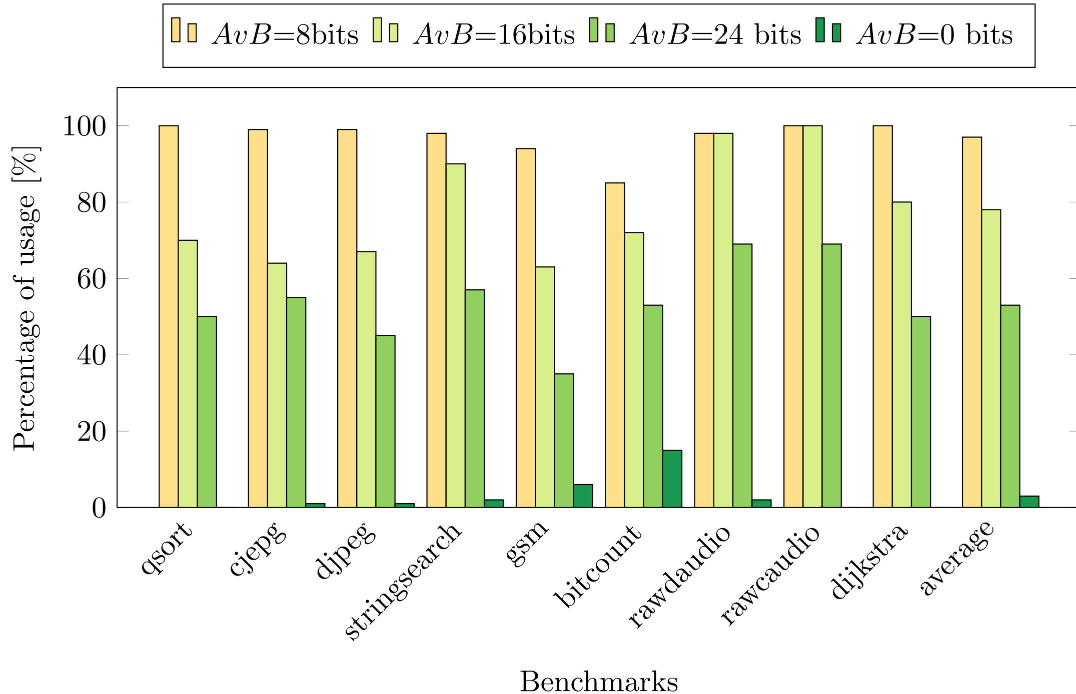


FIGURE 5.4: Distribution of the proposed Available Bits (AvB) of register values for different benchmarks

5.1.2.2 Microarchitectural Support

To achieve a trade-off between the potential overheads and the number of bit upsets to be corrected, we choose $k = 3$. Experiments in [17] reported that 75% of soft errors cannot generate more than 3 simultaneous upsets. Based on our observation raised from Figure 5.4, we propose to exploit the register values where $AvB = 8$ to store a part of the SEC codes generated from encoding the register value matrix as clarified in Figure 5.3(a). The key problem then is that the processor does not have sufficient information to make the two following decisions when a value is read from a register:
1) Is a part of the SEC codes being embedded in the AvB of that register value or not?
2) If it is, were these 8 *available* bits '0's or '1's?
The first point has been previously addressed by the proposed self-π bit associated with each register to make the required decision and we similarly tackle the second question. In other words, two flag bits (self-π, f) are associated with each register and we initially clear them to indicate the absence of any protection. The first flag bit is used to distinguish whether a part of the SEC codes is being embedded in the register value or not and the second flag bit is used to distinguish whether the upper 8 bits (AvB) were '0's or '1's. For the sake of clarity, we explain the proposed architecture in two steps:

1) Writing a value into a register: Whenever an instruction writes a value into a register, the upper 8 bits are checked to decide if they are *available* or not (i.e. leading '0's or '1's). If they are *available*, then the register value will be protected as demonstrated in Figure 5.3(a). Hence, the corresponding flag bit self-π is set to '1' indicating the

existence of protection and f is set to '1' or '0' if the upper 8 bits are leading '1's or '0's, respectively. The ECC encoders are then used to compute the corresponding SEC codes for each divided word. In the second case, where the upper 8 bits of the register value are not *available*, the corresponding flag bit self-π is set to '0' indicating the absence of any protection and the value is written into the register without encoding (the ECC encoders will not be activated). Figure 5.5(a) shows the proposed flow diagram to achieve the desired MBUs protection and Figure 5.5(b) shows an example of how our RESI technique protects a register value against MBUs.
2) Reading a value from a register: In 'read' operations, the self-π flag is used to distinguish between the protection and non-protection cases. In the first case, the value and a part of the SEC codes are stored within that register value and consequently the read value should be decoded. Similarly, f flag is used to determine if the upper 8 bits are '0's or '1's. In the second case, where self-π is '0', the ECC checking operations are skipped.

5.2 On Effectiveness

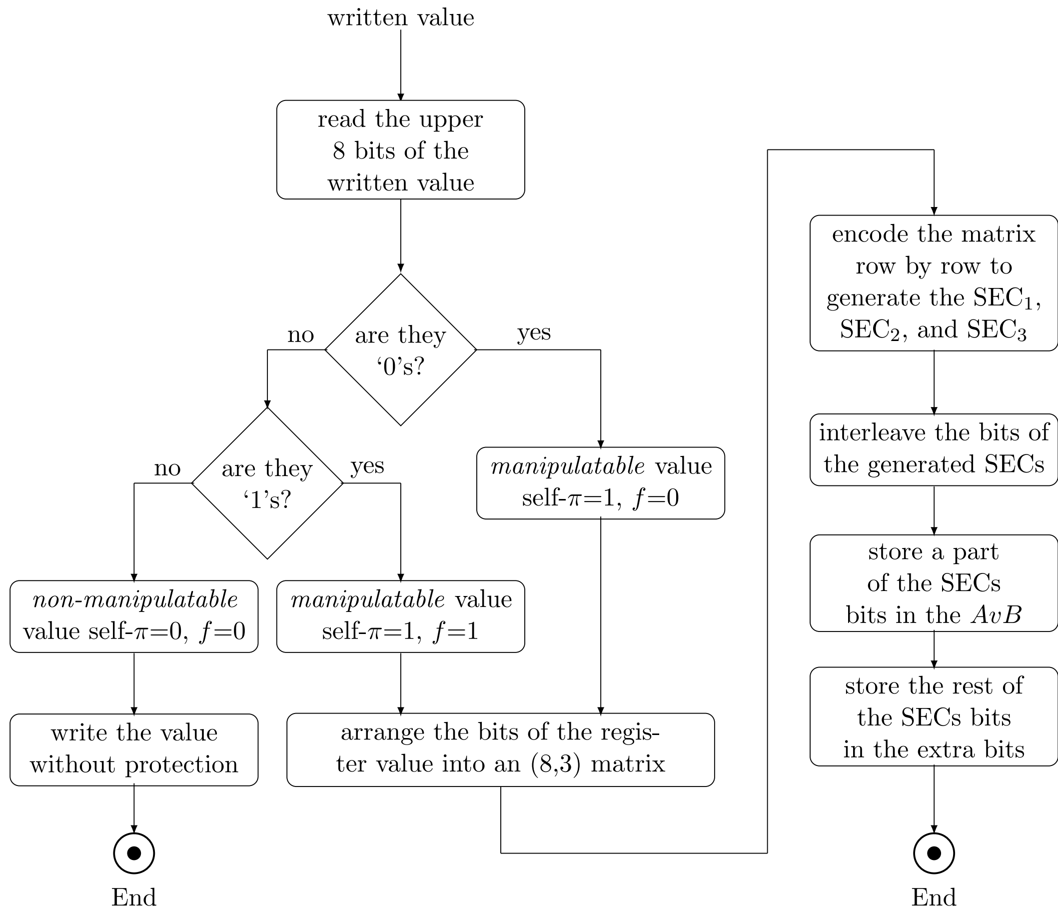
Since the main target of our work is to increase the reliability of the register file with minimum impact on the overhead, we discuss in this section the effects of our introduced technique to cope with MBUs in terms of area and power as well as how it can additionally result in mitigating the aging stress within the SRAM cells of the protected register file.

5.2.1 Impact on Area

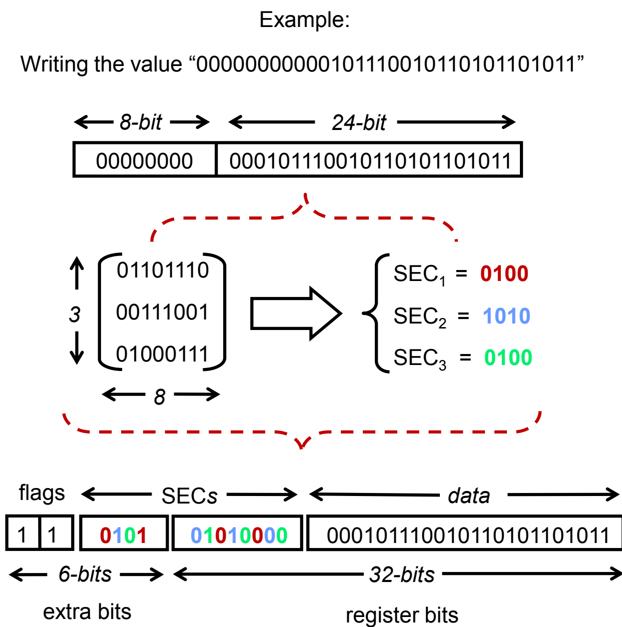
Because of RESI technique arranges the lower 24 bits of the register value into an $M(8, 3)$ matrix and encodes this matrix row by row using three Hamming SEC encoders, the total number of generated bits will be $(3 * (\log_2 8 + 1) = 12 \text{ bits})$, as demonstrated in Figure 5.3(a), 5.5). As discussed, 8 out of 12 bits will be stored within the upper 8 *available* bits of that register and the rest of bits have to be stored in extra 4 bits. Therefore, instead of the need of 12 extra bits to store all of the generated SEC codes, we now only need 4 extra bits in addition to the two flag bits (i.e. 6 extra bits in total).

5.2.2 Impact on Power Consumption

The ECC generation operations, in our architecture, are not performed with each write operation and similarly ECC checking operations are also not performed with each read access as occurs in the conventional protection ways. In fact, our RESI technique decides to protect the written register value only if it has at least 8 AvB, in which case the ECC generators are activated to compute the needed SEC codes. Otherwise, no generation of ECCs is executed. Similarly, on every register read operation, instead of always checking ECCs, we determine whether the ECCs are being embedded in the register value, and only if they are, the ECC checking is performed. For that, our RESI technique promises



(a) Hardware flow diagram illustrating the case of writing a value into a register according to our protection technique against MBUs



(b) Example of how our proposed RESI technique protects a register value against MBUs

FIGURE 5.5: Hardware flow diagram of our proposed RESI technique against MBUs together with a scenario of an application of it

to achieve a good power saving due to not performing ECC generating/checking with each write/read operation, as will be examined in Section 6.2.

5.2.3 Impact on Aging Effects

In order to investigate the impact of our technique on, additionally, mitigating the aging stress in the register file, we rely on our previous analysis of aging stress from different applications that has been presented in Section 4.1. Figure 4.1 presents the analysis of *duty cycle* of the register file SRAMs in the absence of any protection technique. As it can be observed from our analysis that the upper half of the register file suffers from higher aging stress compared to the lower half. In other words, it is most likely that the bits belonging to the upper half will age faster than the lower part which threats the reliability of the entire register file. Moreover, the mean aging stress of the total register file is also far from the well-balanced case (0.5) and falls in a critical level (0.8 on average). In fact, this observation is compatible with the results presented in Figure 5.4 which reported that the AvB of around 80%, on average, of the written values is 16, i.e. the upper 16 bits in 80% of the values written into the register file are either continuous zeros or ones. This, in turn, makes the corresponding SRAM cells, that store these bit values (the upper half of the register file practically), strongly suffer from aging stress (see Figure 2.5) because half of the SRAM transistors (i.e. a PMOS transistor from one of the cross-couple inverters along with an NMOS transistor from the second inverter, as illustrated in Figure 2.3) will be continuously under aging stress for the majority of lifetime. Since we propose to embed the required SEC codes in the upper unused bits of a register, it is expected that the aging stress in these bits will be moderated as they are no longer leading zeros or ones (i.e. continuous aging stress) over the most of execution time (further details will be presented in Section 6.2).

Chapter 6

Evaluation, Comparison and Advantages

After explaining the experimental setup that we employed to evaluate our proposed RESI and RISB techniques, we demonstrate how our RESI technique increases the register file reliability with respect to soft errors and we also clarify the achieved balancing in terms of aging-induced stress in the register file SRAM cells. Then, we study the systems fault coverage after implementing our RESI technique by applying different fault injection simulations representing varied scenarios [1]. Afterwards, we study the impact of our RISB technique on balancing the aging stress within the register file SRAM cells and then we evaluate its impact on mitigating the aging-induced failures [5]. Finally, we present our software/hardware-based techniques to evaluate the overall impact of aging effects in the register file when *conforming workloads* are running on top of an embedded on-chip system [2]. The estimations of the register file reliability for the purposes of evaluating the effectiveness of our techniques and the comparison with state-of-the-art are performed through the employment of our proposed reliability estimation in Chapter 3.

6.1 Experimental Setup

To take into account various register file utilization scenarios, we use a wide range of applications from the MiBench and MediaBench Benchmark Suites [26, 27] compiled for the MIPS architecture [140]. For a fair comparison, we consider the following versions of the MIPS processor in order to fairly evaluate our RESI and RISB techniques:

Base: The regular i.e. unmodified processor where the values are stored in the register file without any modification (i.e. neither protected nor manipulated).
FullySBU: A fault tolerant version, where the register file is *fully* protected against SBUs through the Single-Error Correcting and Double-Error Detecting (SEC-DED) code [141].
IRR: A fault tolerant version, where the "In-Register Replication" technique [99] is

implemented to protect the register file against both SBUs and MBUs (details of the technique are in Section 2.3).
***RESI-SBUs*:** A fault tolerant version, where our RESI protection technique against SBUs is implemented (details of the technique are in Section 5.1.1).
RESI-MBUs: A fault tolerant version, where our RESI protection technique against MBUs is implemented (details of the technique are in Section 5.1.2).
RISB: An aging-aware version, where the aging effects in the register file SRAM cells are mitigated through our RISB technique (details of the technique are in Chapter 4).
BR: An aging-aware version, where the aging effects in the register file SRAM cells are mitigated through the state-of-the-art “Bit Level Rotation” technique [84] (details of the technique are in Section 2.2.2).

6.2 RESI Technique Evaluation

To study the power and area overheads in ASICs, we utilize the Synopsys compiler tool together with the TSMC technology library [142] in order to analyze the different ECC encoder/decoder designs. The required setup for the ***hardware*** evaluation is done by implementing and synthesizing a VHDL model of the DLX processor for a Xilinx Virtex2 FPGA [143]. The DLX CPU has been used there because we target 32-bit embedded processors as well as it is based on the MIPS architecture. The RTL implementation of required ECC encoders/decoders are based on the open-source Hamming VHDL code in [141].

To also evaluate the impact on the *temperature*, we capture the IR images with a DIAS Pyroview 380L thermal camera [144]. It operates at rate of 50 Hz with a spatial resolution of 50 μm. In order to accurately monitor the processor’s IR emissions, it was necessary to remove the chip’s packaging to expose the silicon wafer since the packaging distributes the temperature and thus the ability of precisely reading the processor’s temperature, emitted from the backside of the test FPGA chip, is lost. Once the measurements have been obtained, the camera sends the captured thermal images to host a PC for analysis. Our experimental setup is shown in the left side of Figure 6.4 and further technical details regarding the IR thermal measurements will be later presented in Chapter 7.2.

6.2.1 Impact on Area

To demonstrate the potential impact on area after implementing our technique compared to the *FullySBU* version, we consider a (32x32) register file as an example, which is typical for MIPS architectures [37].

The extra bits: Practically, we arrange the lower 24 bits of the register value into an $M(8,3)$ matrix and encode this matrix row by row using three Hamming SEC encoders. Thus, the total number of bits of the generated SEC codes will be $(3 * (\log_2 8 + 1) = 12)$ bits (see Figures 5.3(a) and 5.5). Thereafter, 8 out of 12 ECC bits will be stored in the upper eight AvB bits of that register and the rest of bits are stored in extra bits. As a

result, instead of the need of 12 extra bits to store all of the generated SEC codes, we now only need 4 extra bits in addition to the two flag bits (i.e. 6 extra bits in total). Thus, our RESI technique comes with a similar overhead in terms of the number of extra bits as compared to protecting the register file against SBUs only (i.e. *FullySBU*)[1].

The ECC Encoders/Decoders: Our technique requires three Hamming (8-bit) encoders, instead of one Hamming (32-bit) encoder. Similarly, it needs three (12-bit) decoders instead of one (38-bit) decoder. To investigate the required area of using three (8-bit) encoders plus three (12-bit) decoders against using one (32-bit) encoder plus one (38-bit) decoder, we implemented and synthesized two different versions of encoders/decoders. For a more general comparison, we targeted both ASICs as well as FPGA platforms (a Xilinx Virtex2 [143]). As shown in Table 6.1, the total area (for the both platforms) of using three of Encoder(8-bit) is smaller than using one Encoder(32-bits). Similarly, employing three of Decoder(12-bit) also comes at a smaller area than one Decoder(38-bits).

The aforementioned comparisons both in terms of the total extra bits and required ECC encoders/decoders, show that our RESI technique occupies a similar area footprint, compared to fully protecting the register file against SBUs only (i.e. *FullySBU*), while it protects the register file against MBUs.

6.2.2 Power Consumption

Table 6.1 presents also the power results for different Encoders/Decoders for the ASIC case. As it can be noticed, using three Encoder(8-bit), consumes less power compared to one larger Encoder(32-bit). Moreover, three Decoder(12-bit) come with less total power consumption than one Decoder(38-bit). The total power consumed in the extra hardware components required for our architecture is computed by multiplying the total number of access of a particular component by the dynamic power consumption per single operation in addition to considering the corresponding leakage power.

$$Total\ Power\ Overhead = \frac{\alpha \cdot P_{detector} + \epsilon \cdot P_{checker} + \beta \cdot P_{decoder} + \vartheta \cdot P_{encoder}}{ExecutionTime} + L;$$

$$L = L_{detector} + L_{checker} + L_{decoder} + L_{encoder}$$

There, α is the total number of 'write' operations. In fact, whenever a new value is being written into a register, the *detector* will check it to decide if it is *manipulatable* value or not. Similarly, ϵ is the total number of 'read' operations and the *checker* checks the read value if it is *manipulatable* or not. β is the total number of ECC decoding operations and ϑ is the total number of ECC encoding operations. $L_{detector}$, $L_{checker}$, $L_{decoder}$ and $L_{encoder}$ are the leakage power of the *detector*, *checker*, *decoder* and *encoder* components, respectively.

[1] The number of required ECC bits is $\log_2 32+1 = 6$. In the case of a dual-error detection, it becomes $\log_2 32 + 2 = 7$.

Additionally, the ECC generation operations are not performed with each 'write' operation and similarly ECC checking operations are also not performed with each 'read' access. As a matter of fact, our RESI technique decides to protect the written register value only if it has at least eight available bits (i.e. $AvB = 8$), in such a case the ECC encoders are activated to compute the needed SEC codes. Otherwise, no generation of ECCs is executed. Similarly, on every register 'read' operation, instead of always checking ECCs, we determine whether the ECCs are being embedded in the register value, and only if they are, the ECC checking is performed. Figure 6.1 reports the power savings when the register file is protected through our RESI to cope with MBUs compared to *FullySBU*. As it can be seen, the power saving in all benchmarks is relatively high reaching 69% on average and up to 76%. In short, the main reason why our RESI technique has low power consumption due to the usage of a less complex ECC Encoder/Decoder and due to not performing ECC generating/checking with each 'write'/'read' operation.

It is noteworthy that because our architecture utilizes lighter ECC encoders/decoders independent of the actual ECC implementation, using an enhanced implementation of the Hamming SEC code (e.g., [145] as discussed in the related work) may lead to an additional power saving.

	FPGA		ASIC		
Design	Area (*slices*)	Delay (ns)	Area (μm^2)	Dynamic Power (μW)	Leakage Power (μW)
Encoder (8-bit)	3	6.2	64	15	0.31
Encoder (32-bit)	17	10.2	273	102	1.26
Decoder (12-bit)	14	9.0	160	33	0.71
Decoder (38-bit)	43	14.9	817	315	3.24

Table 6.1: Comparison of area, delay, and power for different Hamming encoders and encoders

6.2.3 Impact on temperature and performance

Our thermal setup mentioned in 6.1 has been employed to accurately obtain the chip's temperature after implementing different versions of the DLX processor. We also prepared a testbench which toggles all bits of the register file in an infinite loop (i.e. from leading '1's to leading '0's and vice versa), which represents the worst case scenario for the register file in terms of power consumption and it results in to maximizing the potentially generated heat/temperature. The processor thermal images of the three implementations are shown in the right side of Figure 6.4 and illustrate that the increase in the processor's temperature after implementing our technique is less than the temperature increment in the case of *FullySBU*.

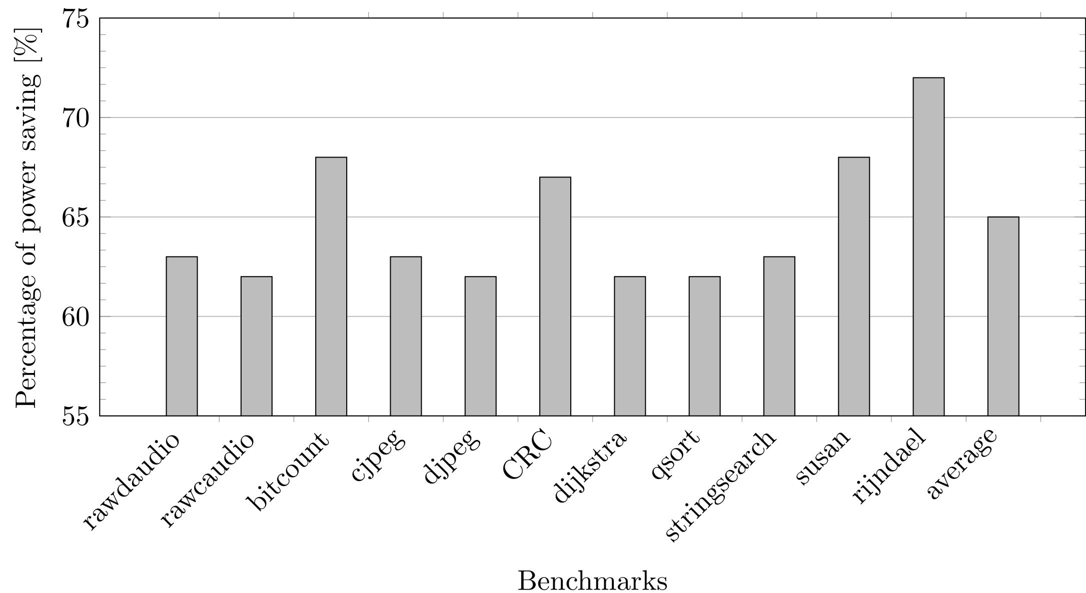


FIGURE 6.1: Power saving (for the case of ASIC estimation) in our RESI technique for MBUs compared to *FullySBU*, i.e. protecting the register file only against SBUs

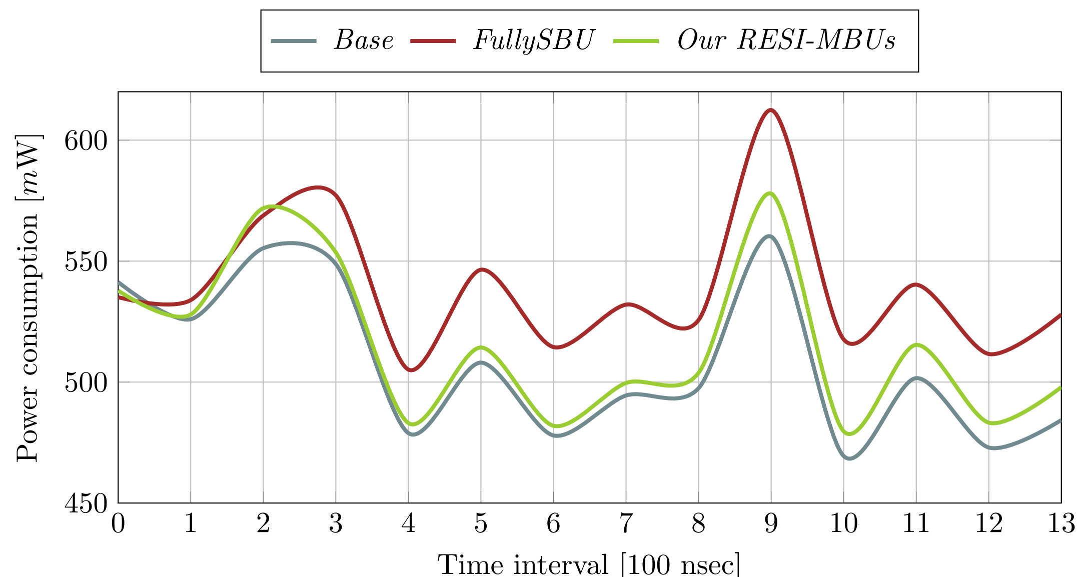


FIGURE 6.2: Total processor power consumption behavior during runtime (for the FPGA implementation) shows that our RESI technique consumes less power

To understand why our RESI technique keeps temperature low, it is worthy to note that it occupies similar area to *FullySBU* as discussed in Section 6.2.1. Furthermore, both techniques consume the same leakage power which is relatively constant in FPGAs at a given temperature. In other words, the difference in power consumption per area mainly comes from the dynamic power consumed from accessing the ECC encoders and decoders. For a deeper clarification, results in Figure 6.2 present the average consumed power of the processor during runtime estimated using the Xilinx Xpower tool [143].

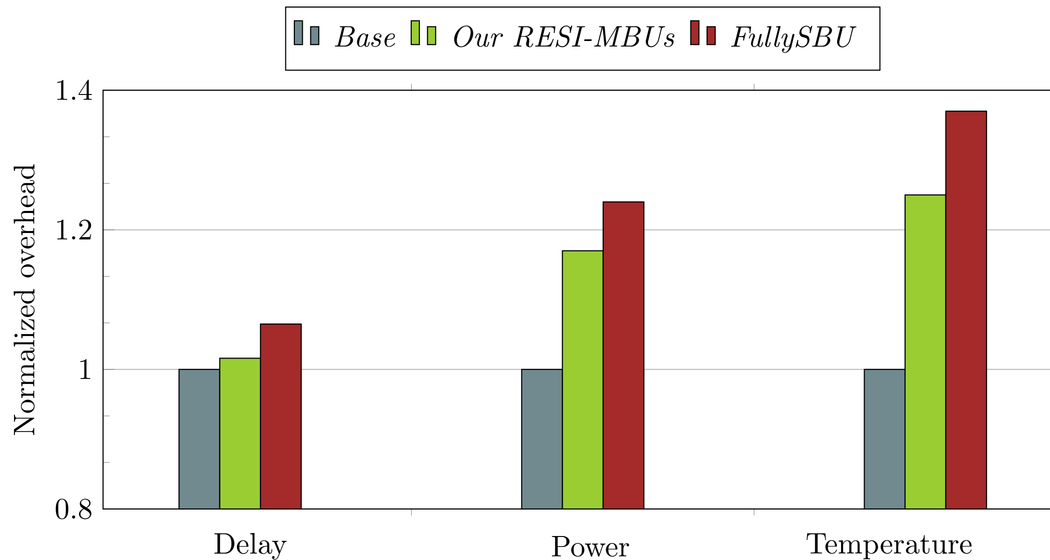


FIGURE 6.3: Normalized Overhead results compared to the *Base* version for the FPGA implementation

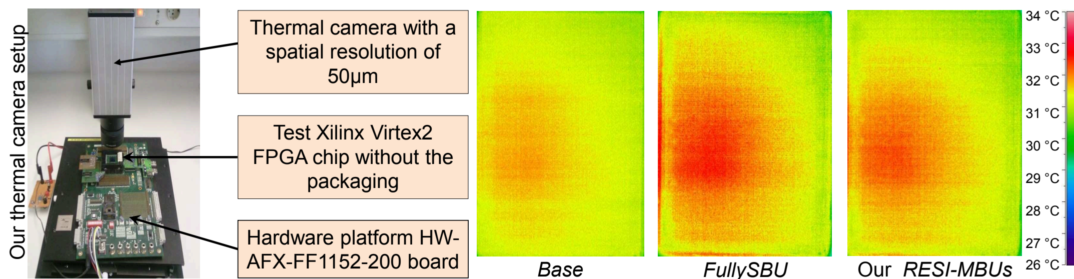


FIGURE 6.4: The used experimental setup for our thermal measurement (left) and the obtained chip thermal image of the three implementations

Because our architecture consumes less power at similar area footprint, its power density is lower, compared to the FullySBU case, which corresponds to less generated heat.

To measure the performance loss, the increase in the delay of the critical path is used as a metric. The overheads (normalized to the *Base* version of the processor) in terms of delay, power consumption and temperature are summarized in Figure 6.3. There, the presented power is the averaged power consumption estimated by the Xilinx Xpower tool [143]. As it can be noticed, we achieve 31%, 75% and 30% reduction in terms of power consumption, delay overhead and temperature, respectively compared to *FullySBU*. The improvement in delay is due to the fact that the latency of our ECC encoders/decoders is smaller (as they are less complex) compared to the larger ECC encoders/decoders in the case of *FullySBU*. The shown delay results in Table 6.1 also indicate to the aforementioned issue.

6.2.4 Register File Vulnerability

In order to investigate the advantages of our technique in terms of vulnerability reduction, we used the AVF metric which is described in Section 2. To this end, we extracted the vulnerable and invulnerable fractions of time for each bit in the protected register file (including the extra bits). For the original data word and the corresponding SEC code, any single error in the produced code word can be recovered [93]. Any MBU protection technique aims to correct the multiple occurred bit upsets in the targeted register. If it is capable to correct all of them (we target up to 3 bit upsets in a register value as earlier mentioned), then the corresponding vulnerable interval is considered as invulnerable since the corrupted value has been corrected. Considering the flag bits, since the self-π flag bit is not protected, it is considered vulnerable when the next access to its register will be a 'read' operation because the occurred bit upset can corrupt the read value and harm other CPU components as well and is considered invulnerable when the next access is a 'write' operation because the bit upset will be harmless due to the overwriting. The f flag bit vulnerability depends, however, on the self-π status (i.e. it is vulnerable when self-π is '1').

We compare our results with the *IRR* technique [99] and the partial protection scheme [101]. These techniques have been selected because they have a similar goal, i.e. increasing the register file immunity against soft errors with minimum overhead. Figure 6.5 reports the AVF reduction compared to the other competitors after implementing our RESI technique against SBUs (*RESI-SBUs*) and against MBUs (*RESI-MBUs*) (see Section 5.1). As shown, the achieved AVF reduction in all of the applications is high and reaches on average 88% and 91% for the case of (*RESI-SBUs*) and (*RESI-MBUs*), respectively.

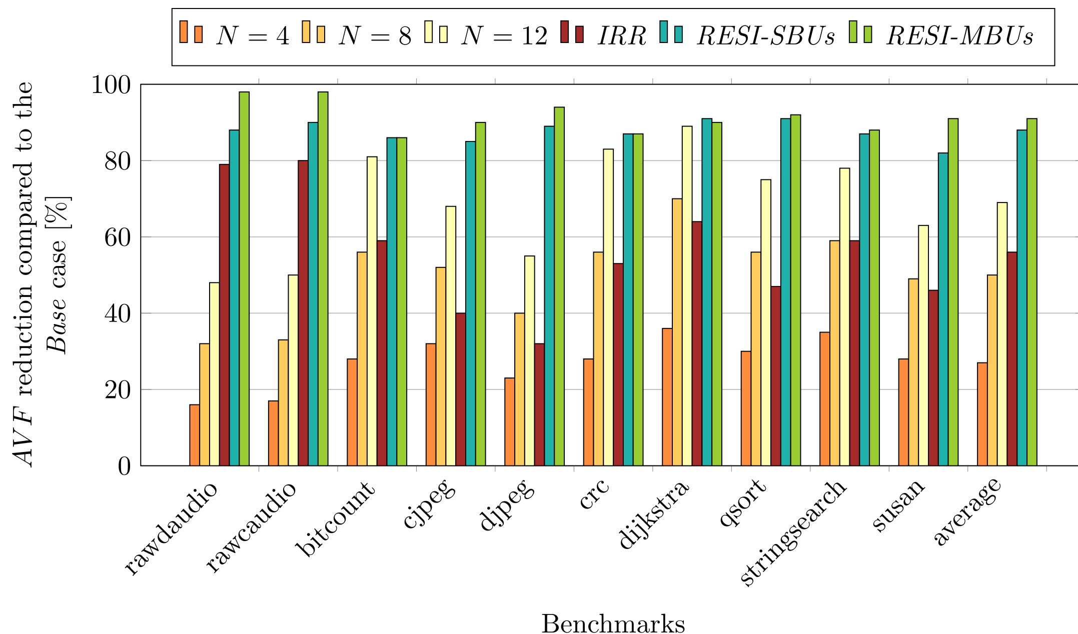


FIGURE 6.5: Comparisons of the register file vulnerability reduction. "N" represents the number of protected registers in the partial protection scheme

Compared to the partial protection scheme, we always achieve a higher *AVF* reduction for all of the varied cases of N. There, N is the number of the protected registers in the partial protection scheme. Similarly, we also achieve a better *AVF* reduction (around 63% on average) compared to the *IRR* technique.

6.2.5 Aging Stress Balancing

As discussed in Section 5.2.3, our RESI-MBUs technique, additionally, mitigates the aging effects in the register file SRAM cells. This is because that the *upper half* of the register file suffers from higher aging stress compared to the *lower half* as our analysis, in the absence of any protection technique, within Figure 4.1, illustrates. Therefore, embedding the generated bits of SEC codes in the upper unused/available bits of a register results in balancing the aging stress in these bits as they are no longer leading zeros or ones (i.e. continuous aging stress) over the most of execution time.

As earlier in Section 2.2.1 clarified, the deleterious effects of aging on 6-T SRAM cell are mitigated when the corresponding *duty cycle* is close to 0.5 (well-balanced case) because the stress on the transistors of the two cross-couple inverters will be evenly distributed. For that, to demonstrate how our technique relaxes the aging stress on the protected register file bits (i.e. register file bits as well as the extra bits), we present in Figure 6.6 the improvement compared to the *FullySBU* in terms of the percentage of the *duty cycle* values of the individual register file bits that are within the range of $\lambda \in [0.4, 0.6]$.

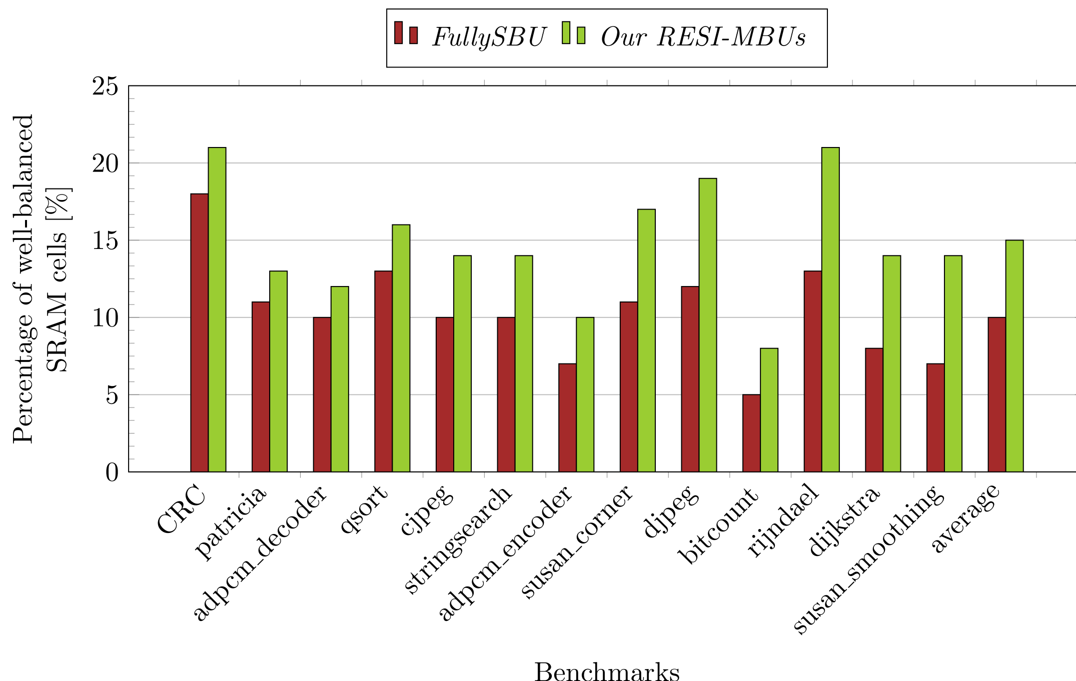
The reason behind choosing this range is that the reliability degradation for these *duty cycle* values is much smaller than the degradation when the *duty cycle* falls outside of this range. This is due to the fact that the minimum degradation due to aging occurs when λ=0.5 (as clarified in Figure 2.5) because aging stress will be evenly distributed between both cross-coupled inverters transistors existed in the SRAM cell (see Figure 2.3 and further details have been presented in Section 2.2.1). This observation is also consistent with other research [84, 85].

As it can be observed in Figure 6.6, compared to the *FullySBU* case, our RESI technique considerably increases (on average 47% and up to 100%) the percentage of the *duty cycle* values of the individual bits, that are within the targeted range where the aging-induced reliability degradation is minimum[2]. In fact, storing the ECCs in the upper *available bits* is the main reason behind balancing the aging stress there. In other words, instead of having a constant stress due to storing leading '0's/'1's most of the execution time, embedding the ECC bits results in frequently changing the applied aging stress which helps in balancing the corresponding *duty cycle*. It worthy to note that our investigation in Chapter 4 established that *frequently-written* SRAM bits have a better (i.e. more balanced) aging stress than *infrequently-written* SRAM bits.

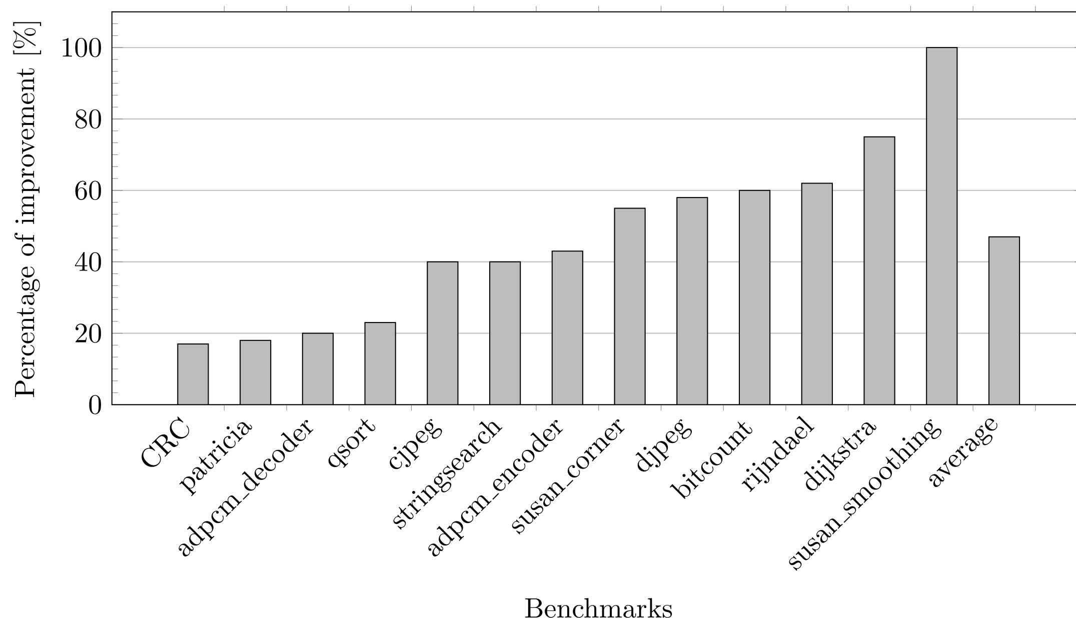
Moreover, the aging effects are directly related to the operating temperature (as established in Section 1.5) and, compared to *FullySBU*, our technique operates at a lower

[2]We found that storing the ones' complement of the generated SEC codes in the *AvB* results in better aging mitigation.

temperature as discussed earlier in Section 6.2.3. This, in turn, neutralizes the deleterious impact of temperature on aging when our RESI technique to increase the register file reliability is implemented.



(a) Comparison in terms of the percentage of the *duty cycle* values of the individual protected register file bits (i.e. SRAMs) that are within the range of $\lambda \in (0.4 - 0.6)$, where reliability degradation is minimum due to aging stress balancing



(b) The corresponding improvement analysis

Figure 6.6: Impact of our RESI technique on mitigating the aging effects

When we looked individually at the two flag bits (self-π, f) which are required in our architecture, we found that the corresponding *duty cycle* of each flag is not well balanced which makes both of them suffer from aging and may fail as a result. One solution to

address this point could be implementing the two flag bits in 8-T SRAM cells which are more resilient to aging than 6-T SRAM cells [146]. The 8-T SRAM cell comes with an additional two NMOS transistors compared to the 6-T SRAM cell. Therefore, the area overhead would be slightly increased but it might be tolerated as it would only be required for the two flag bits.

6.2.6 Fault Coverage Estimation

Towards further evaluation in terms of the impact of our introduced technique on the immunity of the register file against soft errors, we used a fault injection environment to estimate the system fault coverage, similarly to [147, 148]. In our fault injection environment, faults are injected on the fly while the processor executes an application. In each fault injection simulation, one of the 32 registers is randomly selected. Likewise, the cycle at which soft errors are injected is randomly chosen. In the case of injecting an SBU, one bit in the faulty register is picked at random and flipped. For MBUs, three bits in the faulty register are randomly chosen and then flipped. For a wider and more general investigation, we examine the case where these upsets occur only in *adjacent* bits as well as the case where they occur in *random* locations of the faulty register. We also consider a realistic distribution of MBUs generated by neutron irradiation [17], where the number of bits upset is up to 9 bits. Since injected faults might produce an infinite loop, a timer was implemented for the required number of execution cycles and we stop the simulation when the cycle count exceeds the number of cycles in the fault-free case.

When a simulation terminates, the corresponding output information (final results, content of the register file, execution time and state of the processor) is stored and used to classify the simulation to one of the following categories:
Issue-Less: The application normally terminates, the results are correct and the content of the register file is identical to that of the fault-free case.
Latent: The application normally terminates, the results are correct but at the end of simulation the content of the register file is different from that of the fault-free case.
Failure: The rest of the cases fall under this category and lead the processor to fail either in terminating normally or in computing the expected results in the time of the fault-free case (e.g., invalid address exception).

Each benchmark was simulated 1000 times. As a result, 1000 soft errors were randomly injected in the register file. This number complies with those used by other researchers and is chosen as a trade-off between accuracy and simulation complexity. In fact, we found that an increase from 1000 to 10,000 results in only a marginal accuracy increase (less than 2%) while the simulation time significantly increases (approximately by a factor of 10).

The *IRR* technique, similarly to ours, requires a flag bit associated with each register in order to distinguish between the case where the written value is replicated within the register (i.e. protection case) and the case where the value is written within the register without replication (i.e. non-protection case). Two parity bits (one for each replica) are needed to detect where the soft errors occurred and which replica is correct. Therefore,

we first inject the faults only in the register file (i.e. without the extra bits). In such a case, using the same amount of injected faults for both *IRR* and *RESI* techniques is fair because the targeted area footprint is the same. Then, we target the entire protected register file component (i.e. the original register file bits as well as the extra bits) and, there, the amount of injected faults should be increased to take the increase due to the extra bits into account, as later will be demonstrated.

Table 6.2 presents the results of the different scenarios in both cases of random and adjacent MBUs injections. As it can be seen, our technique maintains a very high level of fault tolerance compared to the other versions (*Base* and *IRR*) and it improves the register file resiliency against soft errors by considerably reducing the number of errors as is depicted in the Table. Since latent errors have no effect on the output of an application, they are less harmful. This means that we can safely add the *Latent* category to the *Issue-Less* category since in both categories the final results are still completely correct. Figure 6.7 reports the system fault coverage in the case of SBUs injection. As shown, our presented technique achieves better results compared to the *IRR* technique. The system fault coverage is on average 96%. In the case of injecting 10,000, it reaches 98%, on average, and up to 100% in some benchmarks.

			cjpeg	djpeg	bitcount	qsort	daudio
Issue-Less	*adjacent MBUs*	*Base*	536	536	467	590	230
		IRR	689	706	688	701	669
		RESI-MBUs	817	832	785	870	721
	random MBUs	*Base*	615	619	510	615	267
		IRR	706	720	702	728	686
		RESI-MBUs	814	866	779	823	689
Latent	*adjacent MBUs*	*Base*	117	133	78	139	237
		IRR	78	89	83	124	219
		RESI-MBUs	77	119	83	92	209
	random MBUs	*Base*	137	105	75	137	245
		IRR	75	80	77	116	198
		RESI-MBUs	75	87	72	116	208
Failure	*adjacent MBUs*	Base	347	331	455	271	533
		IRR	233	205	229	175	112
		RESI-MBUs	106	49	132	38	70
	random MBUs	*Base*	248	276	514	248	488
		IRR	219	200	221	156	116
		RESI-MBUs	111	47	149	61	103

TABLE 6.2: Processor behavior for adjacent and random MBUs injection per 1000 simulation runs

When injecting MBUs, the system fault coverage after implementing our technique is also high and reaches, on average, 86% and 80% for the case of adjacent and random MBUs, respectively, as shown in the Figures (6.8 and 6.9). Moreover, our RESI technique achieves better results than *IRR* technique in all of the various applications for both scenarios. In order to cover more scenarios, we also considered a realistic distribution of

MBUs generated per soft error due to neutron irradiation of SRAM presented in [17]. According to the applied distribution, the number of bits upset is up to 9.

Figures (6.10 and 6.11) report the results for the case of injecting adjacent and random MBUs. As shown, implementing our RESI protection technique makes the system effectively cope with MBUs. Indeed, RESI increases the system fault coverage by 44% and 20%, compared to the *Base* version, for the case of adjacent and random MBUs, respectively, and the system fault coverage in both cases reaches, on average, around 96%. This result, in turn, is higher compared to the results in the previous experiments (shown in Figures (6.8 and 6.9)) because the applied distribution also results in SBUs and 2-bit upsets.

Additionally, we demonstrate in Figures (6.12 and 6.13) the system fault coverage comparison for both adjacent and random MBUs, respectively, when the faults are injected in the entire protected register file component (i.e. register file bits as well as the extra bits). Since our *RESI-MBUs* requires more extra bits than the *IRR* technique and therefore occupies a larger area, the probability of particle hitting our protected register file could be higher. Thus, for a fair comparison, we inject 20% more faults more faults in the *RESI-MBUs* case which is a representative of the number of extra bits in our architecture. As it can be noticed in Figures (6.12 and 6.13), we still achieve better results compared to the *IRR* technique for the case of injecting adjacent as well as random MBUs, respectively.

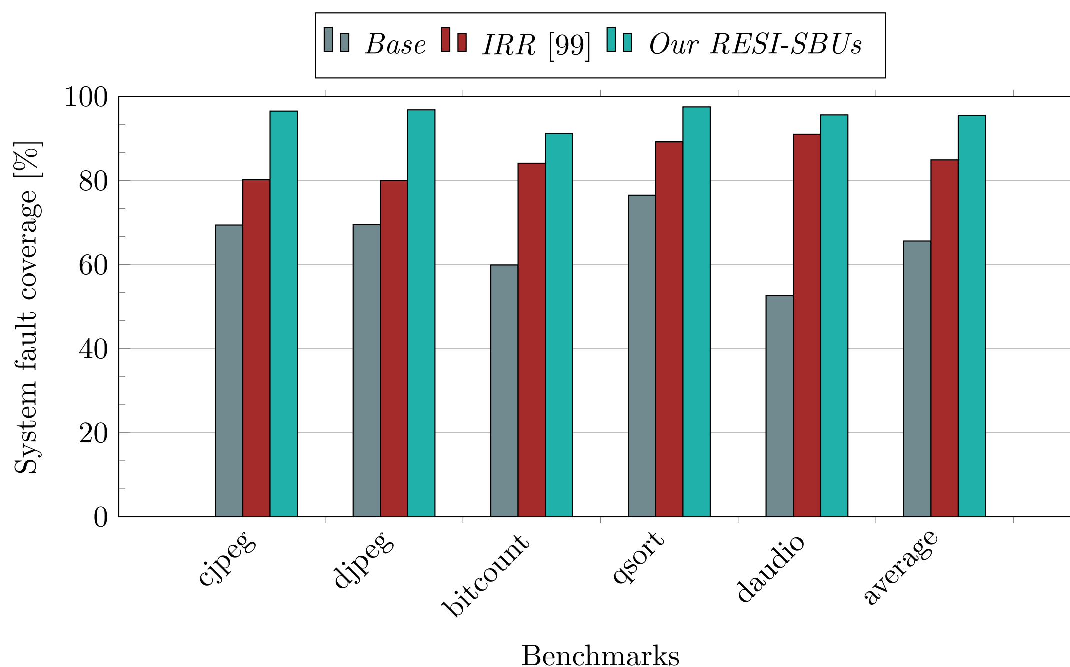


FIGURE 6.7: System fault coverage comparison for different benchmarks in the case of *SBUs* injection

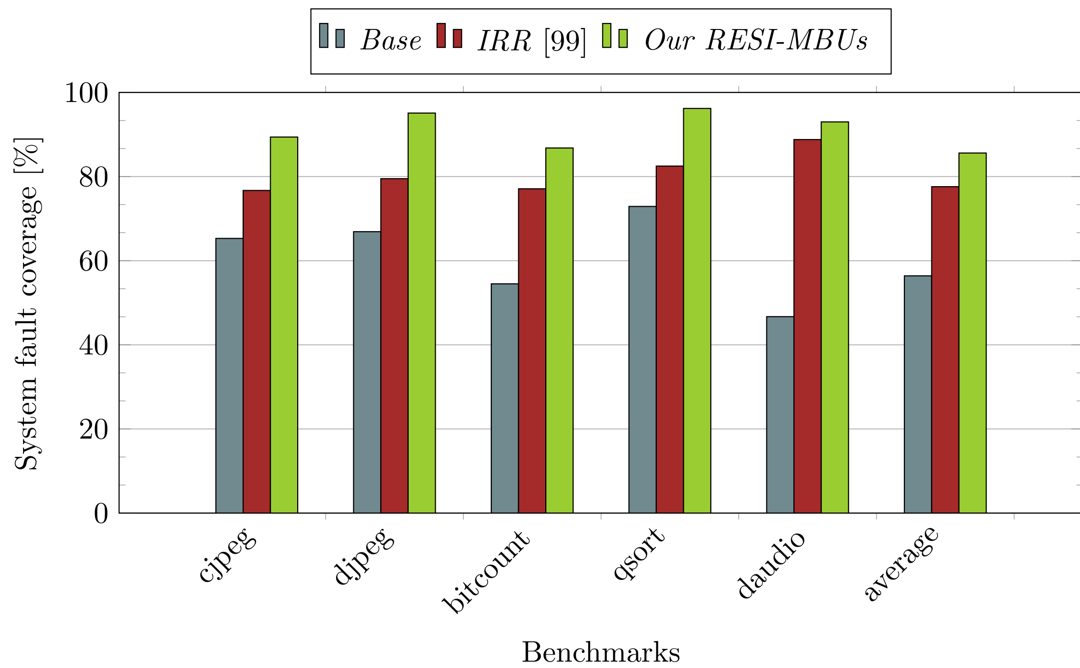


FIGURE 6.8: System fault coverage comparison for different benchmarks in the case of MBUs occurred in *adjacent bits* of the register file

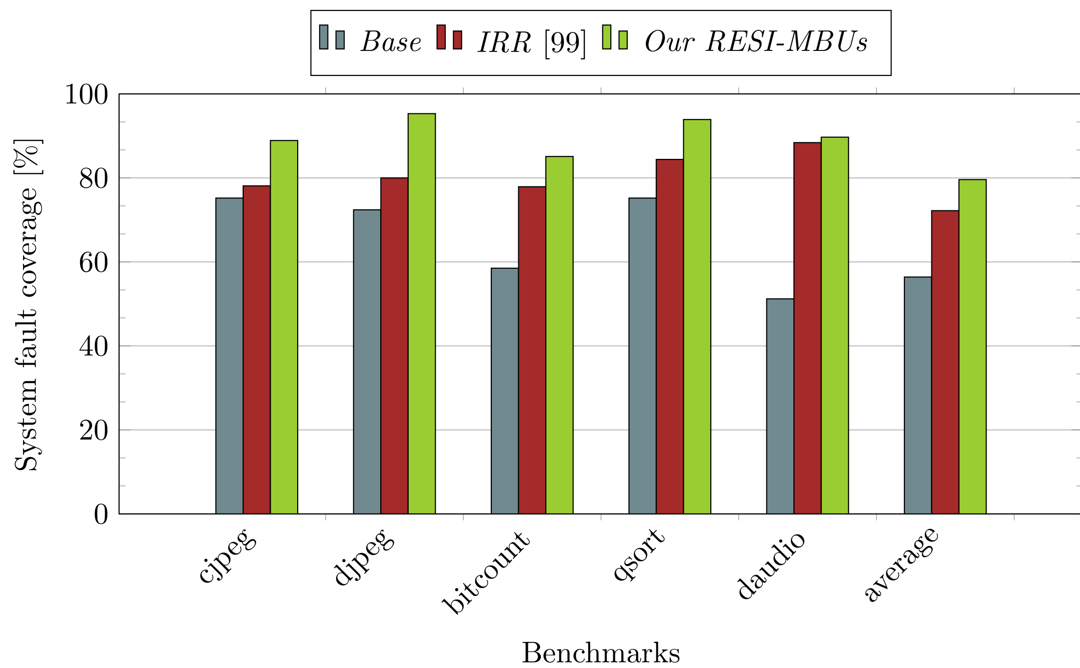


FIGURE 6.9: System fault coverage comparison for different benchmarks in the case of *MBUs* occurred in *random bits* of the register file

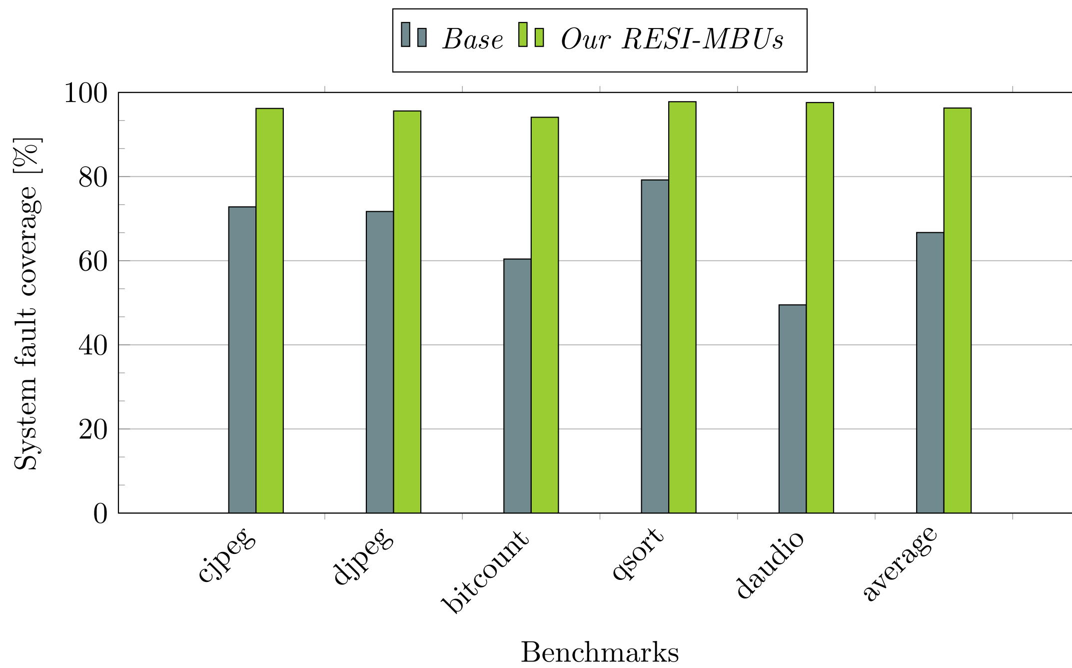


FIGURE 6.10: System fault coverage comparison for different benchmarks in the case of *adjacent MBUs* when a *realistic fault distribution* [17] is applied

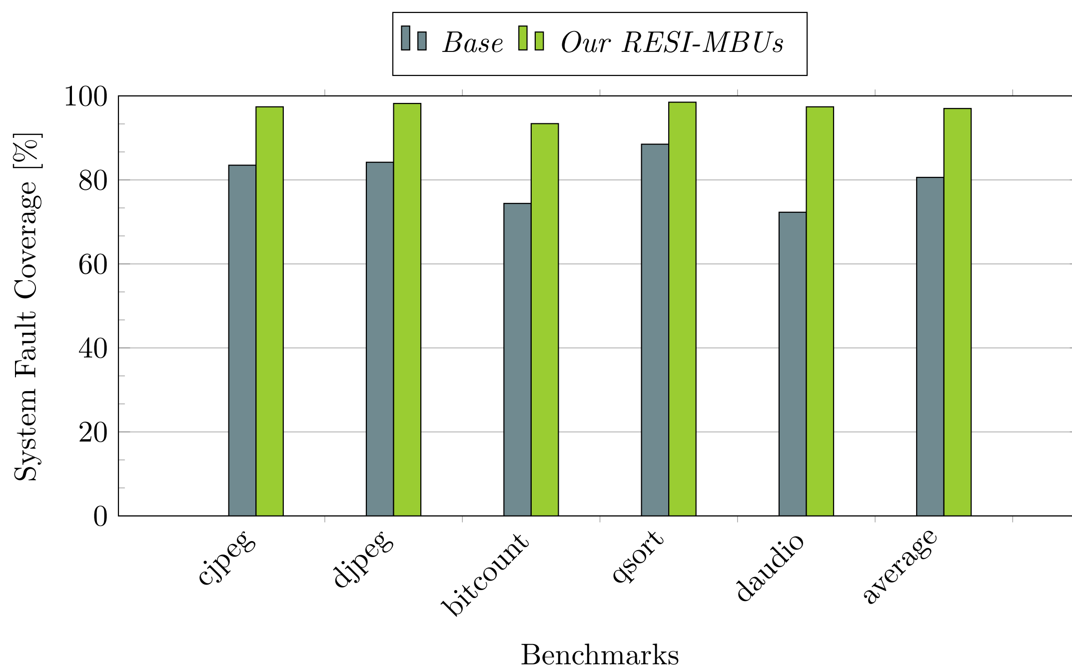


FIGURE 6.11: System fault coverage comparison for different benchmarks in the case of *random MBUs* when a *realistic fault distribution* [17] is applied

FIGURE 6.12: System fault coverage comparison for different benchmarks in the case of *adjacent MBUs* when a *realistic fault distribution* [17] is applied and the entire protected register file is under evaluating

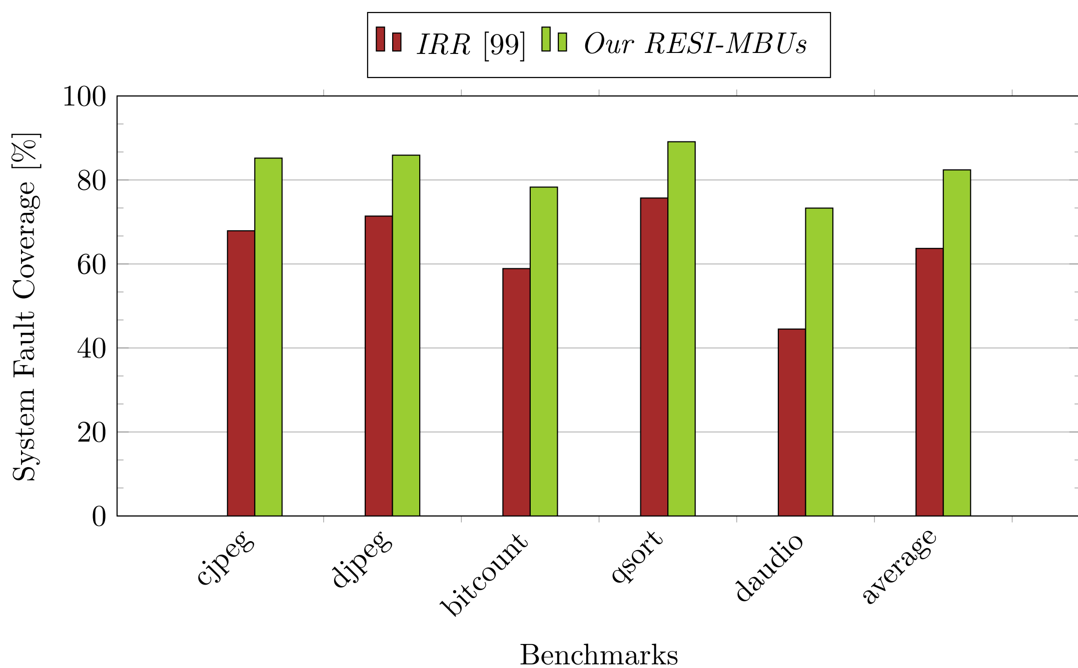


FIGURE 6.13: System fault coverage comparison for different benchmarks in the case of *random MBUs* when a *realistic fault distribution* [17] is applied and the entire protected register file is under evaluating

6.3 RISB Technique Evaluation

Similarly, we employ various diverse applications from the MiBench and MediaBench benchmark suites [26, 27] in order to evaluate the effectiveness of our proposed RISB technique. Such a wide range of applications exhibits varying characteristics enabling us to cover many different possible aging stress scenarios in the register file. The profiling information is gathered from an initial run of each application on a 32-bit MIPS architecture simulator. First we analyze the potential overhead of our technique and then we explore its impact on balancing the aging stress in the register file SRAM cells. Finally, we employ our proposed reliability estimation in Chapter 3 to evaluate the impact of our RISB technique on reducing the probability of failure due to aging along with a comparison to state-of-the-art.

6.3.1 Potential Overhead

To investigate the total overhead of implementing our technique, for both cases of *hardware/software*, in terms of area and power, the Synopsys Design Compiler has again been used. As explained earlier, the *frequent* registers are always relaxed in the hardware and we mean by the *hardware/software* case, the corresponding implementation to tackle the aging stress in the *infrequent* registers. As a matter of fact, our technique requires three flag bits associated with each *frequent register* in addition to the *infrequent state register* (only in the *hardware* case). The total area of extra components is 1402 μm^2 and 1073 μm^2 for the case of *hardware*[3] and *software*, respectively. The total additional consumed power, for each application, is demonstrated in Figure 6.14. On average, it reaches 129 μW and 109 μW which can increase the register file power consumption by approximately 1.8% and 1.4% for the *hardware* and *software* cases, respectively. From the power density perspective, our technique still roughly consumes a similar power per area resulting in operating at a similar temperature. This, in turn, neutralizes the influence of temperature on accelerating aging effects when our technique to mitigate aging is applied.

We also found that the extra three flag bits associated with each *frequent* register (as clarified in Figure 4.3) slightly increase the access latency with around 1.6% (roughly estimated by Faraday Memory Compiler Architecture [149]). Moreover, the conversion operation would affect the processor performance only if the decode/register-access would be on the critical path and reduce the frequency. However, this is not the case for the DLX and LEON3 CPUs when implemented on a Xilinx Virtex-5 FPGA.

On the other hand, our experimental results show that, on average, the number of cycles required for executing an application is only marginally increased (<0.01% on average). Last but not latest, despite the *infrequent* registers being scarcely read or written, we still investigate the total read/write operations occurring in the *infrequent* registers during the execution of an application to ensure that the XOR component, which is required to maintain a correct writing/reading, will not be often accessed. We found that, on

[3] The relative overhead is around 2.5% when the same TSMC library for the register file is used.

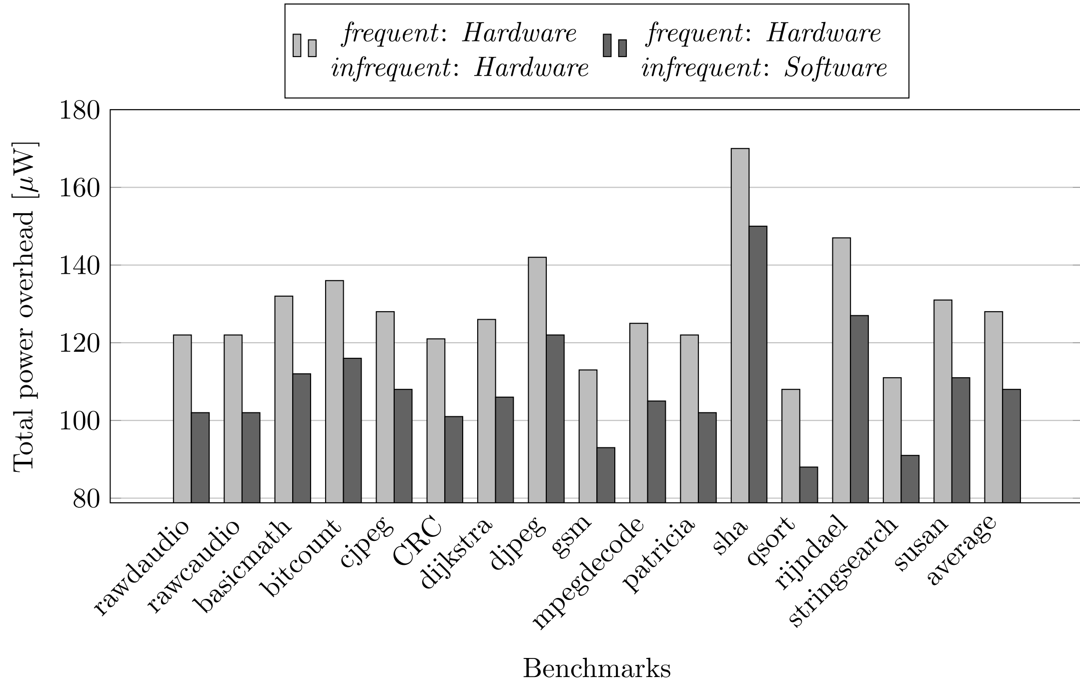


FIGURE 6.14: Total power overhead after implementing our introduced RISB technique to mitigate the aging effects in the register file

average, only 0.08% and 0.07% of the total write and read operations, respectively, occur in the *infrequent* registers.

6.3.2 Aging Stress Balancing

Figure 6.15 presents the aging stress analysis of the *lower* and *upper* halves of the register file and of the total register file after applying our technique[4]. As it can be seen, our technique drives the aging stress of the register file SRAM cells towards the well-balanced case of $\lambda = 0.5$. Moreover, the aging-induced stress is *evenly* distributed between the *lower* and *upper* halves of the register file (the aging stress of both of them is around 0.5) and the *upper* half is not anymore suffering from higher aging stress than the *lower* half as it was in the *Base* case, where no aging mitigation technique is applied to the register file, (see Figure 4.1(a)). Since our technique requires three additional flag bits, the corresponding *duty cycle* of each flag bit also needs to be analyzed. On average, it reaches 0.54, which reflects that the aging stress in the flag bits is also well-balanced. This is because that the flag bits are frequently updated resulting in avoiding the constant aging stress that may be applied to the SRAM cell transistors when the same logic value is stored for prolonged intervals.

[4]In this evaluation, the *hardware* approach has been considered to mitigate the aging effects in the *infrequent* category.

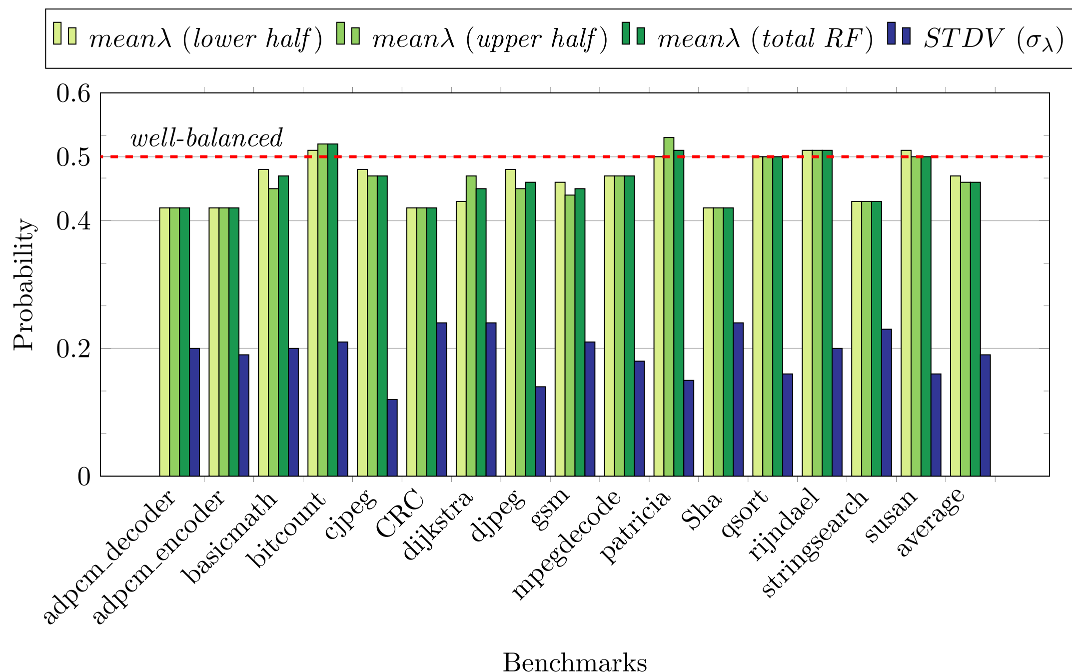


FIGURE 6.15: The aging stress evaluation in the register file after implementing our technique using the *small input* data set (profiling phase)

6.3.3 Sensitivity to input data

Generally, applications are subject to different input data during runtime than were used in the profiling phase. To explore how susceptible the results of our technique are to these inputs and provide a fair evaluation, it was necessary to repeat the experiments with different input data sets. Figure 6.16 shows the corresponding aging stress result for each benchmark. As shown, the aging stress of the *lower/upper* half and the total register file are still very close to the well-balanced value of 0.5 and the results are very close to those presented in Figure 6.15. In other words, modifying the input data does not significantly worsen the results and our technique does not rely too much on the input data set.

Towards more investigating of how effectively our technique balances the aging stress in the individual bits of the register file, we gathered the *duty cycle* (λ) of each bit to build a corresponding histogram. Figure 6.17 demonstrates a comparison, in terms of the percentage of the λ values that are within the range of [0.4 - 0.6], among the *Base* case and the two different aging relaxing strategies of our proposed technique. Indeed, this range in these experiments has been selected because the aging-induced stress is approximately balanced there and the reliability degradation is minimum. In the first case (called *fixed strategy*), the aging stress in all registers is tackled using the same strategy, i.e. we consider all the registers as *frequent* and similarly relax them as explained earlier (see Figure 4.3). Where in the second case (*selective strategy*), we selectively tackle the aging effects in the register file (i.e. aging stress in different registers are differently balanced/mitigated according to their classification *frequent/infrequent*).

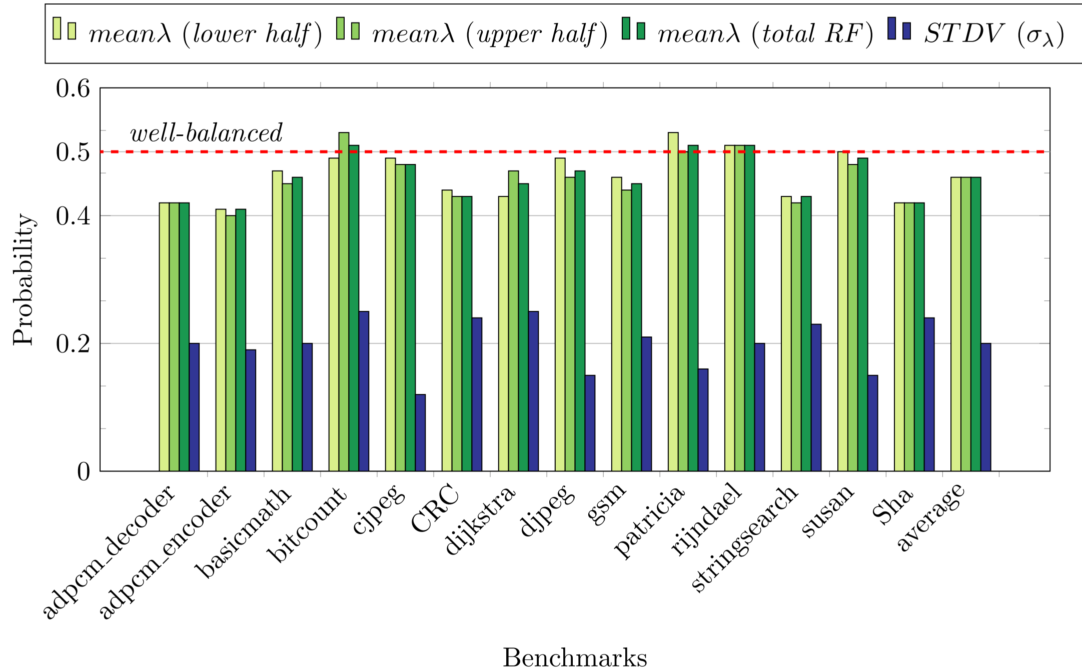


FIGURE 6.16: The aging stress evaluation in the register file after implementing our technique using the *large input* data set (execution phase)

As shown in Figure 6.17, mitigating selectively the aging stress in the register file, which represents the key idea behind our technique, is more effective than applying one aging strategy for all registers, i.e. treating all register similarly. Our technique (after implementing the *selective strategy*), across all benchmarks, considerably increases the percentage of λ values that are within the range of [0.4 - 0.6] and the increase reaches 3.3x, on average, and up to 6.2x. This, in turn, promises to mitigate the aging-induced reliability degradation in the register file because the aging stress is well-balanced in the majority of the register file bits.

6.3.4 Failure Analysis and State-of-the-art Comparison

It is worthy to note that other techniques based on toggling the entire register file at a statistical aging standpoint works fine only if aging stress remains approximately the same across the running applications. Varying aging-stress cases from multiple applications may be interleaved resulting in unbalancing the overall aging stress. Assuming two applications with 0.1 and 0.9 mean λ and a toggling interval corresponding to the time an application is scheduled, the overall λ from the toggling technique is $(0.1 + (1 - 0.9))/2 = 0.1$, which is far from the well-balanced case (0.5) achieved if both applications have the same stress, e.g. $(0.1 + (1 - 0.1))/2 = 0.5$. Indeed, there are two possibilities for employing the toggling mechanism. The first is not to apply it often. In this case the balancing of aging stress in the SRAM cells needs to consider additional effects such as application scheduling. It may not be possible to determine the combined effect of several applications during one toggling interval resulting in a non-deterministic stress balancing. To balance stress, the toggling needs to occur more

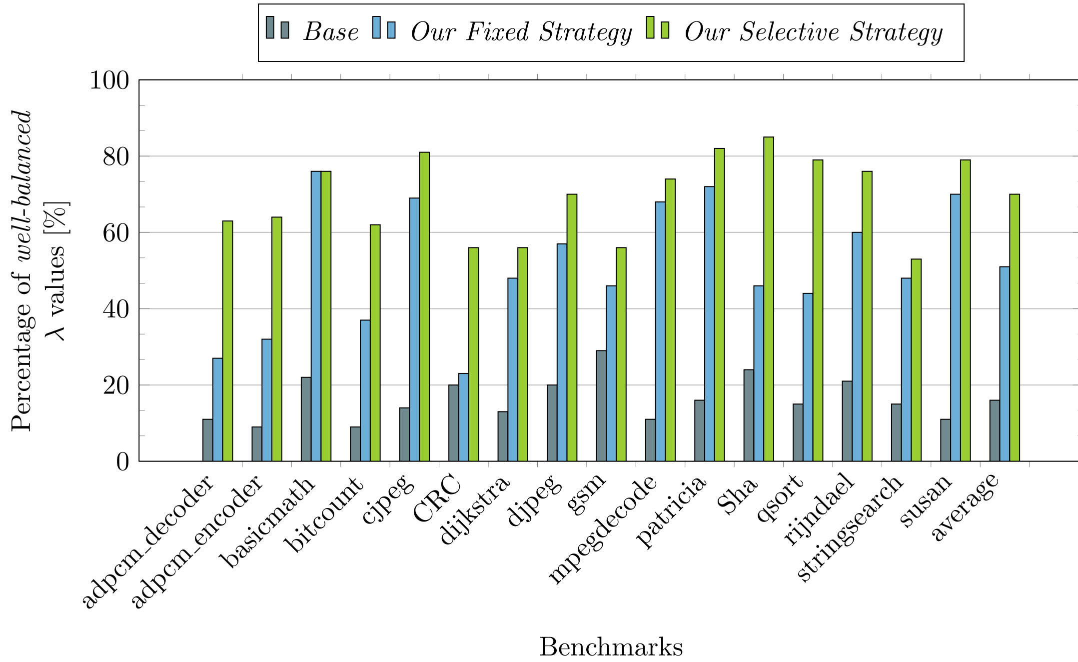


FIGURE 6.17: Percentage of the *duty cycle* (λ) values of the individual register file bits that are within the range of [0.4 - 0.6], where the aging-induced reliability degradation is minimum

repeatedly, incurring a much higher overhead through stalling the processor for several cycles each register file inversion. In contrast, our technique is designed to attain a well

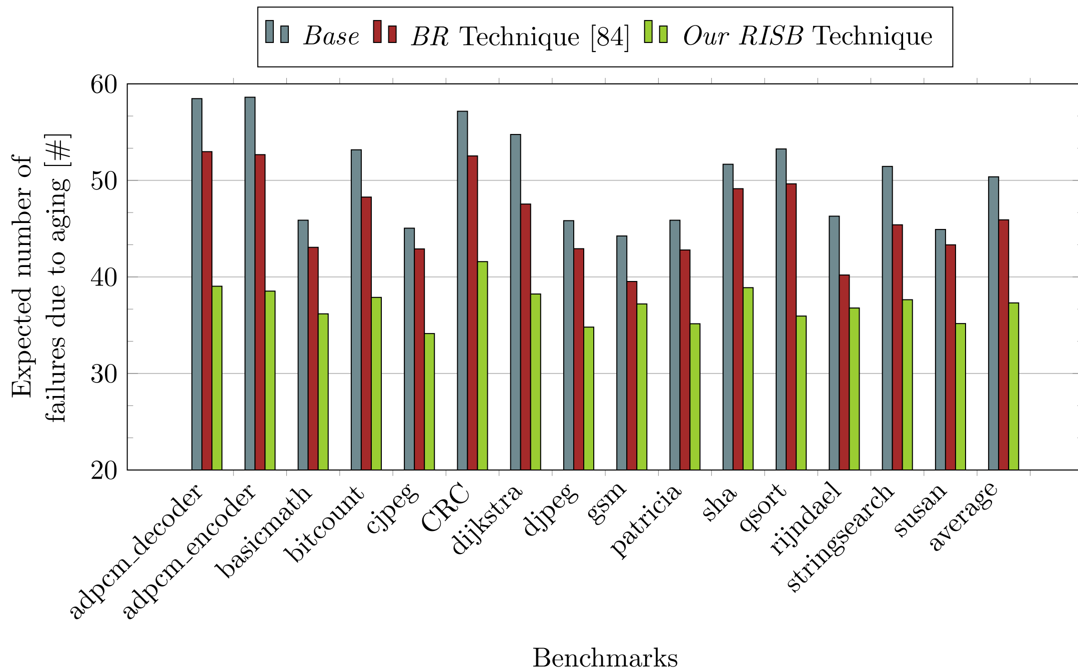


FIGURE 6.18: The reliability degradation for the 22 nm technology node after 10 years compared to state-of-the-art showing that our RISB always achieves better results

stress balancing with a little additional overhead independent of application behavior. According to our technique, all-zero bits in a *frequent* register are only periodically relaxed to '1's, they remain zeros roughly half of the time. Additionally swapping the *lower* and *upper* halves of the register bits compensates for non-balanced upper bits due to non-uniform distribution of leading zeros in subsequent register contents. On the other hand, the potential limitation of our proposed technique is that the static profiling fixes the number of *frequent* or *infrequent* registers at design-time. Without profiling, a runtime adaptive solution would require on-line monitoring to determine *infrequent* registers and then perform the required balancing.

Finally, Figure 6.18 evaluates our technique, in terms of reliability degradation, and presents a comparison with the *Base* case as well as with an implementation of the state-of-the-art "Bit level Rotation" (*BR*) technique [84]. This technique has been selected as it has a similar goal to our work (i.e. balancing the aging-inducing voltage stress in the register file bits). In this comparison, we consider the *selective* strategy as it, in practice, represents our proposed RISB technique.

The impact of the aging-induced degradations on the on-chip system has been evaluated through our reliability estimation that has been presented in Chapter 4.1). As earlier explained, our estimation abstracts the impact of aging effects towards the metric of probability of failure that covers aging-induced failures due to both data corruption as well as timing violations.

As it can be noticed, applying our RISB technique increases the register file resiliency against aging effects through effectively minimizing the P_{fail} and achieving better results in all the analyzed benchmarks compared to the *BR* technique. On average, our introduced technique improves the register file reliability by 26% and 19% compared to the *Base* and the *BR* technique, respectively. The reason behind achieving better results is that our technique has the ability to selectively apply the best-suited aging mitigation strategy for each register class.

6.4 Reliability Evaluation under *Conforming Workloads*

As it can be observed from the so far presented results in Figures (6.18, 3.13 and 3.14 and 3.15) in the scope of reliability estimation, the overall impact of aging on the register file is *driven* by the running applications on top of the embedded on-chip system (i.e. different applications result in different numbers of failures). This is mainly because that aging effects within the SRAM cells of register files depend on the induced aging stress there (i.e. *duty cycle*). The latter, in turn, entirely depends on the written values into the SRAM cells during the execution of applications. Therefore, in spite of the fact that aging effects are induced at the physical level, they are spatially and temporally *driven* by the running applications at the system level (see Figure 1.14). Beside the *duty cycle* factor that influences the aging effects, on-chip temperatures also play an essential role in this matter. As explained in Section 1.5, elevated temperatures stimulate aging more and thus they lead to higher degradations (see Figure 1.12).

The motivational example presented in Figure 6.19 highlights the differences between the aging-stress and temperature profiles (i.e. traces/waveforms) obtained from three applications (*barnes*, *dedup* and *fmm*) individually running on top of the Alpha processor. As it can be noticed, different applications may induce a high or low temporal variation within the profile. While an application (e.g., *dedup*) may induce, over runtime, a balanced aging stress, it may, on the other hand, results in a high temperature. This makes grasping the overall impact of aging at the system level in opaque.

In that motivational example and for the rest of this section, we employ rather than the *duty cycle* metric, to represent the aging stress within an SRAM cell, the *normalized duty cycle* ($\grave{\lambda}$) metric for the sake of clarity. It can be expressed as as follows:

$$
\begin{aligned}
&\grave{\lambda} = |\lambda - 0.5| * 2 \in [0, 1] \quad \textit{where} \\
&\grave{\lambda} = 0 \rightarrow \textit{balanced aging stress} \\
&\grave{\lambda} = 1 \rightarrow \textit{unbalanced aging stress}
\end{aligned}
$$

While, λ indicates that the aging stress within an 6-T SRAM is well balanced when it has a value of 0.5 and it indicates that the aging stress is unbalanced when its value is either 0 or 1, the *normalized duty cycle* ($\grave{\lambda}$) indicates that the aging stress is well balanced when $\grave{\lambda} = 0$ and it is unbalanced when $\grave{\lambda} = 1$. This, in turn, simplifies grasping the presented results in terms of aging stress.

An accurate reliability estimation in register files imposes analyzing the impact of aging in the presence of actual profiles of both aging stress as well as temperature that may be applied to their SRAM cells during lifetime. As matter of fact, the aforementioned challenge becomes more pronounced and is exacerbated when estimating the register file reliability under *conforming workloads*, i.e. as similar as possible to the typical scenarios that may be induced by the end-user software of the on-chip system where, for instance, multiple parallel applications along with an operating system are running on top of

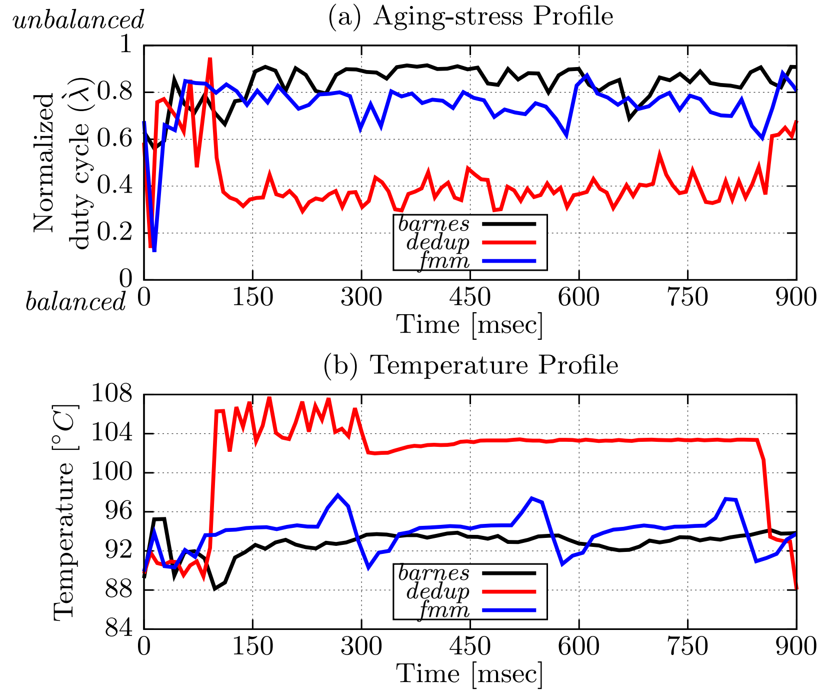


FIGURE 6.19: Motivational example showing how different applications may result in varied aging-stress and temperature profiles and thus they may differently stimulate aging effects (i.e. generate different number of defects at the physical level). Note that each data point represents an interval of 0.1 msec

the embedded on-chip system. This is because that when different applications run along with an Operating System (OS), the induced aging stress by them may interleave with the aging stress induced by the OS itself. Additionally, the OS scheduler, that is necessary to manage the execution of multiple parallel applications, makes the induced aging stress from different applications also interleave with each other. In other words, aging stress from different parallel applications may magnify or compensate each other in the presence of *conforming workloads* that may run on top of embedded on-chip systems. Therefore, estimating the reliability of on-chip systems under *conforming workloads* necessitates analyzing the running applications simultaneously and not individually.

In this section, we tackle this unexplored challenge in order to provide an abstracted, yet sufficiently accurate reliability estimation of the register file that takes into account the interrelations between the physical and system level towards grasping how aging actually degrades the reliability of embedded on-chip system under *conforming workloads*.

6.4.1 Connecting the System Level (i.e. Workloads) and the Physical Level (i.e. Aging Effects)

As a matter of fact, the induced activities by the running applications (e.g., read/write accesses to the register file) need to be monitored during their execution in order to extract the actual temperature and aging-stress profiles that are applied to each 6-T SRAM cell within the register file. In embedded on-chip systems, applications may run either without an OS (i.e. sequentially on bare metal) or on top of an OS that manages

the execution of multiple of them in parallel through its scheduler. While, the reliability estimation of the register file under the first scenario (i.e. bare metal execution) has been already performed in Section 3.6.3 along with the consideration of the worst-case operating temperature (i.e. $125°C$). Therefore, we focus in the following on extracting the aging-stress and temperature profiles that may be induced when *conforming workloads* are running on top of an embedded on-chip system (i.e. multiple parallel applications along with the OS that manages their execution) towards estimating the register file reliability under more typical/realistic scenarios.

For that purpose, we developed *software* and *hardware*-based techniques to investigate how the workloads at the system level *drive* aging effects at the physical level which, in turn, enables us to connect the physical and system level to accurately evaluate the overall impact of aging effects within the register file of embedded on-chip systems.

6.4.1.1 Software-based Technique

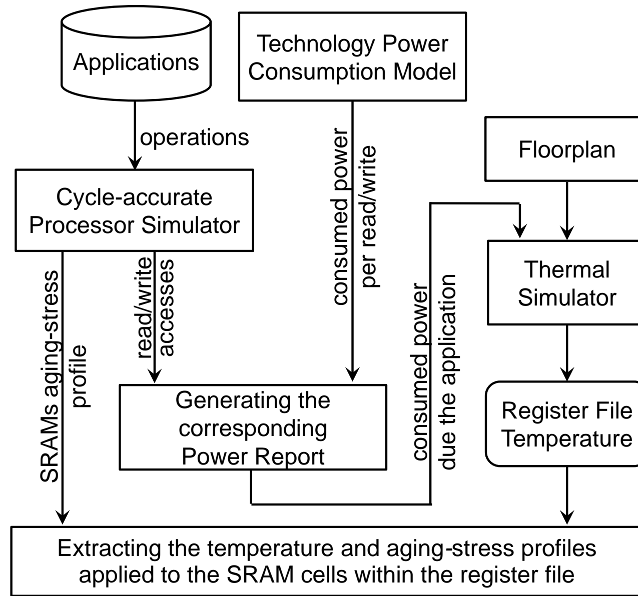
In this technique, we build a simulation tool chain to extract the induced temperature and aging-stress profiles by the running workloads. First, an instruction set simulator of the targeted processor architecture (e.g., gem5 simulator of the Alpha architecture [150]) monitors the activities of the running applications in order to provide the traces of the register file read/write accesses as well as it calculates the $\grave{\lambda}$ in each single SRAM cell within the register file. Then, a technology power consumption simulator (e.g., McPAT [43]) provides us, according to the targeted technology node (e.g., 22 nm), with the estimated dynamic power consumption per each read/write access to the register file and also with the leakage power consumption.

Afterwards, the consumed power over time along with the processor floorplan provides us with the on-chip power densities which then are passed to a thermal simulator (e.g., HotSpot [18]) to estimate the corresponding thermal map of the processor chip[5] including the register file microarchitecture component. We illustrate the required flowchart of the aforementioned steps within Figure 6.20.

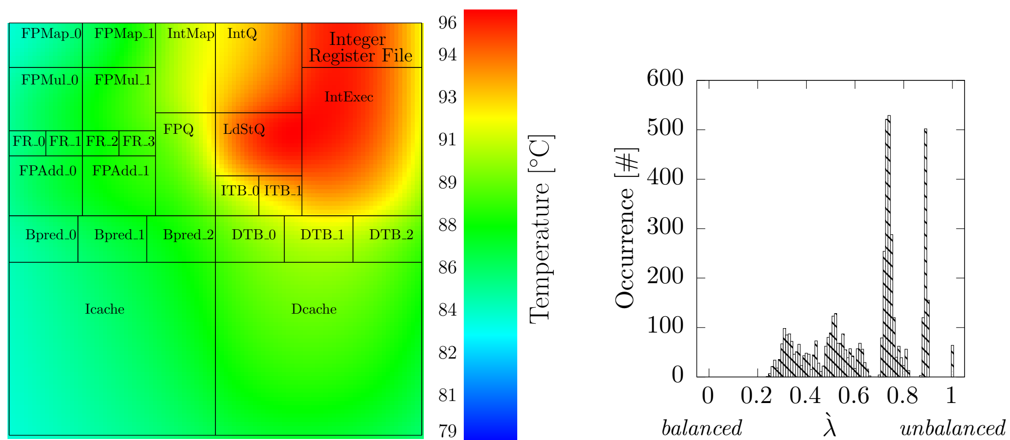
After extracting the required aging-stress and temperature profiles, we employ our reliability estimation that has been presented in Chapter 3 in order to estimate the overall impact of aging effects on the register file. Figure 6.21 summarizes the reliability estimation for each application. In these experiments, we employ diverse applications – exhibiting different behaviors – from the PARSEC benchmark suite [28], which is typically used in research for the Alpha processor. The sitting of 22 nm, V_{dd}=1.0 V and a lifetime of 10 years have been considered.

Key Advantages and Drawbacks: Such a *software-based* technique, while being general (i.e. applicable to any processor architecture kind), it unfortunately comes with a high computational time that is mainly due to the complexity of the simulated workload. For instance, in a reasonable time (e.g., several hours), the desired aging-stress/temperature profiles extraction for a workload (e.g, a single application from the

[5]Both of the steady-state as well as the transient temperatures are obtained.



(a) Flowchart of extracting the induced aging-stress and temperature profiles by the running workloads



(b) Simulated steady-state thermal map in the case of executing the *dedup* application on top of the Linux OS. The L2 cache component has been simulated but excluded from the plotted map for a better visibility of the spatial thermal variation across the presented thermal map

(c) Distribution of the *normalized duty cycle* ($\grave{\lambda}$) of the register file SRAM cells in the case of executing the *dedup* application on top of the Linux OS

FIGURE 6.20: *Software-based* technique to extract the impact of running workloads on *driving* aging effects in the SRAM cells within the register file. The Alpha processor has been targeted here, where its register file consists of 80 registers each register has a bit-width of 64 bits

PARSEC benchmark suite running on top on the Linux OS) can be performed but solely for a specific region of interest (i.e. not the full execution of the studied workload). In other words, this technique is only suitable to analyze superficial/representative workloads. As a matter of fact, *conforming workloads* contain an order of magnitude more operations and thus a complete simulation of them may not be feasible due to computational intensity. Therefore, we additionally introduce another technique that tackles this challenge and is able to provide us with the required analysis in the case of *conforming workloads*.

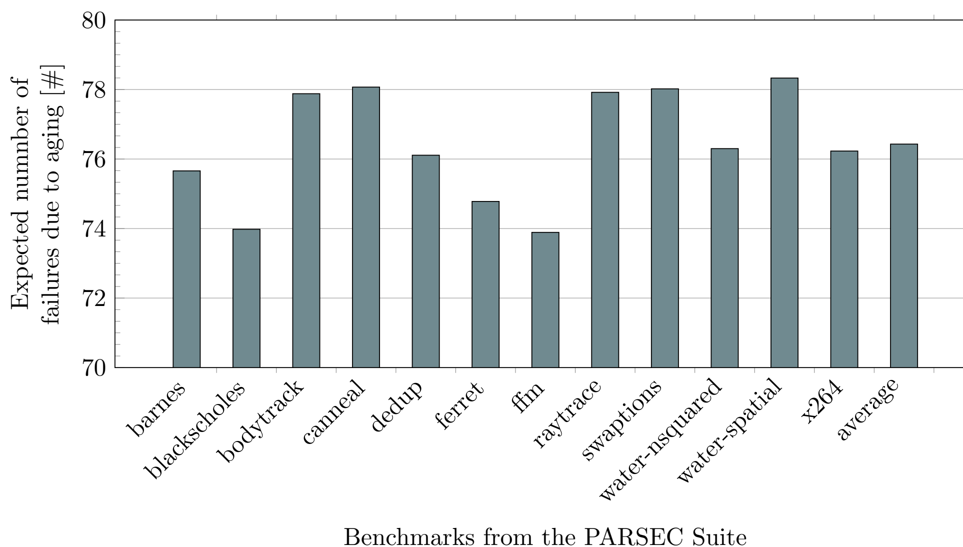


FIGURE 6.21: Register file reliability estimation in the case of running an individual application on top of the Linux OS

6.4.1.2 Hardware-based Technique

To tackle the challenge of estimating the register file reliability under *conforming workloads*, as the key focus of this section, a novel *hardware-based* technique has been developed, where the required analysis is extracted through a hardware platform instead of the simulation tool chain that has been employed in the previous software-based technique.

First, we implement the available open-source Register-Transfer Level (RTL) hardware design of the LEON3 embedded on-chip system in a Xilinx Virtex-5 FPGA platform. LEON3 is a 32-bit processor compliant with the SPARC V8 architecture [25] and it features a register file of 136 registers with a bit-width of 32 bits. Then, we implement our own hardware component to monitor the activities, that are induced by running workloads, within the register file. During the execution, our monitor component performs on-the-fly calculation of the induced aging stress of each cell within the register file. Afterwards, the entire *duty cycle* map of the register file cells is sent through a UART interface to a host PC that analyzes the data in order to calculate the *normalized duty cycle* map (see Figure 6.22(b)). Additionally, the temperature profiles are measured and then sent to the host PC through a thermal camera that accurately captures the IR emissions of the FPGA chip as Figure 6.22(a) shows. Our thermal measurement setup will be later explained, in detail, in Section 7.2.1. Finally, the obtained aging-stress map along with measured temperature is passed to our reliability estimation presented in Chapter 3 in order to be interpreted to the corresponding failure analysis.

The challenge behind implementing such a hardware monitor component is that several aspects need to carefully be taken into consideration in order to maintain a proper operation of the on-chip system. First, the processor critical path should not be negatively influenced by the implementation of the additional hardware monitor component.

Otherwise, the embedded on-chip system may crash during runtime due to timing violations. Secondly, the limited available area (i.e. reconfigurable fabric and resources) within the FPGA imposes an additional constraint restricting the monitor's implementation, i.e. the FPGA chip must jointly implement both the entire embedded on-chip system (e.g., CPU, memory interface, UART interface, Ethernet, etc.) along with our additional hardware monitor.

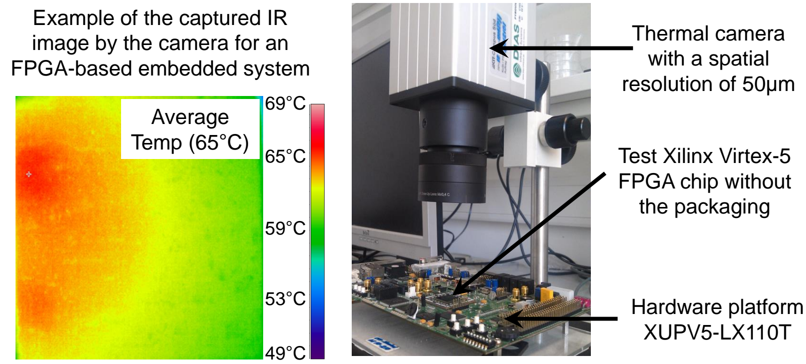
Key Advantages and Drawbacks: It is noteworthy that employing a hardware platform to run complex workloads provides a significant speed up of > 35x compared to the employment of an instruction set simulator [151]. This makes the analysis of the entire execution of *conforming workloads* feasible, unlike the previous *software-based* technique. However, the *hardware-based* technique is only applicable for architectures with an available open-source RTL design. Whereas, the *software-based* technique is more general (i.e. applicable to any architecture kind) as it does not require the actual RTL design of the studied architecture to be implemented.

Similar to the analysis in the previous technique, we also employ here our reliability estimation that has been presented in Chapter 3 in order to estimate the overall impact of aging effects on the register file. The sitting of 22 nm, $V_{dd} = 1.0\,\text{V}$, and 10 years lifetime have also been considered. Figure 6.22(c) summarizes the number of failures due to different workloads for the scenario of an application individually runs on top of the Linux OS. On the other hand, Figure 6.24 summarizes the reliability estimation for different cases of running applications in parallel. In these experiments, we employed diverse applications – exhibiting different behaviors – from the MiBench benchmark suites [27], which is typically used in research for the SPARC architecture. Details regarding the applications that have been selected to conduct these experiments are presented in Section 9.2.

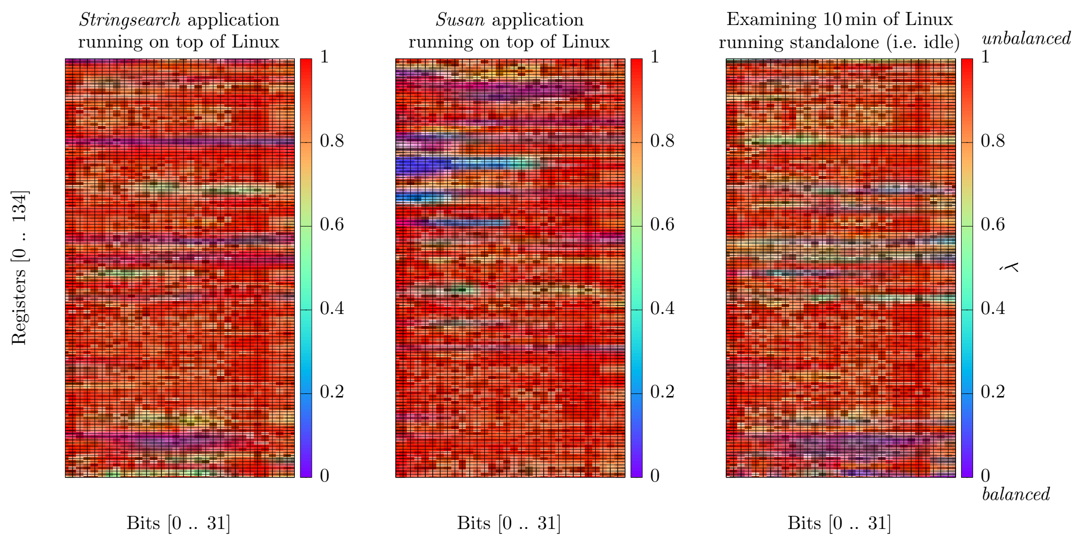
First, we present in Figures 6.23 the *normalized duty cycle* distribution of each application when it is *individually* executed on top of the Linux OS as well as the distribution when multiple parallel applications are *simultaneously* executed in order to demonstrate the impact of *conforming workloads* on aging effects. As it can be noticed, when an application is individually executed, the majority of register file SRAM cells has *unbalanced* aging stress (i.e. $\grave{\lambda}$ is close to 1.0). On the other hand, when multiple parallel applications are *simultaneously* executed, the induced aging stress is more *balanced* compared to the individual execution case. This is due the fact that the aging stress induced by different applications interrelate with each other (e.g., may compensate each other) resulting more *balanced* aging stress. This holds even more when more applications run in parallel as it can be observed when we compare between the *normalized duty cycle* distribution of the scenario of executing 2, 6 and 12 multiple parallel applications. In other words, executing more applications in parallel may result in a better aging stress balancing within the register file SRAM cells. As explained, the reason behind such a behavior is that the aging-stress profiles from the running parallel applications interleave with each other resulting in a compensation of the *unbalanced* aging stress.

Then, we present different comparisons in the Figure 6.24 (6.24(a) - 6.24(d)) in terms of the failure analysis when applications *individually* as well as *simultaneously* run on

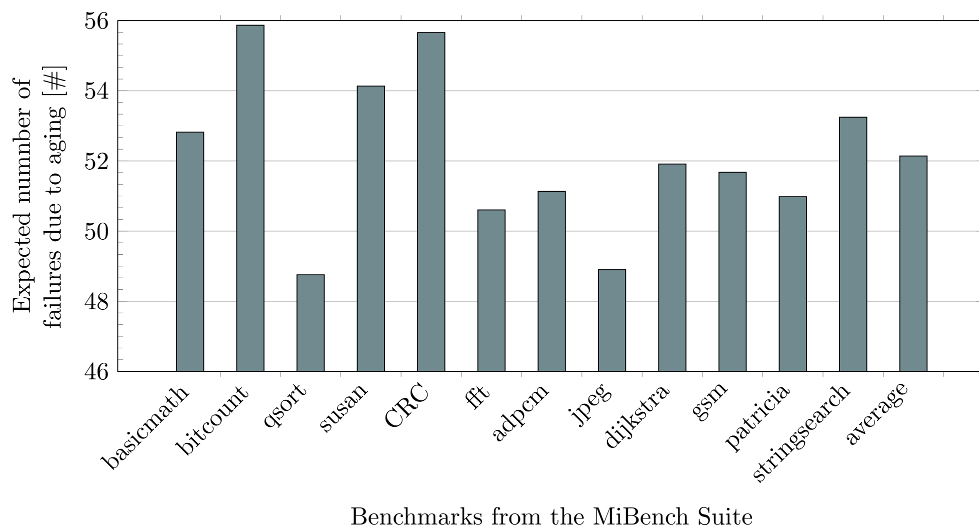
top of the Linux OS. As it can be observed, running applications in parallel results in lower aging-induced failures than running them individually. This is consistent with our previous observation obtained from Figure 6.23 which demonstrated that running more applications in parallel results in more *balanced* aging stress within the SRAM cells of the register file. Finally, Figure 6.24(e) summarizes the expected number of failures due to aging effects from different cases of *conforming workloads*: starting from running 2 applications in parallel and ending up with running 12 applications in parallel.



(a) Our hardware platform where the LEON3 processor has been implemented along with our component to monitor the activities within the register file



(b) Examples of the induced aging stress maps represented by the *normalized duty cycle* ($\grave{\lambda}$)



(c) Register file reliability estimation through our implementation presented in Chapter 3

FIGURE 6.22: Our *hardware-based* technique to extract the role of running workload on *driving* aging effects in the register file

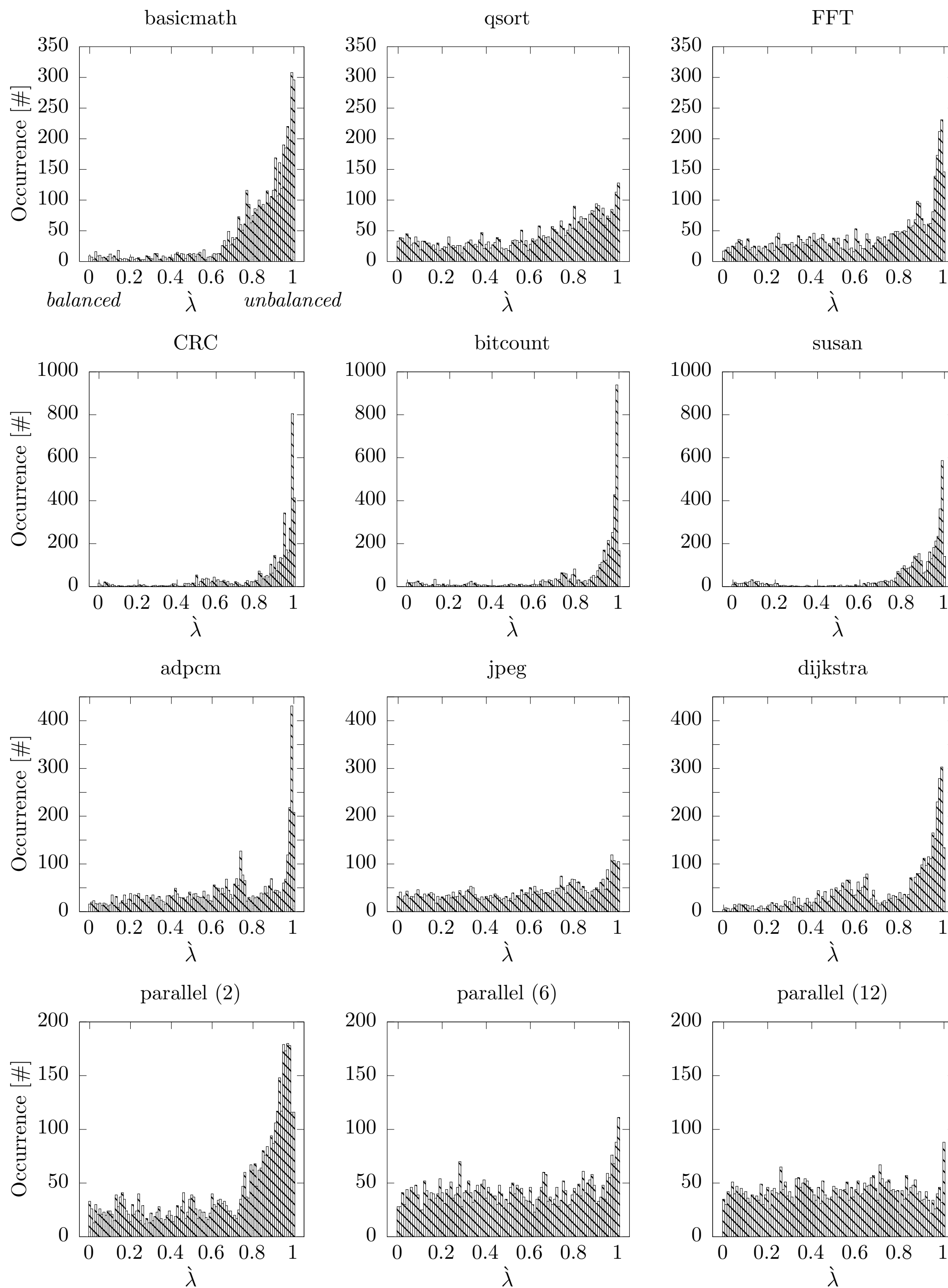
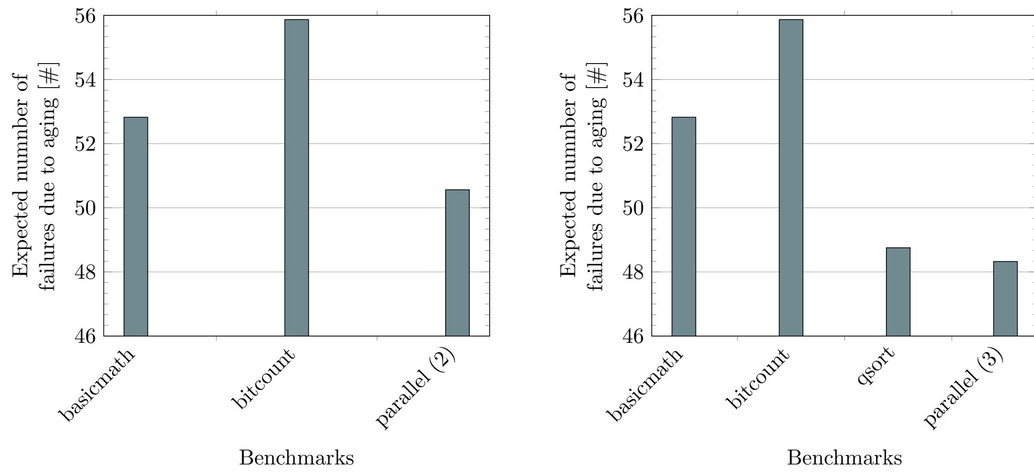
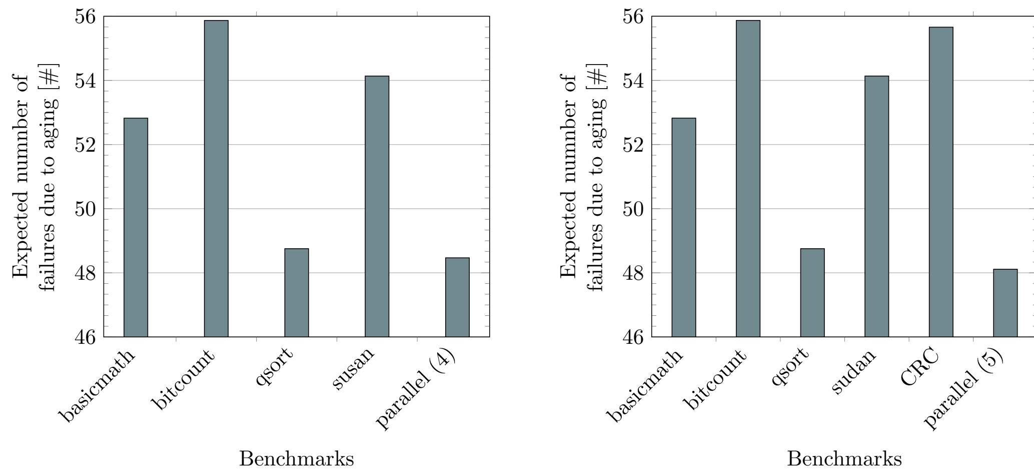


FIGURE 6.23: The aging stress histograms represented by the *normalized duty cycle* ($\grave{\lambda}$) showing the analysis of different applications *individually* as well as *simultaneously* running on top of the Linux OS



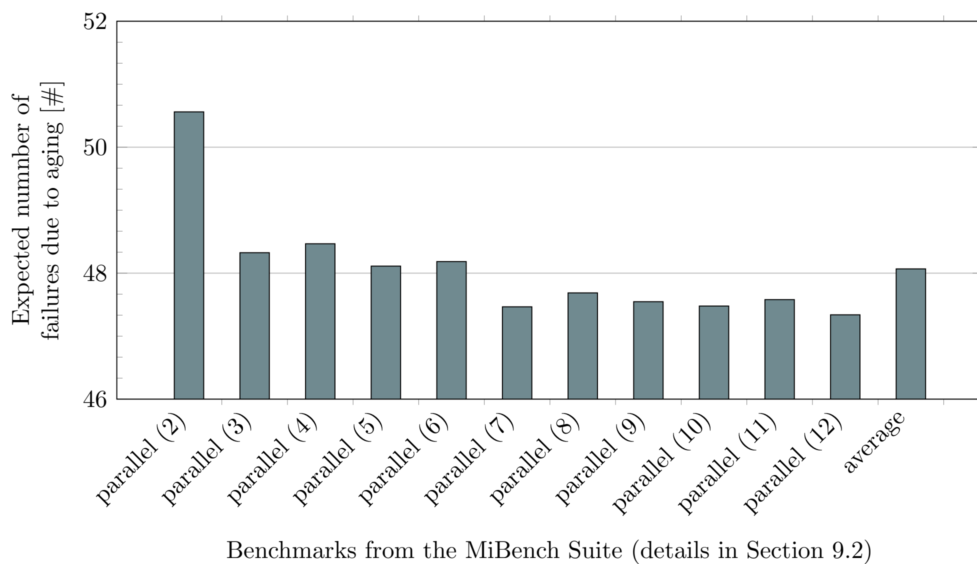
(a) Two parallel applications

(b) Three parallel applications



(c) Four parallel applications

(d) Five parallel applications



(e) Summary of multiple parallel applications

FIGURE 6.24: Register file reliability estimation in the scenario of applications running *individually* as well as *in parallel* on top of the Linux OS

Chapter 7

Lucid Infrared Thermography of Embedded On-Chip Systems

In this chapter we first motivate the need for a new methodology of thermal analysis to counteract the drawbacks of existing ones that have conventionally been employed to explore the thermal characteristics of on-chip systems. To this end, we present an investigation of both the stability of ring oscillator-based thermal sensors as well as the accuracy of simulation-based thermal analysis using an IR camera that captures the chip's thermal images. Then, we go thought our thermal measurement setup for FPGA-based on-chip systems along with an explanation of the *emissivity* aspect that may negatively affect the accuracy of the conducted thermal measurements. Afterwards, we evaluate the thermal behavior of commonly-used FPGA-based processors (i.e. Xilinx Microblaze, IBM PowerPC and Aeroflex Gaisler LEON3) to demonstrate the location of the thermal hot spot in these systems [6]. Then, we exploit this analysis in order to establish that the cache component in these processors plays a substantial role in their thermal behavior and thus a model linking the different cache configurations with the chip's peak temperature can be developed. Finally, we introduce our novel RAMA technique for lucidly capturing the thermal images of ASIC processor chips [3]. It employs our in-house IR-transparent cooling that cools down the measured chip from its rear side to keep its operating temperature within a safe range corresponding to the original cooling. This enables the IR camera to directly capture the thermal emissions and thus providing lucid thermal images which represents the key objective of our built thermal measurement setup. We also illustrate how our setup can be employed to support design-time exploration of the most important thermal characteristics of on-chip systems such as spatial thermal gradients and peak temperatures under different operating scenarios. The performed investigation of the thermal characteristics of Intel 22 nm *octa-core* processor through our RAMA technique demonstrates how state-of-the-art inaccurate thermal analysis leads to incorrectly estimating reliability in multi facets.

7.1 Motivational Case Study

While built-in thermal diode sensors are becoming ubiquitous in modern FPGAs, these are characterized by relatively large margins of error (±4°C) and are limited by the fact that there is usually only one of them [143]. Moreover, the fixed placement of such a sensor often make it unable to capture the peak temperature, particularly when the thermal hot spot is generated far from the sensor's location itself as we examine in this section. To overcome this limitation, most previous research relies on spreading an array of ring oscillators-based thermal sensors across the FPGA die to obtain the spatial temperature distribution or it employs thermal simulations to this end. In this section, in addition to analyzing the accuracy of thermal simulations we also discuss the shortcomings of other ways (i.e. thermal diode and ring oscillators sensors) towards justifying the need for an IR camera-based setup to achieve a more accurate design-time exploration of the thermal characteristics of embedded on-chip systems.

7.1.1 Thermal Simulations

Estimating the temperature of an on-chip system based on a thermal simulation always starts with having to estimate the runtime power consumption trace in different blocks[1] that form the analyzed chip. Then, the power information is used by a thermal simulator such as HotSpot [18] which evaluates an RC model to obtain the thermal map. In the following we explain how the aforementioned steps can be applied to conduct thermal simulations of FPGA-based systems[2]. To estimate the consumed power in the targeted FPGA chip, the XPower Analyzer tool from Xilinx [143] has been used because it gives more accurate values than the device power spreadsheet [143] which has been utilized in other studies (e.g., [105]). The XPower Analyzer takes as input the Value Change Dump (VCD) of the FPGA design, that describes the signals activity, and additionally the corresponding fully-routed Native Circuit Description (NCD) file, that represents the physical circuit description of the design. The Xilinx PlanAhead tool can localize the routing of the design. It takes the netlist file that describes the synthesized design and updates the User Constraints File (UCF) to guarantee that the design will be routed within the desired location.

As we are interested in the thermal distribution across the chip, the temperature estimating through the XPower Analyzer could not be employed because it only provides an average temperature for the entire FPGA. Instead, we turned to the HotSpot simulator [18] as it gives us an estimated thermal map that can be easily compared to the thermal image obtained from an IR camera. HotSpot takes the FPGA flooplan and the generated power distributions within different blocks across the chip to build the corresponding thermal map. To draw the required floorplan we divided the area of the target FPGA into equally-sized blocks. It is worthy to note that Xilinx does not provide

[1]In this scope, a block is a rectangular region of the FPGA which simplifies thermal simulation through abstraction.

[2]Please note that the required steps to conduct thermal simulations of ASIC-based on-chip systems have been already presented in Section 6.4.1.1 (see Figure 6.20(a))

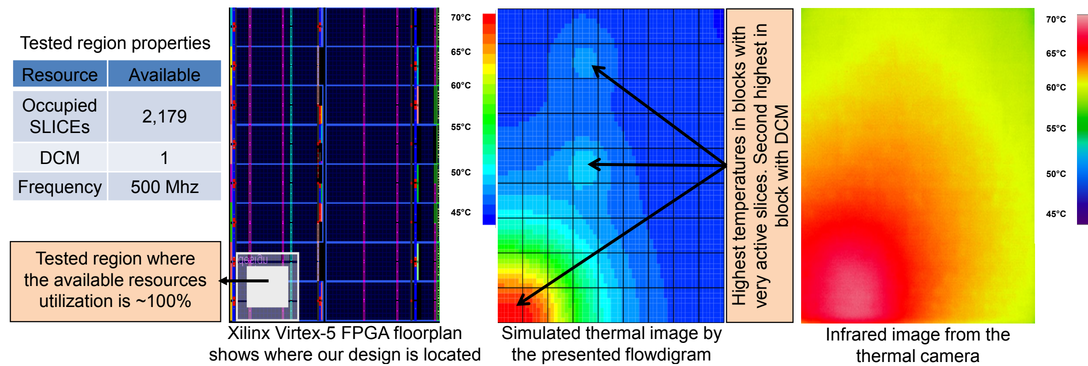


Figure 7.1: Comparison between simulated thermal map and measured IR thermal image showing that the simulated results display larger amounts of thermal variation due to inaccurate simulation of leakage power in blocks without active slices

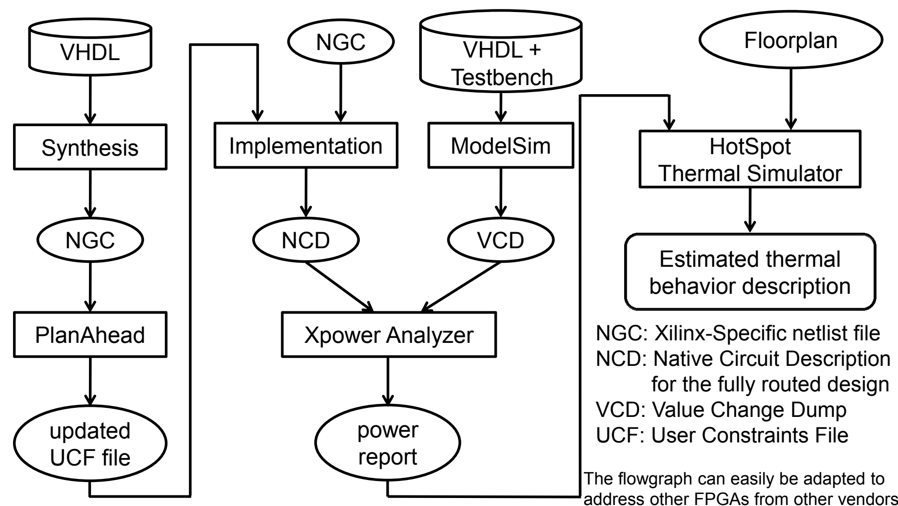


Figure 7.2: The simulation flowgraph to obtain the thermal image of an FPGA

the actual size of the FPGA die. Therefore, we measured an approximation of these dimensions after removing the chip's packaging. The flowgraph to obtain the simulated thermal map of an FPGA along with a comparison between the simulated thermal map and the actual IR image captured by a thermal camera as shown in Figure 7.1. Since the HotSpot thermal model always includes a heat sink and heat spreader their thickness was set as low as possible[3] in order to make their effect on temperature and temperature distribution negligible. This enabled us to fairly compare the simulated results with those captured by the thermal camera from a chip with its packaging removed (further details about the setup come in the next section). As can be observed, the maximum temperature of the simulation is comparable to the measured temperatures, but the maximum thermal variation over the chip, however, is significantly higher (25°C from simulation and 12°C from the thermal camera). This is due mainly to the inaccurate simulation of leakage power in blocks where there are no switching components. Additionally, the amount of detail in the temperature distribution is considerably lower since power values are used at the block abstraction level, i.e. each block has exactly one

[3]In practice, we used 100 nm as at smaller sizes the simulation fails.

summed power value and power variations within a block are not considered. The same holds for temporal power variations since XPower only provides one averaged power value over time. While this is sufficient for some designs, it can lead to inaccurate power estimation in highly-dynamic designs.

7.1.2 On-Chip Thermal Diode Sensor

In order to quantify the disparity between peak temperatures and those measured using the available on-chip thermal diode sensor manufactured at the center of the FPGA, we synthesized and implemented three different designs to take into account diverse possible stress scenarios in the Virtex-5 LX110T chip from Xilinx [143]. All of the designs occupy similar area but utilize different FPGA resources (e.g., LUTs, Block RAMs (BRAMs), and Digital Signal Processings (DSPs)). This leads to various power consumptions per area and various thermal behaviors as a result. Since the effect of ambient temperature in any thermal measurement cannot be neglected, we repeated all the experiments under two different ambient temperatures (20°C and 30°C). The thermal characterizations of the three designs are reported in Table 7.1 and the corresponding thermal images captured by an IR camera are presented in Figure 7.4.

The reason behind why these designs have different thermal characteristics is that all designs have exactly the same area footprint while consuming different amounts of power as reported in Table 7.1. For instance, the last design operates at the highest temperature compared to the others because it consumes higher power while still occupying the same area as the others resulting in the highest power density. As it can be noticed from the table, the on-chip thermal diode sensor fails to capture the peak system temperature – particularly when the thermal hotspot is located far away from the placement of the sensor which is fixed at design time.

Additionally, we present in Figure 7.3 how there is a considerable difference between the temperature at the center of the FPGA chip, where the on-chip available thermal diode is located, the actual chip peak temperature, where the thermal hot spot is generated. In this experiment, another FPGA family from Xilinx has been tested (Virtex-5 FX100T [143]) along with the implementation of an intense-stress design that results in maximizing the potential consumed power in a dedicated region that occupies around 10% of the total FPGA area. In practice, all the FPGA resources (i.e. FFs, LUTs, BRAMs, and DSPs) in that particular region have intensively been used to maximize the generated power density and thus examining the worst-case scenario in terms of temperature.

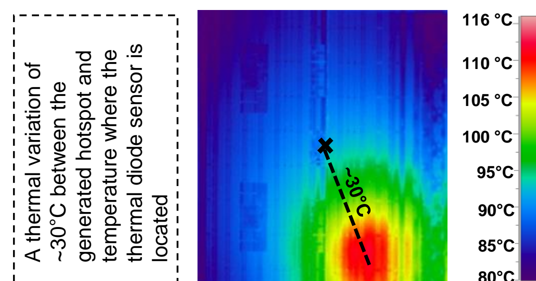


Figure 7.3: The thermal image shows that a single fixed thermal diode sensor may not capture a thermal hot spot when the chip is under an intense stress

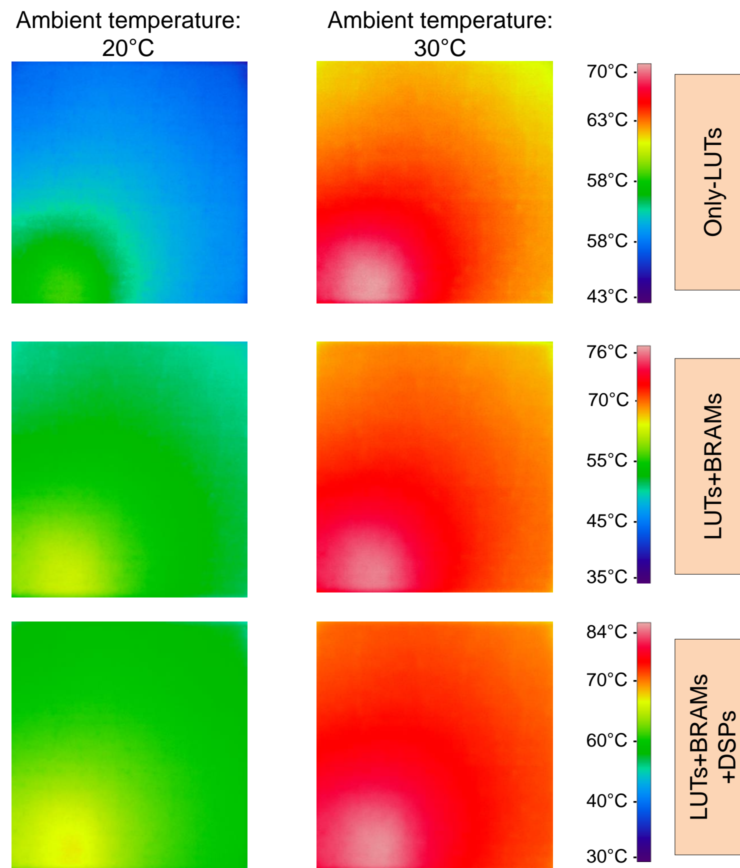


FIGURE 7.4: Measured thermal images of a Virtex-5 FPGA for various designs and under different ambient temperatures

TABLE 7.1: Thermal Behavior for different designs showing how the fixed thermal diode may fail in capturing the actual thermal hots pot

Design	Total Power (W)	Measured Temperature (°C)				
		Amb Temp	Max Temp	Avg Temp	Thermal Variation	Thermal Diode
Only-LUTs	2.69	20	58	51.6	10.6	55
	2.76	30	70	63.6	11.5	65
LUTs+BRAMs	2.80	20	62	54.4	13.5	58
	2.88	30	76.2	69	13.7	70
LUTs+BRAMs +DSPs	2.94	20	67.1	58.4	15.7	62.5
	3.02	30	83.5	75	16.5	77

7.1.3 Ring Oscillator-based Thermal Sensors

To overcome this limitation of the fixed on-chip thermal diode, modern chips (e.g., the Intel SCC [152]) distribute many *ring oscillators* (*RO*s) across the die, that may be utilized as thermal sensors after careful calibrations. The main idea behind *RO*-based thermal sensors is that the temperature increases microelectronic delays and therefore measuring the delay is a way to measure chip temperature through careful calibrations. A *RO* consists of a feedback loop that includes n delay elements including an odd

number of inverters (at least one) in order to produce the desired oscillation frequency, as illustrated in Figure 7.5.

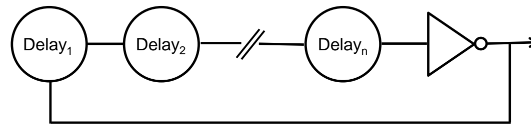


FIGURE 7.5: A ring oscillator-based thermal sensor which can be used to translate the increase of delay to temperature changes

The frequency of an *RO* depends on number of delay elements n and the temperature-dependent propagation delay of a single delay element τ. Then, the approximate frequency can be given as ($f = \frac{1}{2 \cdot n \cdot \tau}$). By feeding these oscillations to a counter, we can obtain a measure of temperature related to the counter value after a fixed period of time (i.e. *sampling interval*). As temperature increases, the counter value will be reduced because the delay of each logic will increase and thus it becomes slower. The ease of implementation of *RO*s have made them the temperature sensor of choice for reconfigurable logic [107, 108, 153].

When using an *RO* to estimate temperature, two parameters need to be adjusted to achieve a proper implementation. The first one is the *number of delay elements.* Having more delay elements will increase the overall delay of one oscillation and thus it will increase the temperature sensitivity of the *RO*. The number of delay elements also influences the effectiveness of the second parameter, the *sampling interval.* The longer the sampling interval, the less susceptible the ring oscillator is to measurement noise.

Due to the ability of run-time reconfigurable hardware in FPGA systems, the properties of potential thermal hotspots like the speed of temperature increase and the maximum temperature are not clear. Therefore, finding the most suitable *RO* parameters (e.g., number of delays and sampling interval) to accurately capture the generated thermal hot spot is not trivial which may make *RO*-based thermal sensors suffer from instability and inaccuracy during runtime.

RO accuracy is, additionally, plagued by their dependence on a stable supply voltage. Unfortunately, smaller process technologies are characterized by an increase in voltage supply noise. Zick et al. [108] mentioned this issue and tried to circumvent it by measuring the chip voltage and including it in their temperature calculations, but even this does not make up for local voltage fluctuations. To examine the issue we ran our own experiments and configured a mesh of 33 ROs covering half of a 40nm Altera Stratix IV FPGA. We found ROs to be accurate as temperature sensors when the FPGA is configured with a design with low amounts of switching activity, resulting in smooth temperature changes and an even supply voltage distribution, as observed in Figure 7.6 which shows the thermal distribution once the temperature stabilized.

On the other hand, configuring a micro-heater [109, 110] into the top corner of the FPGA causing an intense logic activity in the designated area results in severe instability of the *RO* readings.

The 3D plot [C2] showing the measured temperatures derived from the *RO* frequency at one instance in time can be seen in Figure 7.7 compared to Figure 7.6. It must be noted that it is representative of the locations where the ROs were picking up a large

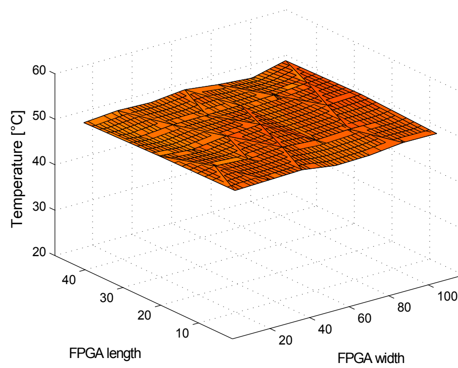


FIGURE 7.6: Estimated 3D thermal profile of a design with a low logic activity resulting in smooth temperature changes

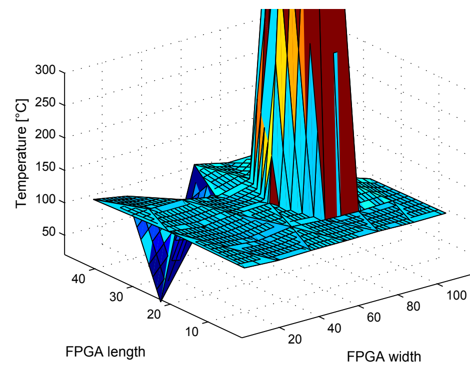


FIGURE 7.7: Estimated 3D thermal profile of an FPGA die in the case of intense logic activity leading to severe *RO* instability

amount of noise. The figure demonstrates that the introduction of intense logic activity in the FPGA fabric leads to sharp changes in temperature readings from the *RO*-based sensors. This makes them unusable as temperature sensors as measured errors reached a maximum of 10,000% with respect to the internal thermal diode sensor. And yet, with the exception of the one problematic sensor (recognizable by the negative peak), the erroneous behavior was contained within the area in which the micro-heater was placed. All other sensors displayed steady, accurate readings leading us to conclude that supply voltage fluctuations are localized. In particular, special care needs be taken when measuring the temperature of areas of intense activity.

In summary, thermal simulation may be imprecise to study the thermal behavior of the FPGA chip. Additionally, the thermal diode sensor may fail to capture the peak chip temperature and *RO*-based thermal measurement setups may suffer from instability. To overcome these issues, a setup employing an IR camera is required for accurate design time temperature exploration to obtain real thermal images containing the detailed thermal distribution of the analyzed on-chip system.

7.2 FPGA-based On-Chip Systems

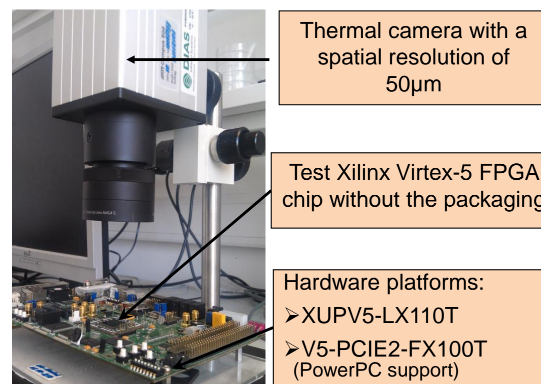
After going through the used IR camera-based thermal setup, we explore the thermal characteristics of FPGA-based embedded systems to capture where the thermal hot spot is often located. Finally, we propose a model to link the cache configuration of FPGA-based CPUs with the chip's peak temperature.

7.2.1 Experimental Setup

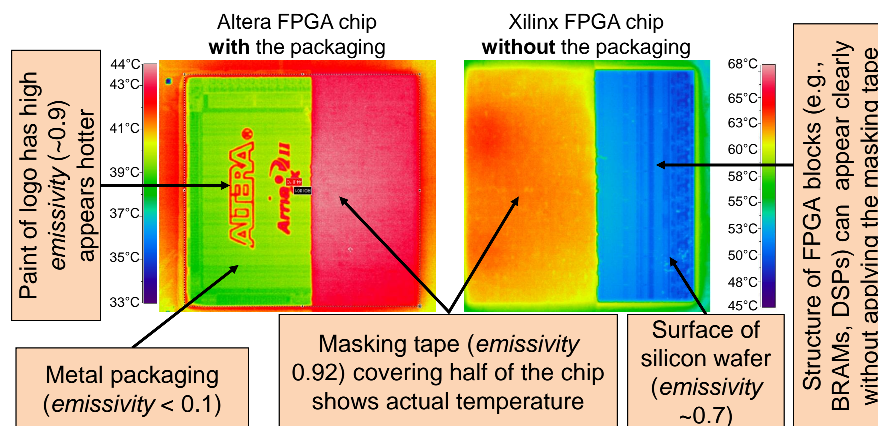
In our experimental thermal measurement setup, we consider 65nm Xilinx Virtex-5 FPGAs [143]. On-chip temperatures are directly measured using a DIAS pyroview 380L compact IR thermal camera capable of precisely capturing temperatures with an accuracy of $\pm 1°$C and a spatial resolution of 50 μm [144]. Figure 7.8(a) illustrates the experimental setup with the required equipment to take the thermal measurements. Once the readings have been obtained, the camera sends them at a frame rate of 50 Hz to a host PC which analyzes them to build the corresponding thermal image of the measured chip.

7.2.1.1 Emissivity Aspect

An important aspect to consider when performing IR measurements is the *emissivity* of the material. This property states what percentage of heat is emitted from the material in the IR spectrum. Ideal measurements can be obtained from a so-called *black body* with an *emissivity* of 1.0. Other materials, such as polished metal, have a very low *emissivity* of around 0.01. Figure 7.8(b) shows an *emissivity* test of an FPGA chip with and without metal packaging. There, the left half of the Altera chip appears cool due to the low *emissivity* of the metal. The actual temperature, however, is much closer to the right coated half of the chip. For this purpose, masking tape (with an *emissivity* of 0.92 [154]) is applied to the material's surface increasing its *emissivity* in the IR spectrum of the camera ($8 - 14$ μm). Masking tape was chosen due to its relatively high *emissivity* and it can easily be removed, compared to, for instance, black paint. The uniform distribution of the temperature measurement (shown in the uncoated half) is the result of the properties of the metal packaging that uniformly distributes the generated heat. To accurately determine local temperatures and thermal hot spots we have removed the packaging from the chip to expose the silicon wafer and carefully monitor the temperature distribution across the die as shown in the Xilinx FPGA case in Figure 7.8(b). The silicon die (in the right half) has an *emissivity* between $[0.75 - 0.9]$. We still apply the masking tape to improve the accuracy. Since the *emissivity* is known the camera software can internally compensate for the missing 8% temperature. As it can be noticed, the variation in temperature between the covered and uncovered halves in this example falsely appear to be up to 20°C. The measurements obtained with the masking tape may then be used for determining the *emissivity* of the exposed silicon wafer by comparison between the camera image and the chip's thermal diode readings.



(a) Our IR camera-based thermal measurement setup



(b) Emissivity aspect test

FIGURE 7.8: Our experimental setup used for thermal measurement and the *emissivity* tests of different chips. This shows that measurements of materials with low *emissivity* are very inaccurate as most of the heat measured is reflected from the surroundings

Since the *emissivity* is known, the camera software can internally compensate the missing temperature for an accurate reading.

7.2.2 Thermal Characteristics Investigation

In this section we examine the commonly-used FPGA-based embedded processors running benchmark applications in order to observe their thermal behavior using our thermal setup from Section 7.2.1. In these experiments we target the *soft* MicroBlaze processor from Xilinx [143] and LEON3 from Aeroflex Gaisler [25] as well as the *hard* PowerPC-440 processor from IBM [155]. A *soft* CPU is a synthesizable VHDL model of a processor that can be built using the FPGA resources whereas the *hard* processor is an ASIC core permanently embedded within the FPGA die. The MicroBlaze system is built by combining blocks of Xilinx IP cores to end up with a 32-bit RISC-based DLX architecture optimized for implementation in Xilinx FPGA reconfigurable logic. It is capable of single-cycle throughput under most circumstances and can be configured with a

3 or 5 stage pipeline. The LEON3 is also a 32-bit *soft-core* microprocessor based on the SPARC-V8 instruction set architecture and implements a 7-stage pipeline. On the other hand, the *hard-core* PowerPC-440 is a 32-bit high-performance superscalar embedded RISC processor consisting of a 7-stage pipeline. Unlike *soft* CPUs that can only operate at relatively low frequencies (around 100MHz), the *hard* PowerPC-440 CPU is able to run at higher frequency of 400MHz. For a fair comparison we use the same FPGA family (Xilinx Virtex-5) to study the three targeted embedded processors.

Our key observation after running different applications from the MiBench [27] Benchmark Suite on the aforementioned FPGA-based processors[4] is that the peak temperature of the FPGA chip when running only from on-chip memory is always in a relatively low temperature range with an average temperature of 55°C. On the other hand, we measured a drastic increase in the chip's temperature exceeding 70°C when a memory interface (e.g. DDR controller) is part of the on-chip system. The IR thermal images in Figure 7.9 show that the thermal hot spot is located on the border of the FPGA chip where the DDR memory interface is implemented[5].

This observation can be explained by the fact that the generated temperature is directly related to the consumed power per area. The consumed dynamic power can be described by the following equation [156]:

$$P_{dynamic} = C_{eff} * f * V_{dd}^2$$

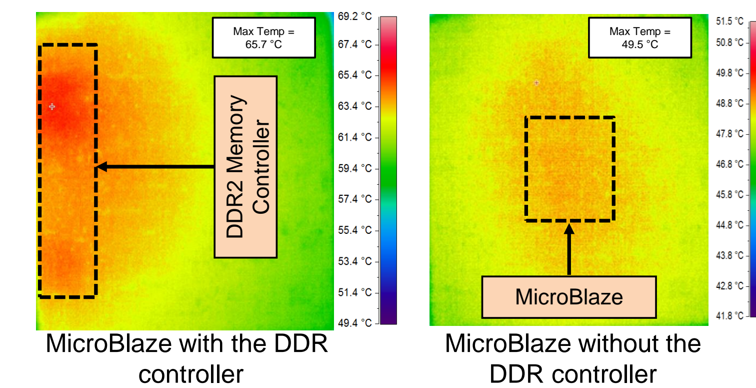
where C_{eff}, f, and V_{dd} are the load capacitance, frequency, and supply voltage, respectively. The load capacitance associated with the input/output pins of the DDR memory interface is significant and the switching frequency is high at 400MHz (transfers at double-data rate with a 200MHz clock). Moreover, the supply voltage of the pins ($2.5V/1.8V$) is considerably higher than that of the rest of the FPGA fabric ($1.0V$). Combined with the fact that a memory interface is made up of many pins (e.g. 113, 115 and 117 pins for the DDR2 memory controller in the MicroBlaze, PowerPC-440, and LEON3 CPUs, respectively) it is obvious to prognosticate that the inclusion of the DDR memory interface in an FPGA-based embedded system results in a thermal hot spot in that location. We also found a very similar observation when interfacing the system with an SRAM memory chip.

7.2.3 Evaluating the Thermal Impact of Cache

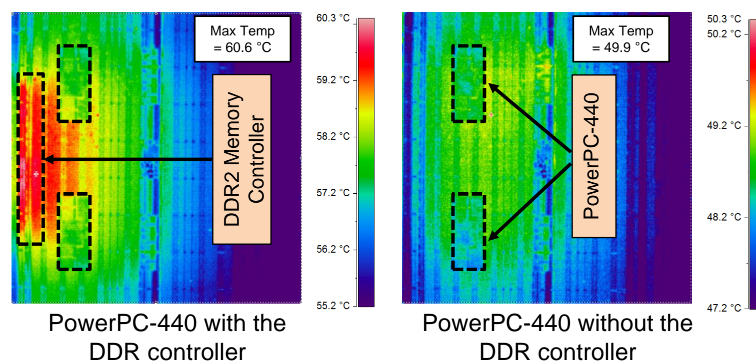
Having determined that frequently accessing the DDR pins significantly increases the temperature of the FPGA chip, we investigate whether the inclusion of a cache in the system impacts the temperature in a meaningful way. A cache miss will force the system to retrieve data from the external memory, while a cache hit means that the CPU can retrieve the data from the cache. For the same cache associativity, and line size, a larger cache generally means fewer misses which leads to fewer DDR memory accesses. Given

[4]We have run the *soft* and *hard* CPUs at their highest possible frequency.

[5]We found that the DDR controller in Xilinx Virtex-5 FPGAs is implemented as a hard-macro that is permanently placed on the side of the FPGA's die adjacent to the DDR memory chip.



(a) Example of the measured FPGA-based *soft* processor

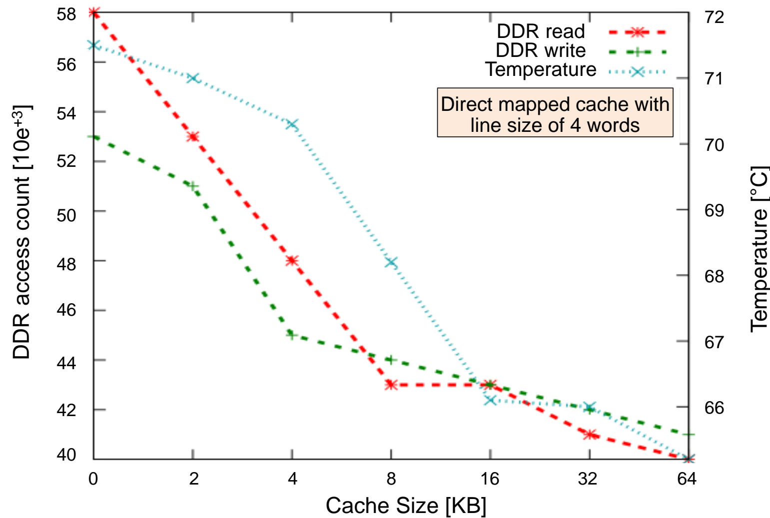


(b) Example of the measured FPGA-based *hard* processor

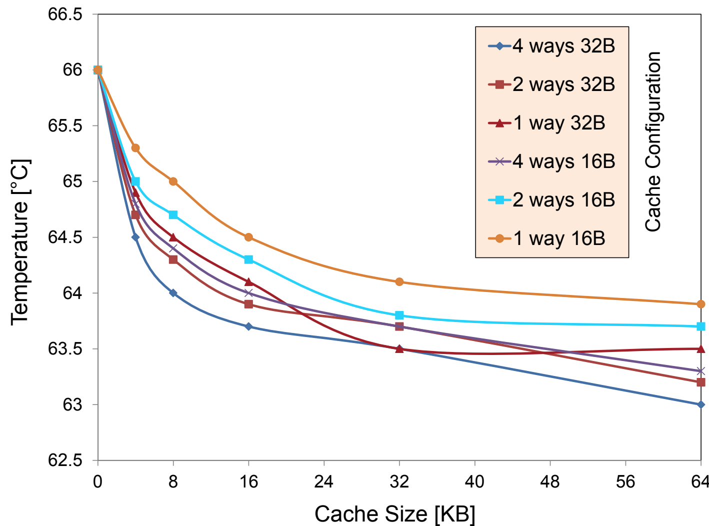
FIGURE 7.9: Infrared images of FPGA-based embedded processors obtained by our thermal setup show that the thermal hot spot is located on the chip's border due to the dominant role of the memory interface on the entire FPGA temperature

that the DDR memory interface has been shown to be the most likely candidate for a thermal hot spot in an FPGA-based embedded systems, we investigate whether a lower miss rate has a positive impact on FPGA die temperature. Another factor that comes into play is cache configuration: the combination of cache associativity, line size, and cache size directly influences cache miss rates depending on the memory access patterns of the application. As such, a large cache with a *non-proper* configuration for a given application will perform worse (i.e., higher miss rate) than a smaller cache with an *proper* configuration. For this reason, we analyze both temperature and cache configuration for different applications in this section.

Figure 7.10(a) shows the temperature results obtained when varying the cache size of a MicroBlaze embedded system running a JPEG encoding application, together with both read and write DDR accesses registered with each cache size. Our evaluation here raised the observation that the temperature of an FPGA-based embedded system strongly correlates with the number of DDR accesses. As a consequence, increasing the cache size can have a positive effect on FPGA temperature as it may result in reducing the need to access the DDR memory due to a higher cache hit rate. This is contrary to existing cache power impact research [111] which usually focuses on ASIC



(a) The influence of cache size of the MicroBlaze processor on the FPGA temperature as well as the DDR memory read/write accesses



(b) The influence of different cache parameters of the LEON3 CPU when running the *ADPCM decoder* application on the peak temperature

FIGURE 7.10: Analyzing the impact of cache component in FPGA-based embedded systems on the chip's temperature

cache implementations. In our case, the low switching density of the FPGA fabric that the cache is built upon combined with the high power consumption of DDR pins means that lower miss rates are always synonymous with lower power consumption and on-chip temperature.

7.2.4 Our Proposed Thermal Cache Model

Adding a memory interface alone adds a base offset in temperature compared to a system where a processor uses purely on-chip memory, largely due to signals such as the DDR clock. While the latter alone had an average temperature of $T_{\text{BASE}} = 55°\text{C}$, adding the

TABLE 7.2: Model parameters in our Xilinx Virtex-5 FPGA platform

α	β	T_{MEM}	T_{BASE}
4.01	0.52	7°C	55°C

Associativities	1, 2, 4
Line sizes (B)	16, 32
Cache sizes (kB)	0, 4, 8, 16, 32, 64

TABLE 7.3: Maximum estimation error between our model and infrared camera measurements for different benchmarks

ADPCM	Matrix Multiplication	qsort	Random	Average
0.53°C	0.64°C	1.2°C	0.2°C	0.64°C

memory interface raises this to $T_{\text{BASE}}+T_{\text{MEM}} = 63°\text{C}$. Afterwards, we observed that the rise in peak temperature can be estimated by the number of accesses per time interval, the access rate r.

$$T = \alpha \cdot e^{-1/\beta r} + T_{\text{MEM}} + T_{\text{BASE}} \tag{7.1}$$

α, β, T_{MEM}, and T_{BASE} can all be obtained experimentally using select temperature profiles obtained as shown in Figure 7.11 and are constant for a particular FPGA platform (assuming that the ambient temperature is constant). It can be noted from Eq 7.1 that the temperature increase for an initial increase in r is more significant than a change where r is already large, mainly due to heat conduction.

In order to model the effect of cache on temperature we have taken measurements using a number of different cache configurations using the LEON3 processor, whose cache is more configurable than the cache available for the MicroBlaze processor and PowerPC-440 that has a permanent cache configuration. An example of the thermal impact of cache configuration of the LEON3 processor on the system's temperature is shown in Figure 7.10(b). As the cache behavior is largely application dependent, we have examined various applications using a cache simulator and taken their cache miss rates as a metric for memory accesses. These results were then stored in $N \times M$ matrices of rates for N different line sizes and M different associativities for each cache size k. The estimated peak temperature is thus given as an extension of Eq 7.1.

$$T_k = \alpha \cdot f(I_k + D_k) + T_{\text{MEM}} + T_{\text{BASE}} \tag{7.2}$$

where f is a function defined as

$$\begin{aligned} f : \mathbf{R}^{N\times M} &\rightarrow \mathbf{R}^{N\times M} \\ x_{n,m} &\rightarrow e^{-1/\beta x_{n,m}} \end{aligned} \tag{7.3}$$

and I and D are the cache miss rate matrices for instruction and data cache, respectively. For $k = 0$, the cache configuration is irrelevant meaning that $\forall i_{n.m} \in I_0, i_{n,m} = r_I$ and

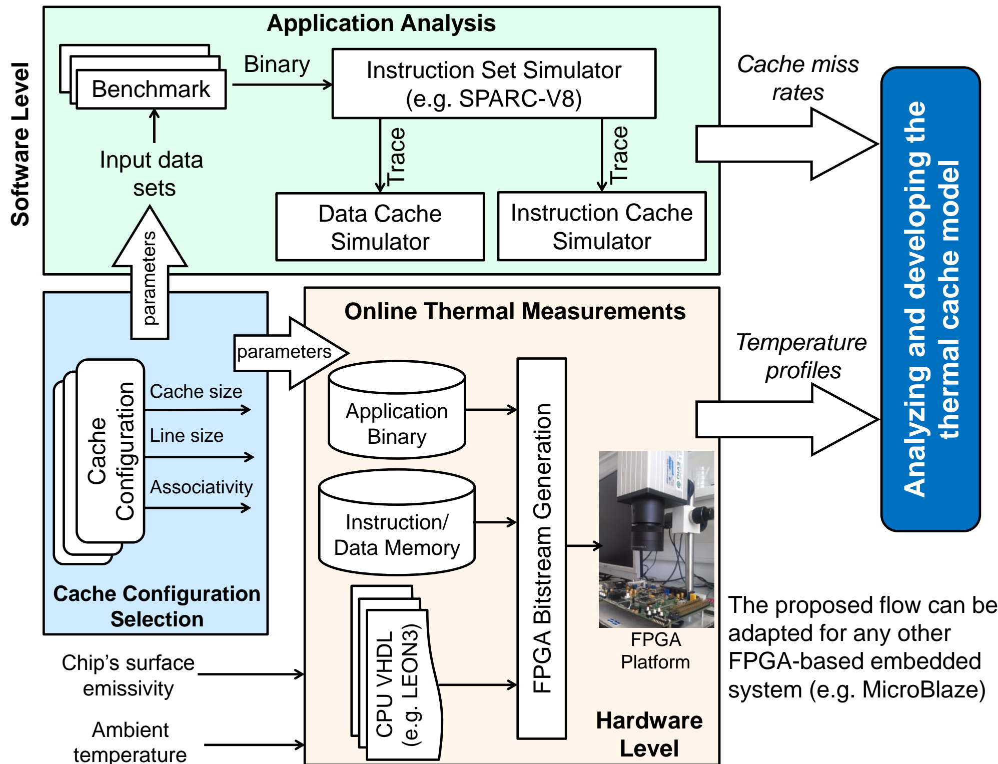


Figure 7.11: Our flow diagram to develop the model linking the cache configuration with the peak system temperature

$\forall d_{n,m} \in D_0, d_{n,m} = r_D$ with r_I and r_D being the maximum memory access rates for instruction and data memory, respectively, of the examined application.

We analyzed our model using various benchmark applications by comparing the peak temperatures estimated by the model with those obtained using the IR thermal camera. In our experiments the parameters were set as shown in Table 7.2, and r was taken as memory accesses per 10^3 cycles. As can be seen in Table 7.3 the maximum estimation error remains low and is within the accuracy bound of the camera ($\pm 1°$C) for all examined applications except for *qsort*. This is partly due to an overestimation of the average r in the instruction memory through various short peaks during application execution, but also due to a slightly higher ambient temperature resulting in a higher T_{BASE}.

The *Random* application is comprised of random accesses to memory that allow us to examine the case where the data cache hit rate remains low regardless of cache size. Since these accesses are performed in a small loop, small instruction cache sizes are already sufficient to prevent cache misses. Due to these reasons, the maximum estimation error of our model for the *Random* application is the lowest, and it is also the application with the smallest range in temperature at the memory interface.

7.3 ASIC-based Multi-Core Systems

As earlier motivated, building such an IR thermography-based measurement setup requires removing the cooling and packaging of the chip to allow the IR emissions to reach the camera's lens. Unlike FPGA-based processors that can still properly operate after exposing its bare silicon, due to their relatively low operating frequency, measuring the temperature of ASIC-based processors presents a more challenging problem as it necessitates building an alternative IR-transparent cooling to allow the IR radiation emitted from the chip to reach the thermal camera and concurrently counteract the very rapid increase in temperature due to the excessive power densities.

An intuitive solution may be to reduce the ambient temperature, as the chip's temperature is almost linearly proportional to the ambient, through putting the entire setup inside a thermal chamber. However, the chip, after removing its cooling, generates heat significantly faster than what the surrounding cold air can dissipate as the latter has a very low thermal conductivity (e.g., 1000x lower than the Aluminum that is often used in heat sinks). To elaborate further, we examined based on the thermal HotSpot simulator [18], the impact of removing the cooling on the Alpha processor temperature[6] and figured out that the peak temperature may increase from $85°C$ (when the cooling is in action) to $> 300°C$ (when both packaging and heat sink are removed). Thus, the ambient temperature needs to be tremendously reduced to prevent the chip from overheating which may, anyway, be not feasible and/or unsustainable for the board and chip. Another important reason to make the solution of putting the entire thermal setup inside a thermal chamber not suitable for IR measurements is the condensation phenomenon due to the conversion of water from air into a liquid, i.e. when the warm air rises from the hot chip and then cools down due to the very cold surrounding air. Such water drops on top of the measured chip, which can be quickly generated, reduce the lucidity of the captured images as well as the accuracy of the conducted measurements because water does not allow the emitted IR radiations from the chip to go through towards the camera due to its low IR transparency. Additionally, the generated water may be harmful for the board and chip.

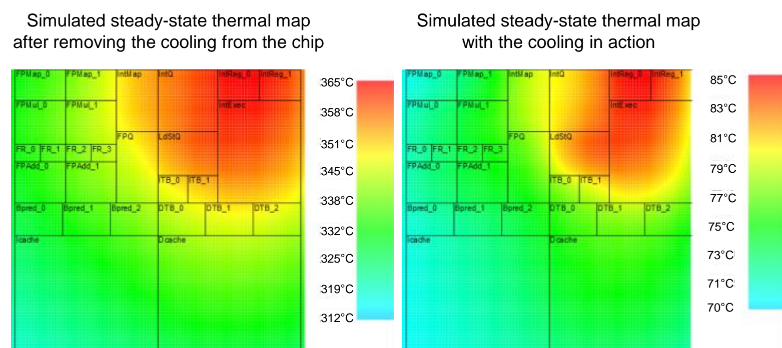


FIGURE 7.12: Simulated steady-state thermal maps through the thermal HotSpot tool [18] to demonstrate the potential temperature increase due to removing the cooling (heat sink and packaging) from the chip

[6]The required steps to conduct these simulations have been earlier presented in Section 6.4.1.1.

7.3.1 State-of-the-art Liquid-based Setups for IR Thermography

As motivated in Section 2.4.2, state-of-the-art in IR thermography of ASIC-based processors deploys a liquid-based setup to protect the chip from damage during conducting the required thermal measurements. In the following, we discuss how such a setup can be built along with its main drawbacks with respect to the quality of captured thermal images.

Building an IR-transparent heat sink using a coolant liquid flowing directly on top of the bare silicon of the chip necessitates choosing a liquid that must be IR transparent to enable the IR emissions to obtrusively reach the camera. An example of the used liquid for this purpose is a mineral oil [102, 112] specifically designed for IR spectroscopy. By continuously circulating the coolant oil, the generated heat from the processor chip can be dissipated and hence protect it from overheating. An external pump is then used for the purpose of controlling the flow speed. To constraint the flow, a window on top of it is also needed. Such a window must be an IR window (e.g. sapphire window, calcium fluoride window etc.) to allow energy at a specific electromagnetic wavelength to pass. Figure 7.13 illustrates the liquid-based thermal measurement setup with its required pieces showing how additional layers may interfere with the emitted IR radiation.

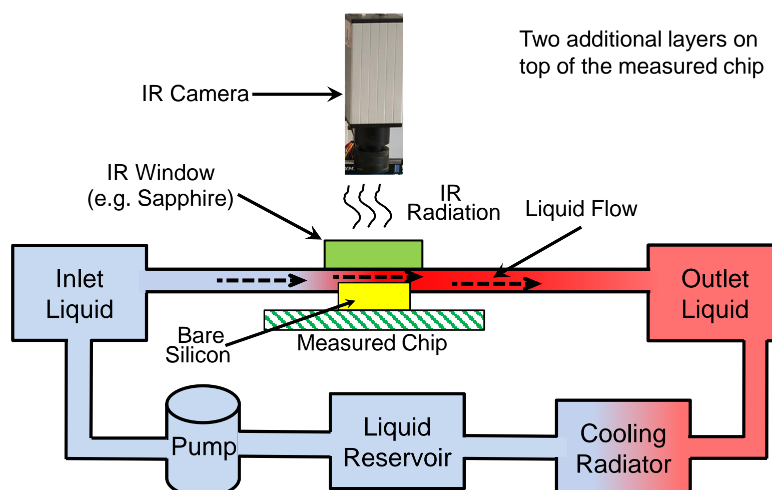


FIGURE 7.13: Liquid-based thermal measurement setup applying an IR coolant oil on top of the measured chip to prevent overheating

7.3.2 Obstacles behind Capturing Lucid IR Images

The key drawback of the aforementioned setup is that several aspects interfere with the measured IR radiation resulting in equivocal (i.e. non lucid) IR images.

***Compatibility*:** The deployed materials to build the setup parts have diverse IR spectroscopy properties that complicates compatibility. As a matter of fact, each IR camera operates at a specific wavelength that covers a corresponding spectrum of electromagnetic radiation. For instance, short wavelength IR cameras cover a wavelength of 1-3μm and long wave IR cameras cover a wavelength of 8-14μm [157]. The transmission range

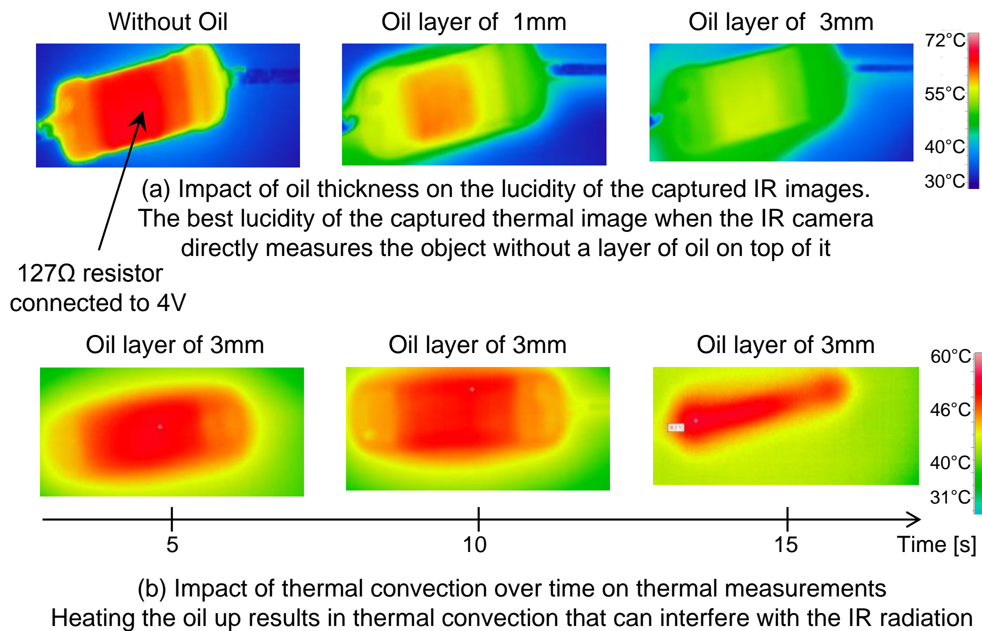


Figure 7.14: Thermal measurement experiments demonstrating the impact of mineral oil applied to a hot object

indicates to energy in the region of the electromagnetic radiation spectrum. When new layers such as the coolant liquid and the IR window are added on top of the measured chip, their transmission ranges need to be carefully chosen to maintain compatibility. A sapphire IR window, for instance, has a transmission range of 0.17 to 5.5μm [157] which makes it compatible only with short wavelength cameras whereas long wavelength cameras cannot collect IR data through it because a sapphire window does not transmit beyond 5.5μm [157]. Similarly, the transmission range of the selected oil needs to be compatible with the transmission ranges of the used IR window and camera.

***Thermal Convection*:** The IR camera captures the chip's thermal image by capturing the electromagnetic waves that radiate from the chip and directly transport IR radiation through air. When a coolant liquid is applied to the surface of the hot chip, the thermal convection phenomenon starts to appear due to the transfer of heat from one region to another by the movement of liquid. This diminishes the thermal images lucidity of the measured object as it can be seen in Figure 7.14. There, IR mineral oil that is used in many current thermal setups [46, 102, 103] has been tested.

***Complexity*:** The thickness of the applied oil layer plays a non-negligible role in determining the lucidity of the captured IR images because it exacerbates the occurred thermal convection phenomenon. This can be seen in Figure 7.14 (a) where thicker oil layer made the image less lucid. Furthermore, the properties of the applied oil are additional sources that increase the setup complexity such as its viscosity and flow speed, which influence the thermal convection. It is also necessary to keep the liquid free from pollutants over time that compromise its transparency.

In Summary: State-of-the-art in IR thermography is challenging to build as it is subject to multiple aspects that negatively interfere with the emitted IR radiation. That, in turn, negatively influences the lucidity of the captured images. *Therefore, we propose an IR-transparent cooling technique that simplifies the construction of the setup as well as enhances its quality by avoiding any additional layer between the chip and camera.*

7.3.3 Our Proposed RAMA Technique for Lucid IR Images of Multi-Cores Processors

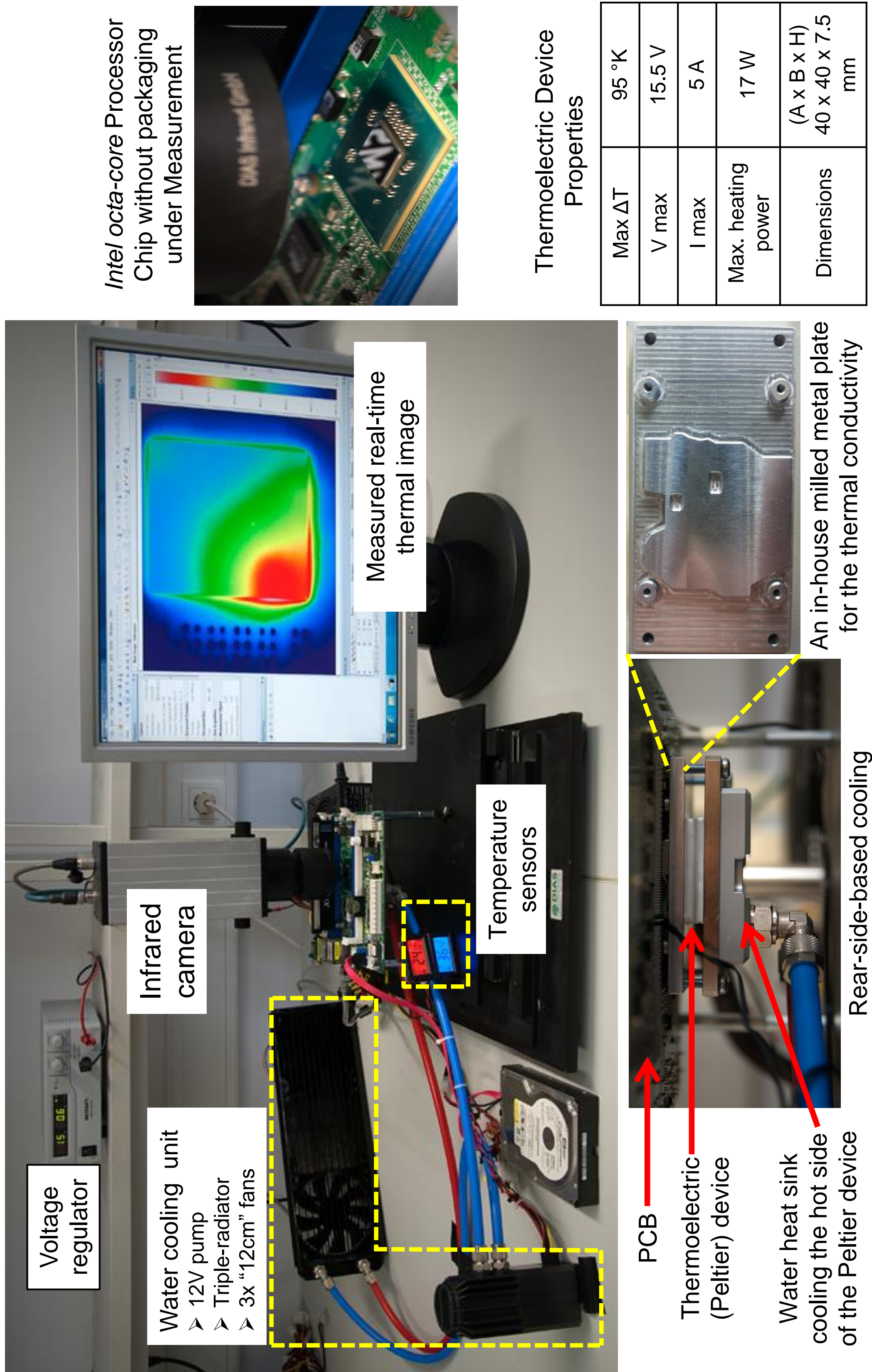
To keep the measured chip in operational conditions after removing its cooling unit, we continuously chill it from the rear side, i.e. through the PCB to which the chip is attached as shown in Figure 7.16. Thermoelectric device is employed as it is capable of providing a stable and controlled source of cooling. The basic concept is the *Peltier effect*, which is a phenomenon that occurs when electrical current flows through two dissimilar semiconductor materials (i.e. N-type and P-type) [158], resulting in a thermal variance.

It heats up one of the thermoelectric sides of the device and chills the other due to conduction of heat from one side where it is absorbed to the other side where the heat is released. The generated heat must be continuously dissipated from the thermoelectric device hot side in order to avoid overheating and to allow for a continued proper operation. Therefore, we use in our setup a water-cooling unit to quickly dissipate the heat from the thermoelectric device's hot side. The temperature difference between both thermoelectric device sides can be controlled by regulating the applied voltage. Our built thermal measurement setup is depicted in Figure 7.15. We investigate chips with flip-chip technology [102], as it allows mechanical removal of the packaging as opposed to e.g., a wire-bond chip. This means that the silicon wafer of the processor will be exposed after removing the cooling unit and chip packaging. On-chip temperatures are then directly measured using a DIAS pyroview 380L compact IR camera capable of capturing temperatures with an accuracy of $\pm 1°C$ and a spatial resolution of 50 μm per pixel [144]. Once the readings have been obtained, the camera sends them at a frame rate of 50Hz to a PC that analyzes them to build the corresponding thermal image of the measured chip. Actually, we employed the same the IR camera, that has been utilized in our thermal analysis of FPGA-based on-chip system (see Section 7.2.2), and the *emissivity* aspect has been similarly tackled (see 7.2.1.1).

It is noteworthy that our setup is not restricted to a specific kind of IR cameras, unlike state-of-the-art that necessitates employing an IR camera operating at a specific wavelength which must be compatible with both IR oil and IR window.

7.3.4 Evaluation, Comparison and Advantages

To evaluate our novel thermal measurement setup, we use a 45nm dual-core processor from Intel along with the *CPUBurn* benchmark [21]. The selected benchmark is designed to load x86 CPUs as heavily as possible for the purposes of maximizing heat



Max ΔT	95 °K
V max	15.5 V
I max	5 A
Max. heating power	17 W
Dimensions	(A x B x H) 40 x 40 x 7.5 mm

FIGURE 7.15: Our proposed RAMA technique for lucid IR thermography of processors

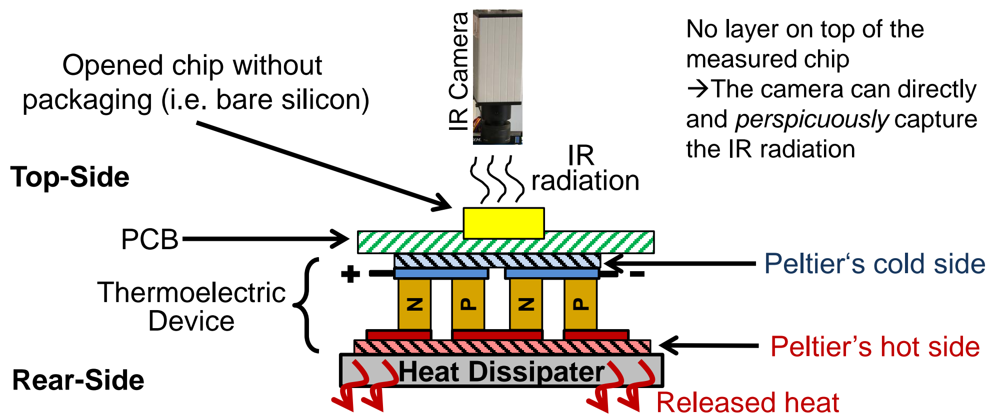


FIGURE 7.16: The employment of a thermoelectric device in our RAMA setup to avoid adding any layer on top of the chip

production from the CPU and putting an intensive stress on the chip itself as well as the cooling unit [21]. Towards evaluating our setup under the highest possible stress, we operate the two cores at the maximum frequency (1.8 GHz) with hyper-threading enabled. As a result, four *CPUBurn* programs are simultaneously run which maximizes the workload on the entire chip. The steady-state temperature of the chip, under such an intense scenario, reaches around $55°C$ which is far from the critical temperature ($80°C$ as Intel defines). This shows that our setup protects the chip from damage during IR measurements that are required for performing the desired thermal analysis. Some of the captured IR images are shown in Figure 7.17 which demonstrates the real-time chip thermal images at different points of time from the start of program execution.

We feed the thermoelectric device with the appropriate voltage to make our cooling setup mimics the behavior or the regular/original one (i.e. the heatsink-based cooling). However, tuning the applied voltage can provide us with a wide range of applied cooling to select the most suitable case for the targeted measured chip, i.e. corresponding to its regular/original cooling unit.

7.3.4.1 Steady-State Temperature Equivalent

Since different cooling may have different thermal impact on the chip, it is necessary to study in how far our proposed cooling setup is representative with respect to the original setup, i.e. with the fabricated cooling unit. Therefore, we directly compare. For a fair comparison, both processors start at the same initial temperature (i.e. $30°C$). Then, we concurrently run the same intense workload on both[7] for the same interval of time – around 5 minutes which is sufficient to reach the steady-state temperature. During operation, the thermal-diode sensor readings of each core are recorded to compare the thermal impact of our setup with the original cooling. As presented in Figure 7.17, the

[7]The aforementioned intensive scenario has been employed here, i.e. running simultaneously four *CPUBurn* programs

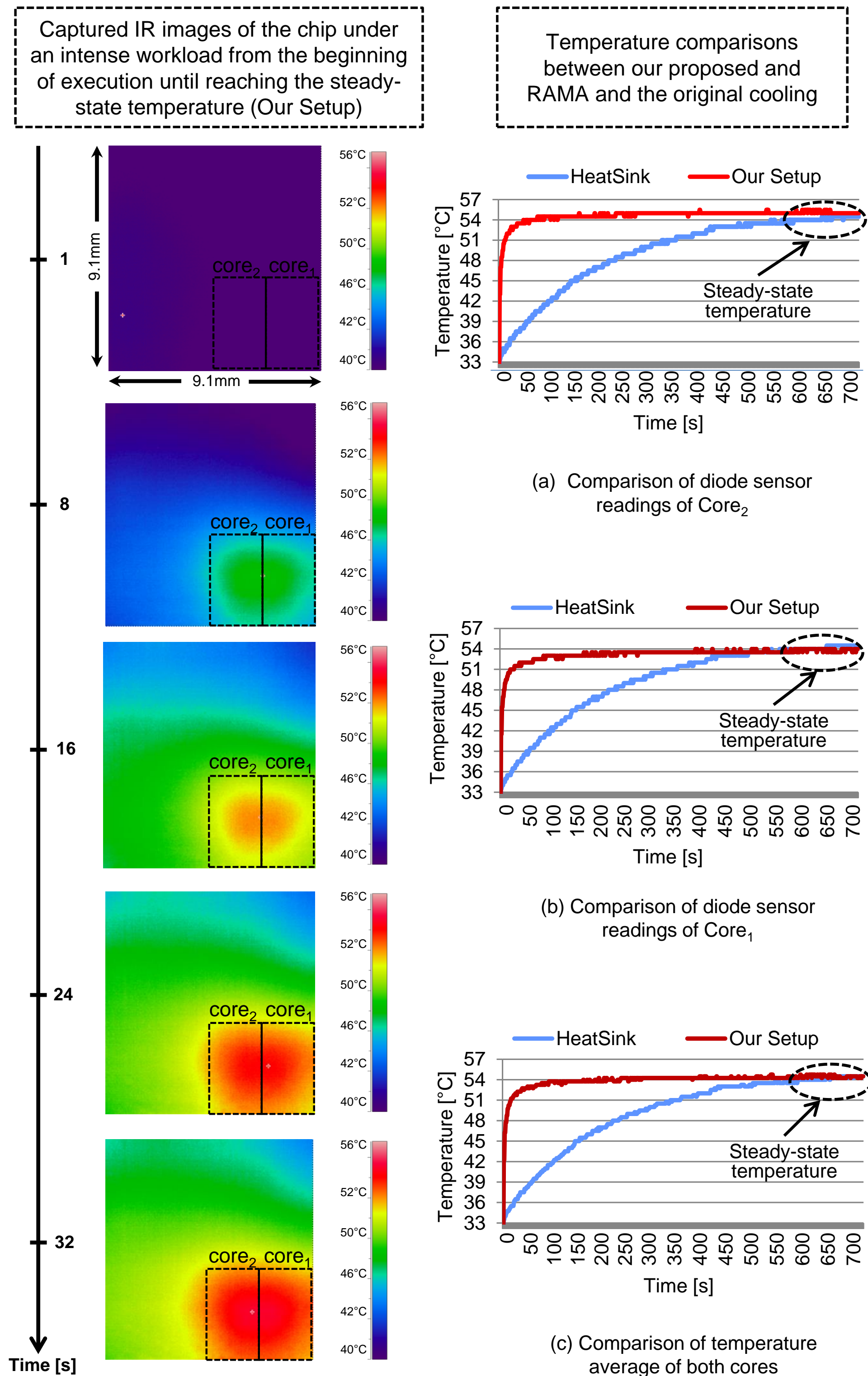


FIGURE 7.17: Measured IR images of run-time behavior of the tested processor under an intense stress (left) with the temperatures of individual cores when our proposed cooling technique is applied compared to those when the original heat sink-based cooling is utilized (right) showing that our RAMA technique is representative of the original cooling

steady-state temperature of each core with our cooling setup in action is very similar to the steady-state temperature in case of the original cooling unit[8].

Under identical conditions with respect to initial temperature, steady-state temperature and workload, our introduced IR-transparent cooling setup for thermal measurements achieves a very good representation of the original cooling unit in terms of steady-state temperatures.

Unlike FPGA-based processors, where the thermal hot spot is located at the memory interface as we demonstrated earlier in the Section 7.2.2, we observe in ASIC-based processors that the thermal hot spot is caused by the processor cores themselves, as shown in Figure 7.17. This is mainly due to the significantly higher power densities of cores. Therefore, thermal management needs to focus on the cores' temperatures. For instance, task migration-based thermal management techniques (e.g., [159]) may be applied. In the following, we investigate the thermal characteristics when tasks are migrated between cores.

For a more general evaluation, we employ a state-of-the-art Intel 22nm *octa-core* processor running at a maximum frequency of 2.4 GHz. The scenario of running the intense workload (i.e. eight *CPUBurn* programs simultaneously running[9]) has been first examined and it showed that our setup is able to protect chip from damage during measurement under such an intense stress (the peak temperature was around $75°C$). Figure 7.19 presents some of the measured IR images of the chip and the corresponding calculated spatial thermal gradient maps for the scenario of migrating the running tasks from two cores to others every 10 sec. The thermal video of this experiment is available here: (`https://db.tt/bD3HJIji`) to enable the reader to evaluate the dynamics that can otherwise not be presented in a written thesis. As shown in Figure 7.19, different scenarios exhibit different characteristics and the maximum spatial thermal gradient can reach up to 15°C/mm. The gradient maps have been calculated according to the formula within Figure 7.18 that provides an integration in order to prohibit wrong gradients caused by the pixel resolution of the thermal camera [132]. This is necessary to avoid the noise in pixel values during the computation of pixelwise difference.

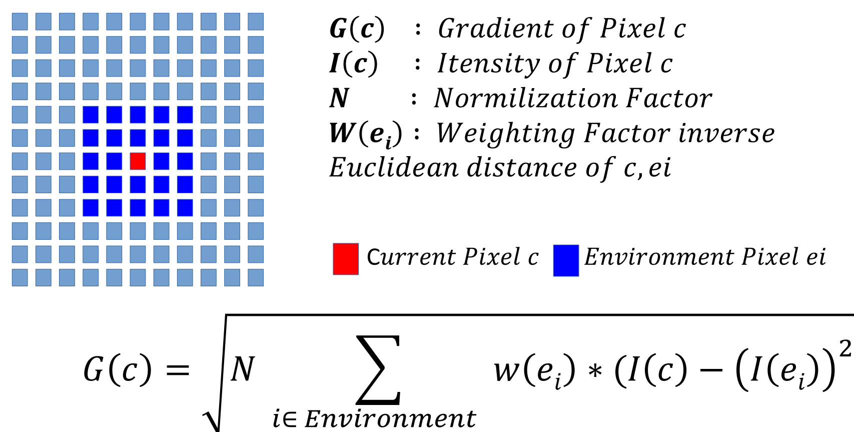
The operating conditions of the employed thermoelectric device in these experiments ($I_{in} = 0.6\,\mathrm{A}$, $V_{in} = 1.5\,\mathrm{V}$) were carefully selected to make our setup mimic the impact of the original cooling. As it can be noticed from the specifications in Figure 7.15, the thermoelectric device is not maxing out in its input current and it still has more current (around 8.3x) to use. This enables providing a wide range of cooling to select the most suitable one for the targeted chip, e.g. providing a higher cooling to examine more severe scenarios, that might be generated in other processors with higher power densities.

[8]An online adaptation of the applied voltage to the thermoelectric device can at runtime adjust our cooling setup to also mimic the behavior of the transient temperature of processor cores.

[9]The hyper-threading is not available in this processor.

7.3.4.2 Jeopardy of Equivocal IR Images

In Figure 7.20, we clarify the loss in the IR images lucidity due to the addition of a thin layer of IR oil on top of the chip. Due to the transfer of heat from one region to another by the movement of liquid, thermal convection interferes with the IR radiation and results in equivocal IR images. Whereas, the IR image captured by our setup is lucid, as the camera perspicuously captures the IR radiation. Importantly, an equivocal IR image negatively impacts the spatial thermal gradient analysis as the thermal contour maps[10] and histograms establish. As shown, the oil causes the spatial thermal gradient analysis to appear different from the case of doing the measurements without oil. This leads to ambiguous vision for designers regarding the potential spatial thermal gradients that may occur during operation. In this particular scenario within Figure 7.20, we measure a 7°C variation in terms of the maximum gradient. However, this may be higher when stressing more cores as the latter further heats up the oil.



$$G(c) = \sqrt{N \sum_{i \in Environment} w(e_i) * (I(c) - (I(e_i))^2}$$

FIGURE 7.18: Spatial thermal gradient calculation

7.3.4.3 Impact of Inaccurate Thermal Analysis on Reliability

Spatial thermal gradients result in a temperature variation within a small distance. This causes identical circuits within a component (e.g., SRAM cells of CPU cache) have distinct characteristics, which degrades the entire reliability. One of the key reasons behind deploying thermal setups is enabling designers to accurately estimate the reliability requirements of their systems. In the following, we investigate for the first time under high accurate gradients maps extracted from lucid thermal images how such a variation of 7°C in estimating the maximum spatial thermal gradient – that comes from applying the oil during measurements– significantly impacts the reliability analysis.

To this end, 22 nm SRAMs operating at $V_{dd} = 0.8\,\text{V}$ have been studied through our proposed reliability estimation in Chapter 3. SRAMs have been employed as a representative example due to their total chip area that may reach up to 70% [123]. The three

[10]Thermal contour is a means to represent the thermal gradients where the absolute temperature difference between two lines is the same

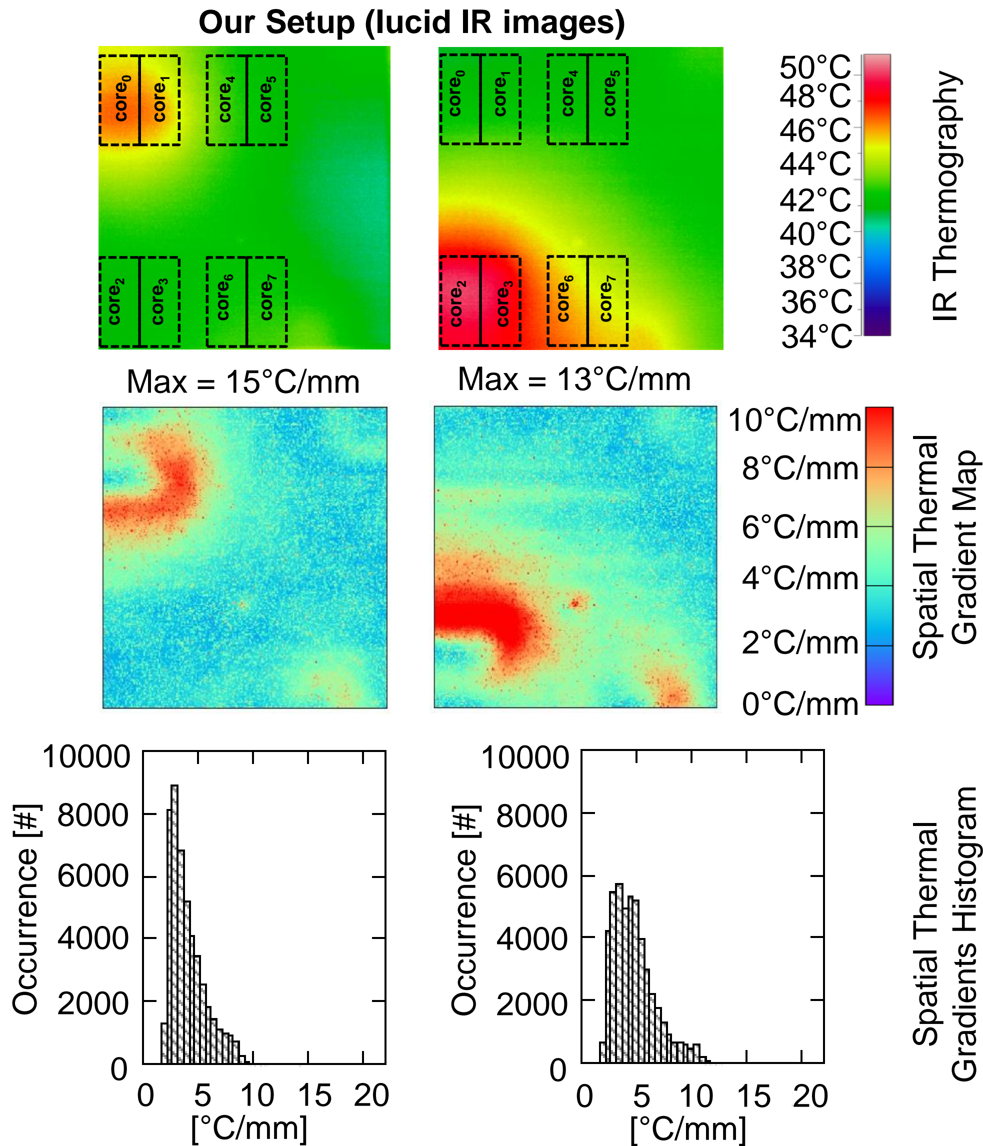


FIGURE 7.19: Measured IR images along with the corresponding spatial thermal gradient maps of Intel 22 nm *octa-core* chip under different scenarios of running an intense workload on two particular cores

key reliability metrics of SRAM cells (SNM, RAT, Q_{crit}) have been in this analysis explored. Details regarding these metrics have been earlier presented in Section 1.2.

As shown in Figure 7.22(a), the temperature rise shifts the SNM distribution to the left resulting in lower resiliency against noise. A similar behavior can also be observed in Figure 7.22(b) which shows how a temperature increase noticeably makes the SRAMs more susceptible to radiation. In Figure 7.22(c), the temperature rise shifts the RAT distribution to the right making SRAMs slower.

As it can be noticed, a thermal variation of $7°C$ increases the RAT by 4%, on average, and thus there is a need for operating the SRAMs at 4% lower frequency to avoid time

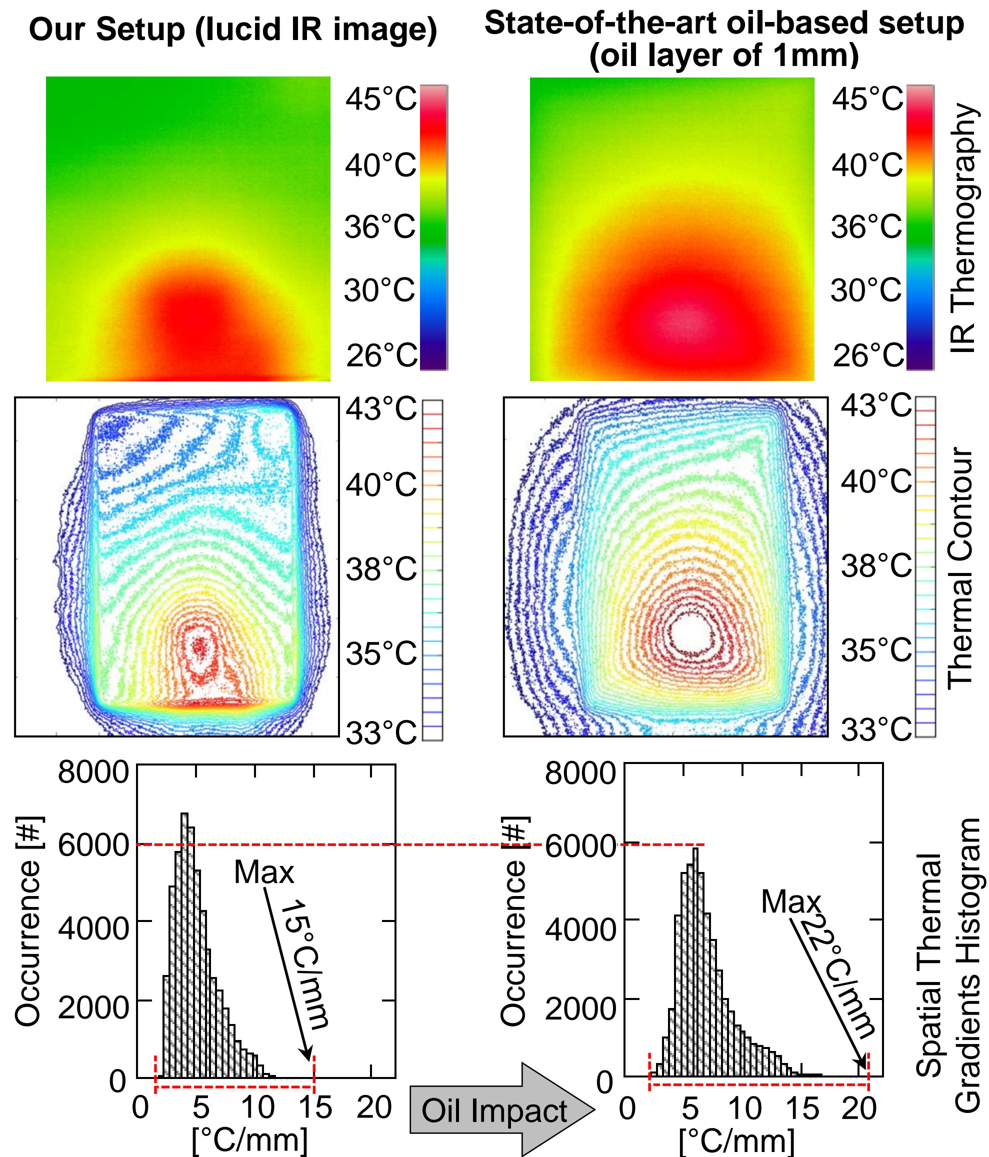


FIGURE 7.20: Adding a layer of oil on top of the measured chip losses the captured IR images its *lucidity*, due to the thermal convection, which jeopardizes the spatial thermal gradient analysis

violations. In other words, estimating inaccurately the potential spatial thermal gradients, that might be generated during operation, would make some SRAMs not meet their timing constraints.

When aging analysis comes into play, the aforementioned problem of incorrectly estimating reliability is exacerbated. In Figure 7.21, we illustrate the jeopardy of a 7°C mis-measurement on overestimating the impact that aging effects may have on the transistors reliability (i.e. aging-induced threshold voltage increase (ΔV_{TH})) as well as the SRAMs reliability. We employed in this analysis our presented reliability estimation in Chapter 3 that takes both BTI and HCID into account.

As shown, inaccurate thermal analysis during testing may lead to overestimate the impact of aging on, for instance, *RAT* by 16% and thus the chip will be later sold with

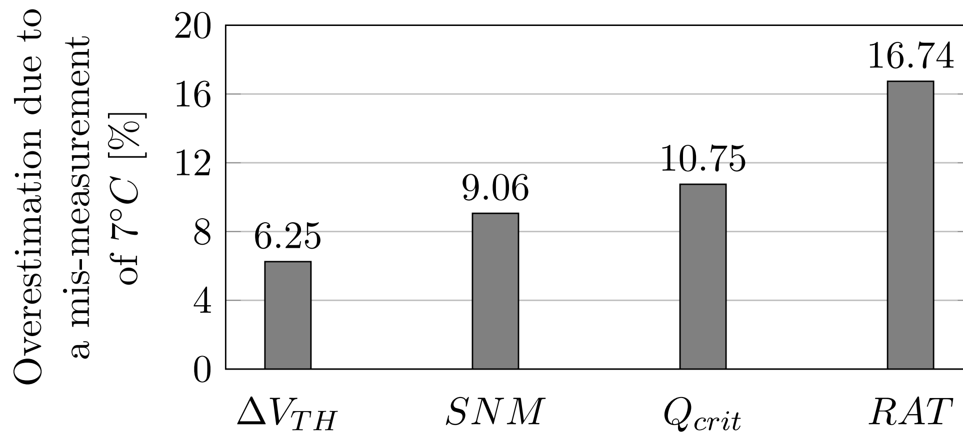


FIGURE 7.21: Impact of state-of-the-art inaccurate thermal analysis on incorrectly estimating aging for a lifetime of 10 years

a 16% lower frequency than what is actually needed to avoid time violations that will not, anyway, occur during runtime.

All in all, having lucid thermal images allows designers to accurately explore the thermal characteristics of on-chip systems (i.e. peak on-chip temperatures, spatial thermal gradients, etc.) and thus correctly estimating their reliability.

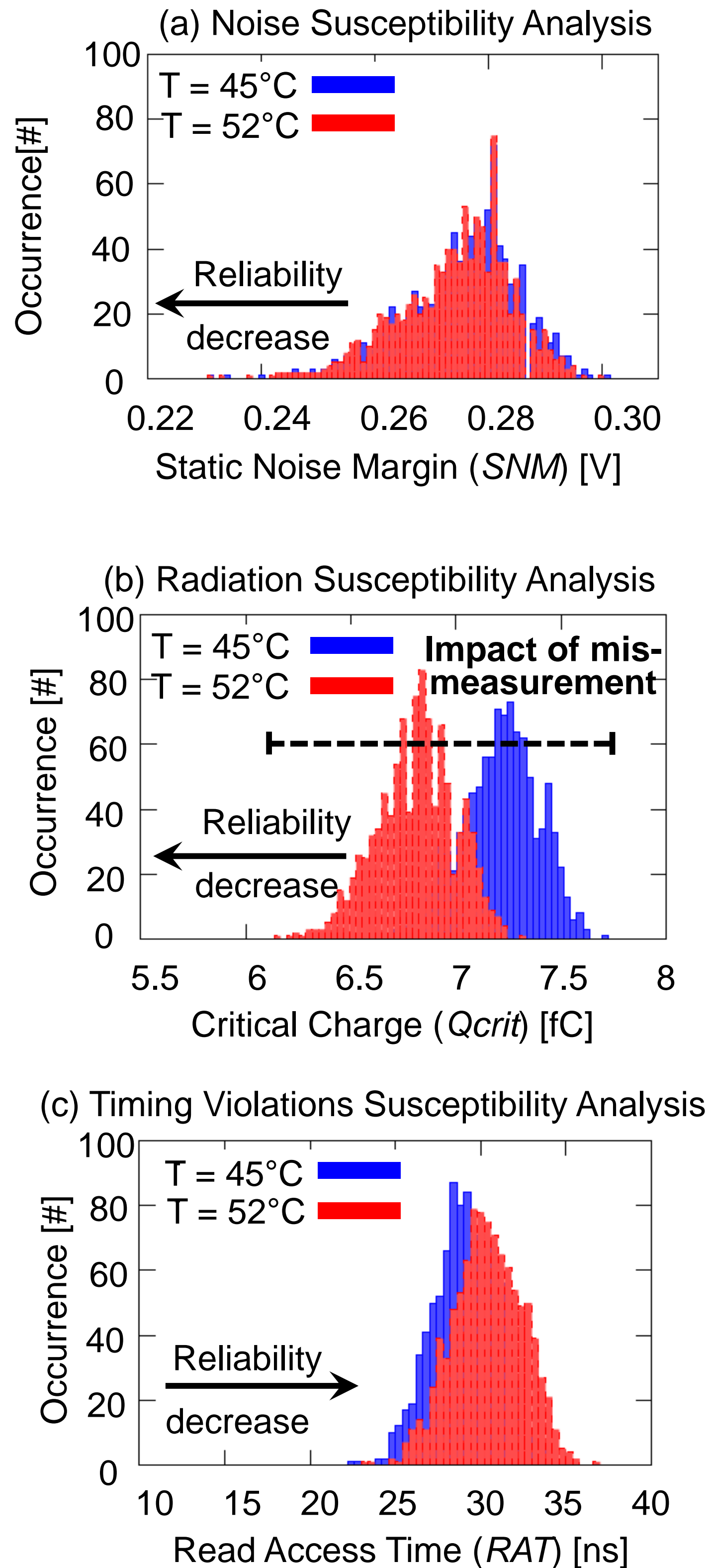


FIGURE 7.22: Impact of a thermal mis-measurement (as the case for current state-of-the-art thermal setups) of *7°C* on the key reliability aspects of 22nm SRAMs and how it noticeably alters the SRAMs characteristics (i.e. their resiliency against failures). Such a variation comes as a result from analyzing the spatial thermal gradient based on IR images captured *when a thin layer of oil (1mm) is applied on top of the chip* during measurements.

7.4 Summary of Our Thermal Investigation

Technology scaling has made temperature concerns one of the fundamental challenges that faces designers due to their negative impact on reliability. Towards understanding the thermal characteristics in modern chips, real-time thermal measurements are substantially needed to provide an accurate thermal analysis. Our thermal setup employing an IR camera enabled us to quantify the stability and error of conventional thermal analysis methods. We demonstrated that the on-chip thermal diode sensor is not sufficient in studying the chip temperature and the thermal difference between its reading and the peak FPGA temperature can reach up to 20°C. We also showed that the dominant part of thermal hot spots comes from the memory interface when targeting FPGA-based embedded processors. Based on our observation that different cache configurations result in different memory access rates affecting the thermal behavior, we proposed a thermal cache model linking cache miss rates with FPGA die temperature. This model exhibits an average maximum error of 0.64°C. Additionally, we introduced a novel methodology of building an IR transparent cooling system enabling thermal cameras to directly capture thermal images of ASIC-based multicore processors. The proposed RAMA technique overcomes the drawbacks of the state-of-the-art liquid-based cooling (i.e. compatibility and complexity). Our evaluation showed that the proposed setup is capable to keep the chip's temperature in a safe range under intense computational stress scenarios. Finally, our analysis established that the proposed setup can also be equivalent in terms of steady-state temperature to the original heat sink-based cooling.

Chapter 8

Conclusion and Outlook

8.1 Thesis Summary

The relentless technological pressures on the semiconductor industries to massively increase the number of transistors per chip caused so-called of nano-CMOS era, where transistor feature sizes are in the nanometer scale. In spite of the superior improvements in multiple facets such as performance that the nano-CMOS era brought, the reliability of on-chip systems has been seriously jeopardized. Thus, Technology Roadmaps have predicted that conventional constraints such as cost may be overtaken by reliability concerns [35]. Aging effects and soft errors, that were until recently not critical for the everyday embedded on-chip systems, became at the forefront of these concerns due to their momentous ability to cause failures in the underlying hardware. As a result, the susceptibility of current and upcoming embedded on-chip systems to such threats has been magnified. Unfortunately, this runs counter to the growing need of most state-of-the-art on-chip systems where maintaining reliability during the entire lifetime cannot be easily scarified. As a matter of fact, on-chip temperatures play an essential role in intensifying the susceptibility of circuits to a wide range of reliability degradations, as was discussed in Chapter 1. Therefore, accurately investigating the thermal characteristics of on-chip systems is substantial to explore the deleterious effects that the generated temperatures at runtime may have in the short as well as long terms and thus accurate thermal analysis is prerequisite to design reliable on-chip systems.

Hence, this thesis presents different techniques to increase the reliability of embedded on-chip systems with respect to aging effects, soft errors and temperature-dependent behavior. Our techniques tackle different abstraction levels: from the device/circuit (Chapter 3) to multi-cores (Chapter 7) level through the microarchitectural (Chapters 4 and 5) level. Evaluations of the proposed techniques were conducted through simulations and partially through implementations on hardware platforms.

First, we developed a new reliability estimation that presents one step by combining the simultaneous effects of multiple aging mechanisms examining their interdependencies from the physical level, in order to derive a probability of failure as a meaningful reliability abstraction, yet sufficiently accurate for system-level designers enabling them

to compromise between cost of the chip and its reliability and to additionally expose the most susceptible parts to aging in their systems. Afterwards, the developed reliability estimation has been employed to investigate the reliability of register file within an embedded on-chip system under *bare-metal workloads* as well as *conforming workloads*. Akin to the computational intensity that comes from targeting *conforming workloads*, the latter imposed to not employ conventional software/simulation-based techniques. For that purpose, a novel real-time hardware-based technique has been implemented in an FPGA platform enabling us to explore the impact that several parallel applications along with an operating system may have on aging effects. This, in turn, makes connecting the system level (where applications are run) and the physical level (where aging effects are originated) feasible, towards grasping how workloads *drive* aging.

Our proposed RISB technique increases the reliability of SRAM-based register file components of microprocessors with respect to aging effects and it employs our developed reliability estimation to evaluate the effectiveness. Unlike current techniques aiming to increase the resiliency of SRAM cells within register files against aging effects, where all registers are tackled in the same manner, RISB proposes a selective strategy to consider that different access patterns of individual registers have an important affect on the effectiveness of the aging mitigation technique.

For embedded systems under tight constraints, where area, performance, power and reliability cannot be easily compromised, our RESI technique tackles the challenge of increasing the register file reliability with respect to soft errors (both SBUs and MBUs) along with a light impact on overheads. It additionally mitigates the aging effects within the SRAM cells of the register file. The experimental results demonstrated that RESI reduces the register file vulnerability against soft errors and it achieves a high system fault coverage under different scenarios.

The reason behind the focus on the register file microarchitectural component when increasing the reliability of embedded on-chip systems is due to its high utilization (i.e. it is very repeatedly accessed) which makes failures due to aging effects or soft errors may rapidly spread from there throughout the entire computational system. Thus, the register file is recognized as one of the critical components when it comes to reliability.

On the other hand, for circumventing the lack of information of thermal characteristics in modern processors, real-time thermal measurements are intrinsically needed. Therefore, we present RAMA which is a novel technique for IR-transparent cooling enabling thermal cameras to perspicuously capture the chip's thermal images. RAMA overcomes the drawbacks of the state-of-the-art liquid-based cooling in order to provide lucid IR thermography of multi-cores on-chip systems. In addition to the right thermal setup, we show for the first time in how far critical circuit characteristics for reliability are correlated with temperature. If the thermal setup would not be as accurate as the one presented within this thesis, reliability would be incorrectly estimated.
In all aspects, it is shown that capturing lucid thermal images has a significant impact on accurately estimating reliability.

8.2 Current Limitations and Future Work

Another key reliability challenge due to the smallness of the transistor feature sizes has already been reported to be a severe issue, namely Random Telegraph Noise (RTN) [32]. It is expected to be one of the major difficulties to maintain reliable operations in scaled SRAMs due to its strong ability to induce random V_{TH} fluctuations. Additionally, technology scaling has exceeded the tolerances of equipment used to manufacture semiconductor circuits [160], resulting in ever increasing fluctuation in the dopant concentrations which, in turn, leads to further fluctuations in the V_{TH} of transistors. RTN and Random Dopant Fluctuation (RDF) besides aging phenomena (e.g., NBTI, PBTI and HCID) will make reliability become unaffordably expensive due to the inevitable need for wider and wider safety margins. This will open the door for developing new hardware/software co-design techniques towards acquiring reliable embedded on-chip systems in the scope of the tight cost constraints (e.g., performance, power, etc.).

On the other hand, an important research that can be based upon the presented reliability estimation within this thesis, is to broaden the scope of analyzing the SRAM cells reliability through studying other kinds of structures/designs besides the typical 6-T SRAM, that has been targeted in this work, such as 8-T SRAM cells and others.

Finally, supporting circuits of SRAM cells such as sense amplifiers may also suffer from aging effects which, in turn, contribute to additional degradations in the reliability of the entire SRAM cell as the likelihood, that the SRAM cell unreliably operates, increases (i.e. when the supporting circuits age, the SRAM cell becomes slower and/or less resilience to noise resulting in failures due to timing violations and/or data corruption, respectively). Therefore, investigating the aging effects in both SRAM cell as well as its supporting circuits through our proposed reliability estimation in Chapter 3 provides the designers with a more clear version of how SRAM-based components (e.g., register files, caches, etc.) within embedded on-chip systems may fail during the lifetime.

Chapter 9

Appendixes

9.1 Aging Analysis

9.1.1 Parameters of Equations

Symbol	Description
$\boldsymbol{\lambda}$	SRAM *duty cycle*
$\boldsymbol{\Lambda}$	transistor *duty cycle*
$\boldsymbol{\lambda_e}$	mean free path of hot electrons
$\boldsymbol{\Phi_{\mathrm{it},e}}$	critical electron energy interface trap creation
$\boldsymbol{\tau}$	stress time
$\boldsymbol{\tau_r}$	recovery time
$\boldsymbol{C_{\mathrm{ox}}}$	oxide capacitance
$\boldsymbol{E_{\mathrm{m}}}$	channel electric field
$\boldsymbol{E_{\mathrm{AIT}}}$, $\boldsymbol{E_{\mathrm{AHT}}}$, $\boldsymbol{E_{\mathrm{AOT}}}$	activation energy of interface/hole/ oxide traps
$\boldsymbol{E_{\mathrm{ADH2}}}$	activation energy of H_2 diffusion
$\boldsymbol{E_{\mathrm{Akf}}}$, $\boldsymbol{E_{\mathrm{Akr}}}$	activation energy bond dissociation/ bond annealing
$\boldsymbol{E_{\mathrm{cr}}}$	critical field for velocity saturation
$\boldsymbol{N_{\mathrm{IT}}}$, $\boldsymbol{N_{\mathrm{HT}}}$, $\boldsymbol{N_{\mathrm{OT}}}$	number of interface/hole/oxide traps
$\boldsymbol{t}$	current time
$\boldsymbol{V_{\mathrm{DSAT}}}$	potential at channel pinchoff
$\boldsymbol{x_j}$	source/drain junction depth
$\boldsymbol{t_{ox}}$	oxide thickness
$\boldsymbol{\epsilon_{Si}}$, $\boldsymbol{\epsilon_{SiO_2}}$	relative permittivity of Si and SiO_2
$\boldsymbol{n_1}$, $\boldsymbol{n_2}$	BTI/HCID time exponent
ABC, **DE**, $\boldsymbol{\alpha\eta}$	fitting parameters
$\boldsymbol{\Gamma_{\mathrm{IT}}}$, $\boldsymbol{\Gamma_{\mathrm{HT}}}$, $\boldsymbol{\Gamma_{\mathrm{OT}}}$	field acceleration factors
$\boldsymbol{\beta_{\mathrm{OT}}}$, $\boldsymbol{\beta_{\mathrm{HT}}}$	hole trapping parameter
$\boldsymbol{\beta_{\mathrm{HTR}}}$	hole trap recovery parameter
$\boldsymbol{\mu_0}$	carrier mobility at t=0

9.1.2 Material Structures

Crystalline materials have regular structures unlike the amorphous materials as Figure 9.1, taken from [19], illustrates. Some SiO_2 rings are bigger and some of them are smaller compared to the crystalline structure. Such variations may make breaking some bonds easier than others due to the less required activation energy.

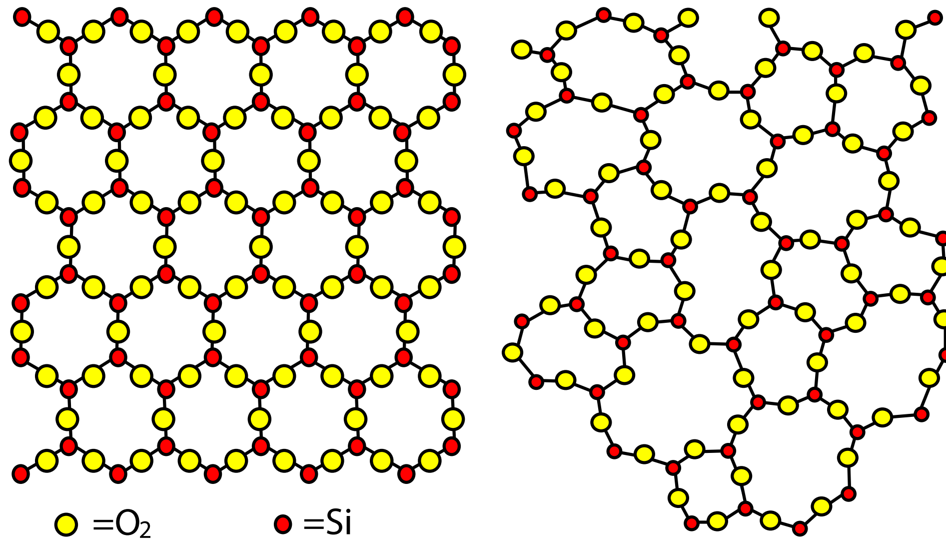
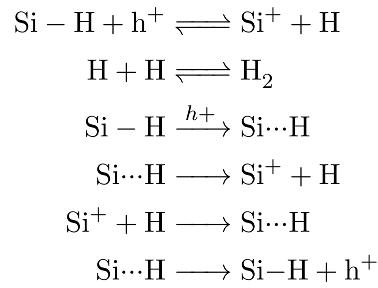


FIGURE 9.1: Crystalline and amorphous structure of the SiO_2. The figures are from [19]

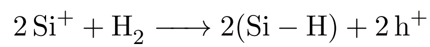
9.1.3 Breaking and Healing Si−H Bonds

The described interface traps generation in Section 3.3.1 can be expressed as follows [116, D3]:

Si−H Bonds Dissociation:



Si−H Bonds Healing:



9.2 Conforming Workloads Analysis

The employed applications in the experiments of our hardware-based technique presented in Section 6.4.1.2 are: basicmath, bitcount, qsort, susan, crc, fft, adpcm, jpeg, dijkstra, gsm, patricia and stringsearch from the MiBench Benchmark Suite [27].

When exploring the impact of *conforming workloads* on aging effects the following scenarios of executing multiple simultaneous applications in parallel on top of the Linux 2.6 OS have been examined as follows:

Parallel (2)	basicmath, bitcount
Parallel (3)	basicmath, bitcount qsort
Parallel (4)	basicmath, bitcount, qsort, susan
Parallel (5)	basicmath, bitcount, qsort, susan, crc
Parallel (6)	basicmath, bitcount, qsort, susan, crc, fft
Parallel (7)	basicmath, bitcount, qsort, susan, crc, fft, adpcm
Parallel (8)	basicmath, bitcount, qsort, susan, crc, fft, adpcm, jpeg
Parallel (9)	basicmath, bitcount, qsort, susan, crc, fft, adpcm, jpeg, dijkstra
Parallel (10)	basicmath, bitcount, qsort, susan, crc, fft, adpcm, jpeg, dijkstra, gsm,
Parallel (11)	basicmath, bitcount, qsort, susan, crc, fft, adpcm, jpeg, dijkstra, gsm, patricia
Parallel (12)	basicmath, bitcount, qsort, susan, crc, fft, adpcm, jpeg, dijkstra, gsm, patricia, stringsearch

Bibliography

[1] H. Amrouch, T. Ebi, and J. Henkel, "RESI: Register-embedded self-immunity for reliability enhancement," *Computer-Aided Design of Integrated Circuits and Systems (TCAD), IEEE Transactions on*, vol. 33, no. 5, pp. 677–690, May 2014.

[2] H. Amrouch, J. Martin-Martinez, V. van Santen, M. Moras, M. Nafria, R. Rodriguez, and J. Henkel, "Connecting the physical and application level towards grasping aging effects," in *Reliability Physics Symposium (IRPS), 2015 IEEE International Conference on*, Apr 2015, pp. 3D. 1.1–3D. 1.8.

[3] H. Amrouch and J. Henkel, "Lucid infrared thermography of thermally-constrained processors," in *Low Power Electronics and Design (ISLPED), 2015 IEEE/ACM International Symposium on*, 2015, pp. 347–352.

[4] H. Amrouch, V. van Santen, T. Ebi, V. Wenzel, and J. Henkel, "Towards interdependencies of aging mechanisms," in *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*, Nov 2014, pp. 478–485.

[5] H. Amrouch, T. Ebi, and J. Henkel, "Stress balancing to mitigate NBTI effects in register files," in *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, June 2013, pp. 1–10.

[6] H. Amrouch, T. Ebi, J. Schneider, S. Parameswaran, and J. Henkel, "Analyzing the thermal hotspots in fpga-based embedded systems," in *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, Sept 2013, pp. 1–4.

[7] H. Amrouch and J. Henkel, "Self-immunity technique to improve register file integrity against soft errors," in *VLSI Design (VLSI Design), 2011 24th International Conference on*, Jan 2011, pp. 189–194.

[8] T. Ebi, H. Amrouch, and J. Henkel, "COOL: Control-based optimization of load-balancing for thermal behavior," in *Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/ Software Codesign and System Synthesis*, Oct 2012, pp. 255–264.

[9] D. Palomino, M. Shafique, H. Amrouch, A. Susin, and J. Henkel, "hevcDTM: Application-driven dynamic thermal management for high efficiency video coding," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, March 2014, pp. 1–4.

[10] H. Khdr, T. Ebi, M. Shafique, H. Amrouch, and J. Henkel, "mDTM: Multi-objective dynamic thermal management for on-chip systems," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, March 2014, pp. 1–6.

[11] A. Amouri, H. Amrouch, T. Ebi, J. Henkel, and M. Tahoori, "Accurate thermal-profile estimation and validation for fpga-mapped circuits," in *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on*, April 2013, pp. 57–60.

[12] J. Henkel, T. Ebi, H. Amrouch, and H. Khdr, "Thermal management for dependable on-chip systems," in *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, Jan 2013, pp. 113–118.

[13] T. Ebi, "Thermal management for dependable on-chip systems," Ph.D. dissertation, 2014, karlsruhe, KIT, Diss., 2014. [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000048029

[14] K. Joshi, S. Mukhopadhyay, N. Goel, and S. Mahapatra, "A consistent physical framework for N and P BTI in HKMG MOSFETs," in *Reliability Physics Symposium (IRPS), 2012 IEEE International*, 2012, pp. 5A.3.1–5A.3.10.

[15] A. Chaudhary and S. Mahapatra, "A Physical and SPICE Mobility Degradation Analysis for NBTI," *Electron Devices, IEEE Transactions on*, vol. 60, no. 7, pp. 2096–2103, 2013.

[16] Y. M. Randriamihaja, V. Huard, X. Federspiel, A. Zaka, P. Palestri, D. Rideau, D. Roy, and A. Bravaix, "Microscopic scale characterization and modeling of transistor degradation under {HC} stress," *Microelectronics Reliability*, vol. 52, no. 11, pp. 2513 – 2520, 2012.

[17] G. Georgakos, P. Huber, M. Ostermayr, E. Amirante, and F. Ruckerbauer, "Investigation of increased Multi-Bit failure rate due to neutron induced seu in advanced embedded SRAMs," *VLSI Circuits, IEEE Symposium on*, pp. 80–81, June 2007.

[18] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, pp. 501–513, May 2006.

[19] M. A. Alam, "Ece 695a lecture 5: Amorphous material/interfaces," Jan 2013. [Online]. Available: https://nanohub.org/resources/16548

[20] "Bsim4," http://www-device.eecs.berkeley.edu/bsim/?page=BSIM4.

[21] "BurnCPU Bechmark," www.es.ubuntu.com/manpages/precise/man1/cpuburn.1.html.

[22] R. Dennard, V. Rideout, E. Bassous, and A. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *Solid-State Circuits, IEEE Journal of*, vol. 9, no. 5, pp. 256–268, Oct 1974.

[23] "The End of Dennard Scaling," https://cartesianproduct.wordpress.com/2013/04/15/the-end-of-dennard-scaling.

[24] "Definition of: High-K/Metal Gate," http://www.pcmag.com/encyclopedia/term/58937/high-k-metal-gate.

[25] "LEON3: A soft-core microprocessor," http://www.gaisler.com/index.php/products/processors/leon3.

[26] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: a tool for evaluating and synthesizing multimedia and communicatons systems," *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*, pp. 330–335, 1997.

[27] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," *Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop*, pp. 3–14, 2001.

[28] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, 2008, pp. 72–81.

[29] "Probability density function definition," http://en.wikipedia.org/wiki/Probability_density_function.

[30] "Probability density function calculation," http://mathworld.wolfram.com/ProbabilityFunction.html.

[31] I. of Electrical and E. Engineers, "IEEE standard glossary of software engineering terminology," *IEEE Std 610.12-1990*, pp. 1–84, Dec 1990.

[32] "International technology roadmap for semiconductors (ITRS)," http://www.itrs.net.

[33] G. E. Moore, "The future of integrated electronics," *Fairchild Semiconductor internal publication*, vol. 2, 1964.

[34] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique, M. Tahoori, and N. Wehn, "Reliable on-chip systems in the nano-era: Lessons learnt and future trends," in *Proceedings of the 50th Annual Design Automation Conference (DAC)*, 2013, pp. 99:1–99:10.

[35] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "Lifetime reliability: toward an architectural solution," *Micro, IEEE*, vol. 25, no. 3, pp. 70–80, May 2005.

[36] J. Martin-Martinez, B. Kaczer, M. Toledano-Luque, R. Rodriguez, M. Nafria, X. Aymerich, and G. Groeseneken, "Probabilistic defect occupancy model for NBTI," in *Reliability Physics Symposium (IRPS), 2011 IEEE International*, April 2011, pp. XT.4.1–XT.4.6.

[37] J. Tandon, "The openrisc processor: open hardware and linux," *Linux Journal*, vol. 2011, no. 212, p. 6, 2011.

[38] R. Preston, R. Badeau, D. Bailey, S. Bell, L. Biro, W. Bowhill, D. Dever, S. Felix, R. Gammack, V. Germini, M. Gowan, P. Gronowski, D. Jackson, S. Mehta, S. Morton, J. Pickholtz, M. Reilly, and M. Smith, "Design of an 8-wide superscalar risc microprocessor with simultaneous multithreading," *Solid-State Circuits Conference, 2002. Digest of Technical Papers. ISSCC. 2002 IEEE International*, vol. 1, pp. 334–472 vol.1, Feb 2002.

[39] G. Memik, M. Kandemir, and O. Ozturk, "Increasing register file immunity to transient errors," in *Design, Automation and Test in Europe, 2005. Proceedings*, March 2005, pp. 586–591.

[40] J. A. Blome, S. Gupta, S. Feng, and S. Mahlke, "Cost-efficient soft error protection for embedded microprocessors," in *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, ser. CASES, 2006, pp. 421–431.

[41] N. Kurd, S. Bhamidipati, C. Mozak, J. Miller, T. Wilson, M. Nemani, and M. Chowdhury, "Westmere: A family of 32nm IA processors," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, 2010, pp. 96–97.

[42] E. Donkoh, T. S. Ong, Y. N. Too, and P. Chiang, "Register file write data gating techniques and break-even analysis model," in *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED, 2012, pp. 149–154.

[43] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "The mcpat framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing," *ACM Trans. Archit. Code Optim.*, vol. 10, no. 1, pp. 5:1–5:29, 2013.

[44] I. K. Y. Han and C. Mritz, "Temperature aware floorplanning," *In the second workshop on Temperature-Aware Computer Systems (TACS)*, 2005.

[45] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," *Computer Architecture, Proceedings. 30th Annual International Symposium on*, pp. 2–13, 2003.

[46] F. J. Mesa-Martinez, J. Nayfach-Battilana, and J. Renau, "Power model validation through thermal measurements," *SIGARCH Comput. Archit. News*, pp. 302–311, Jun 2007.

[47] F. Kurdahi, A. Eltawil, Y.-H. Park, R. Kanj, and S. Nassif, "System-level sram yield enhancement," in *Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on*, 2006, pp. 6 pp.–184.

[48] R. Joshi, R. Kanj, and S. Saroop, "Novel 4 GHz interleaved SRAM cells with asymmetrical precharge in 45 nm pdsoi technology," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 27, no. 2, pp. 278–286, 2014.

[49] G. H. Loh, Y. Xie, and B. Black, "Processor design in 3d die-stacking technologies," *Micro, IEEE*, vol. 27, no. 3, pp. 31–48, 2007.

[50] A. Pavlov and M. Sachdev, *CMOS SRAM Circuit Design and Parametric Test in Nano-Scaled Technologies: Process-Aware SRAM Design and Test*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[51] E. Amat, T. Kauerauf, R. Rodriguez, M. Nafria, X. Aymerich, R. Degraeve, and G. Groeseneken, "A comprehensive study of channel hot-carrier degradation in short channel mosfets with high-k dielectrics," *Microelectron. Eng.*, vol. 103, pp. 144–149, March 2013.

[52] S. Mahapatra, N. Goel, S. Desai, S. Gupta, B. Jose, S. Mukhopadhyay, K. Joshi, A. Jain, A. Islam, and M. Alam, "A comparative study of different physics-based nbti models," *Electron Devices, IEEE Transactions on*, vol. 60, no. 3, pp. 901–916, March 2013.

[53] A. Kerber and E. Cartier, "Bias temperature instability characterization methods," in *Bias Temperature Instability for Devices and Circuits*, T. Grasser, Ed. Springer New York, 2014, pp. 3–31. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-7909-3_1

[54] S. Khan and S. Hamdioui, "Trends and challenges of SRAM reliability in the nano-scale era," *Design and Technology of Integrated Systems in Nanoscale Era, 5th International Conference on*, pp. 1–6, 2010.

[55] S. Mahapatra, D. Saha, D. Varghese, and P. B. Kumar, "On the generation and recovery of interface traps in MOSFETs subjected to NBTI, FN, and HCI stress," *Electron Devices, IEEE Transactions on*, vol. 53, no. 7, pp. 1583–1592, 2006.

[56] S. Borkar, "Design challenges of technology scaling," *Micro, IEEE*, vol. 19, no. 4, pp. 23–29, Jul 1999.

[57] E. Cartier, A. Kerber, T. Ando, M. Frank, K. Choi, S. Krishnan, B. Linder, K. Zhao, F. Monsieur, J. Stathis, and V. Narayanan, "Fundamental aspects of HfO2-based high-k metal gate stack reliability and implications on tinv-scaling," in *Electron Devices Meeting (IEDM), 2011 IEEE International*, Dec 2011, pp. 18.4.1–18.4.4.

[58] K. Joshi, S. Hung, S. Mukhopadhyay, V. Chaudhary, N. Nanaware, B. Rajamohnan, T. Sato, M. Bevan, A. Wei, A. Noori, B. McDougal, C. Ni, G. Saheli, C. Lazik, P. Liu, D. Chu, L. Date, S. Datta, A. Brand, J. Swenberg, and S. Mahapatra, "HKMG process impact on N, P BTI: Role of thermal IL scaling, IL/HK integration and post HK nitridation," in *Reliability Physics Symposium (IRPS), 2013 IEEE International*, April 2013, pp. 4C.2.1–4C.2.10.

[59] "Microsemi, Neutron-Induced Single Event Upset (SEU)," http://www.microsemi.com/document-portal/doc_download/130760-neutron-seu-faq.

[60] J. Barth, C. Dyer, and E. Stassinopoulos, "Space, atmospheric, and terrestrial radiation environments," *Nuclear Science, IEEE Transactions on*, pp. 466–482, 2003.

[61] J. Autran *et al.*, "Real-time neutron and alpha soft-error rate testing of CMOS 130nm SRAM: Altitude versus underground measurements," *Integrated Circuit Design and Technology and Tutorial, IEEE International Conference on*, pp. 233–236, June 2008.

[62] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*, ser. Frontiers in electronic testing. Springer, 2010.

[63] S. Pagani, H. Khdr, W. Munawar, J.-J. Chen, M. Shafique, M. Li, and J. Henkel, "Tsp: Thermal safe power: Efficient power budgeting for many-core systems in dark silicon," in *Proceedings of the 2014 International Conference on Hardware/-Software Codesign and System Synthesis*, ser. CODES, 2014, pp. 10:1–10:10.

[64] "Transconductance equation from a lecture slide from the eecs instructional labs and computing at berkeley," http://www-inst.eecs.berkeley.edu/~ee105/fa98/lectures_fall_98/091898_lecture11.pdf.

[65] X. Li, J. Qin, B. Huang, X. Zhang, and J. Bernstein, "A new SPICE reliability simulation method for deep submicrometer CMOS VLSI circuits," *Device and Materials Reliability, IEEE Transactions on*, vol. 6, no. 2, pp. 247–257, June 2006.

[66] A. K. Coskun, R. Strong, D. M. Tullsen, and T. Simunic Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS, 2009, pp. 169–180.

[67] J. Srinivasan, S. V. Adve, P. Bose, J. Rivers, and C.-K. Hu, "Ramp: A model for reliability aware microprocessor design," *IBM, Poughkeepsie, NY*, 2003.

[68] V. Huard, C. Parthasarathy, A. Bravaix, C. Guerin, and E. Pion, "CMOS device design-in reliability approach in advanced nodes," in *IEEE international reliability physics symposium*, 2009, pp. 624–633.

[69] J. B. Bernstein, M. Gurfinkel, X. Li, J. Walters, Y. Shapira, and M. Talmor, "Electronic circuit reliability modeling," *Microelectronics Reliability*, vol. 46, no. 12, pp. 1957–1979, 2006.

[70] Z. Liu, B. W. McGaughy, and J. Z. Ma, "Design tools for reliability analysis," in *Proceedings of the 43rd annual Design Automation Conference*, 2006, pp. 182–187.

[71] F. Oboril and M. B. Tahoori, "Extratime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level," in *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*, 2012, pp. 1–12.

[72] W. Wang, V. Reddy, A. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, "Compact modeling and simulation of circuit reliability for 65-nm CMOS technology," *Device and Materials Reliability, IEEE Transactions on*, vol. 7, no. 4, pp. 509–517, 2007.

[73] X. Li, J. Qin, and J. B. Bernstein, "Compact modeling of MOSFET wearout mechanisms for circuit-reliability simulation," *Device and Materials Reliability, IEEE Transactions on*, vol. 8, no. 1, pp. 98–121, 2008.

[74] S. Kothawade and K. Chakraborty, "Analysis and mitigation of BTI aging in register file: An application driven approach," *Microelectronics Reliability*, vol. 53, no. 1, pp. 105–113, 2013.

[75] R. Tu, E. Rosenbaum, W. Chan, C. Li, E. Minami, K. Quader, P. Ko, and C. Hu, "Berkeley reliability tools-BERT," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 12, no. 10, pp. 1524–1534, Oct 1993.

[76] V. Huard, R. Chevallier, C. Parthasarathy, A. Mishra, N. Ruiz-Amador, F. Persin, V. Robert, A. Chimeno, E. Pion, N. Planes, D. Ney, F. Cacho, N. Kapoor, V. Kulshrestha, S. Chopra, and N. Vialle, "Managing sram reliability from bitcell to library level," in *Reliability Physics Symposium (IRPS), 2010 IEEE International*, May 2010, pp. 655–664.

[77] H. Abrishami, S. Hatami, and M. Pedram, "Multi-corner, energy-delay optimized, NBTI-aware flip-flop design," in *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, 2010, pp. 652–659.

[78] M. Bagatin, S. Gerardin, A. Paccagnella, and F. Faccio, "Impact of NBTI aging on the single-event upset of SRAM cells," *Nuclear Science, IEEE Transactions on*, vol. 57, no. 6, pp. 3245–3250, 2010.

[79] K. Kang, H. Kufluoglu, K. Roy, and M. Ashraful Alam, "Impact of negative-bias temperature instability in nanoscale SRAM array: Modeling and analysis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, pp. 1770–1781, Oct. 2007.

[80] J. Abella, X. Vera, and A. Gonzalez, "Penelope: The NBTI-aware processor," *Microarchitecture. 40th Annual IEEE/ACM International Symposium on*, pp. 85–96, Dec. 2007.

[81] L. Li, Y. Zhang, J. Yang, and J. Zhao, "Proactive NBTI mitigation for busy functional units in out-of-order microprocessors," *Proceedings of the Conference on Design, Automation and Test*, pp. 411–416, 2010.

[82] A. Bansal, R. M. Rao, J.-J. Kim, S. Zafar, J. H. Stathis, and C.-T. Chuang, "Impacts of NBTI and PBTI on SRAM static/dynamic noise margins and cell failure probability." *Microelectronics Reliability*, vol. 49, pp. 642–649, 2009.

[83] T. Siddiqua and S. Gurumurthi, "Recovery boosting: A technique to enhance NBTI recovery in SRAM arrays," *Proceedings of the IEEE Annual Symposium on VLSI*, pp. 393–398, 2010.

[84] S. Kothawade, K. Chakraborty, and S. Roy, "Analysis and mitigation of nbti aging in register file: An end-to-end approach," *Quality Electronic Design (ISQED), 12th International Symposium on*, pp. 1–7, March 2011.

[85] S. Kothawade, D. Ancajas, K. Chakraborty, and S. Roy, "Mitigating nbti in the physical register file through stress prediction," *Computer Design (ICCD), IEEE 30th International Conference on*, pp. 345–351, 2012.

[86] Y. Kunitake, T. Sato, and H. Yasuura, "Signal probability control for relieving NBTI in SRAM cells," *Quality Electronic Design (ISQED), 11th International Symposium on*, pp. 660–666, March 2010.

[87] S. Wang, T. Jin, C. Zheng, and G. Duan, "Low power aging-aware register file design by duty cycle balancing," pp. 546–549, March 2012.

[88] I. Koren and C. Krishna, *Fault-Tolerant Systems.* San Mateo, CA: Morgan Kaufmann, 2007.

[89] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on neutron-induced soft error in srams from a 250 nm to a 22 nm design rule," *Electron Devices, IEEE Transactions on*, vol. 57, pp. 1527–1538, 2010.

[90] N. Seifert, P. Slankard, M. Kirsch, B. Narasimham, V. Zia, C. Brookreson, A. Vo, S. Mitra, B. Gill, and J. Maiz, "Radiation-induced soft error rates of advanced CMOS bulk devices," *Reliability Physics Symposium Proceedings, IEEE International*, pp. 217–225, March 2006.

[91] G. Neuberger, F. de Lima, L. Carro, and R. Reis, "A multiple bit upset tolerant sram memory," *ACM Trans. Des. Autom. Electron. Syst.*, pp. 577–590, Oct. 2003.

[92] P. Reviriego and J. Maestro, "Study of the effects of multibit error correction codes on the reliability of memories in the presence of MBUs," *Device and Materials Reliability, IEEE Transactions on*, vol. 9, no. 1, pp. 31–39, March 2009.

[93] R. Naseer, R. Z. Bhatti, and J. Draper, "Analysis of soft error mitigation techniques for register files in IBM cu-08 90nm technology," *Circuits and Systems, 49th IEEE International Midwest Symposium on*, vol. 1, pp. 515–519, Aug. 2006.

[94] M. Pflanz, K. Walther, C. Galke, and H. Vierhaus, "On-line techniques for error detection and correction in processor registers with cross-parity check," *Journal of Electronic Testing*, vol. 19, pp. 501–510, 2003.

[95] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," *Microarchitecture, Proceedings. 36th Annual IEEE/ACM International Symposium on*, pp. 29–40, Dec. 2003.

[96] M. Fazeli, S. Ahmadian, and S. Miremadi, "A low energy soft error-tolerant register file architecture for embedded processors," *High Assurance Systems Engineering Symposium*, pp. 109–116, Dec. 2008.

[97] P. Montesinos, W. Liu, and J. Torrellas, "Using register lifetime predictions to protect register files against soft errors," *Dependable Systems and Networks, 37th Annual IEEE/IFIP International Conference on*, pp. 286–296, June 2007.

[98] J. Lee and A. Shrivastava, "A compiler optimization to reduce soft errors in register files," *SIGPLAN Not.*, vol. 44, pp. 41–49, June 2009.

[99] M. Kandala, W. Zhang, and L. Yang, "An area-efficient approach to improving register file reliability against transient errors," *Advanced Information Networking and Applications Workshops, 21st International Conference on*, vol. 1, pp. 798–803, May 2007.

[100] J. Hu, S. Wang, and S. Ziavras, "In-register duplication: Exploiting narrow-width value for improving register file reliability," *Dependable Systems and Networks, International Conference on*, pp. 281–290, June 2006.

[101] J. Yan and W. Zhang, "Compiler-guided register reliability improvement against soft errors," *Proceedings of the 5th ACM international conference on Embedded software*, pp. 203–209, 2005.

[102] F. Mesa-Martinez, M. Brown, J. Nayfach-Battilana, and J. Renau, "Measuring power and temperature from real processors," *Parallel and Distributed Processing, IPDPS. IEEE International Symposium on*, pp. 1–5, 2008.

[103] S. Reda, "Thermal and power characterization of real computing devices," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 1, no. 2, pp. 76–87, 2011.

[104] L. Bauer and J. Henkel, *Run-time Adaptation for Reconfigurable Embedded Processors.* Springer, 2011.

[105] P. Sundararajan, A. Gayasen, N. Vijaykrishnan, and T. Tuan, "Thermal characterization and optimization in platform FPGAs," *Computer-Aided Design, IEEE/ACM International Conference on*, pp. 443–447, Nov. 2006.

[106] E. I. Boemo and S. López-Buedo, "Thermal monitoring on fpgas using ring-oscillators," *Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications*, pp. 69–78, 1997.

[107] S. Lopez-Buedo and E. Boemo, "Making visible the thermal behaviour of embedded microprocessors on FPGAs: a progress report," *Proceedings of the ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pp. 79–86, 2004.

[108] K. M. Zick and J. P. Hayes, "On-line sensing for healthier FPGA systems," *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, pp. 239–248, 2010.

[109] M. Happe, H. Hangmann, A. Agne, and C. Plessl, "Eight ways to put your fpga on fire - a systematic study of heat generators," *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, pp. 1–6, Dec. 2012.

[110] R. Cochran, A. N. Nowroz, and S. Reda, "Post-silicon power characterization using thermal infrared emissions," *Low-Power Electronics and Design (ISLPED), ACM/IEEE International Symposium on*, pp. 331–336, Aug. 2010.

[111] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," *Computer Architecture, Proceedings of the 27th International Symposium on*, pp. 83–94, June 2000.

[112] A. N. Nowroz, R. Cochran, and S. Reda, "Thermal monitoring of real processors: techniques for sensor allocation and full characterization," *Proceedings of the 47th Design Automation Conference*, pp. 56–61, 2010.

[113] J. Keane, X. Wang, D. Persaud, and C. Kim, "An all-in-one silicon odometer for separately monitoring hci, bti, and tddb," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 4, pp. 817–829, April 2010.

[114] G. Bersuker, D. Heh, C. Young, H. Park, P. Khanal, L. Larcher, A. Padovani, P. Lenahan, J. Ryan, B. H. Lee, H. Tseng, and R. Jammy, "Breakdown in the metal/high-k gate stack: Identifying the "weak link" in the multilayer dielectric," in *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, Dec 2008, pp. 1–4.

[115] S. Zafar, A. Kumar, E. Gusev, and E. Cartier, "Threshold voltage instabilities in high-k gate dielectric stacks," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 1, pp. 45–64, March 2005.

[116] M. A. Alam, "Ece 695a reliability physics of nanotransistors," https://nanohub.org/resources/16560, Jan 2013.

[117] N. L. Anderson, R. P. Vedula, P. A. Schultz, R. M. Van Ginhoven, and A. Strachan, "First-principles investigation of low energy e center precursors in amorphous silica," *Physical review letters*, vol. 106, no. 20, p. 33004, 2011.

[118] M. Alam, K. Roy, and C. Augustine, "Reliability- and process-variation aware design of integrated circuits," *Reliability Physics Symposium (IRPS), IEEE International*, pp. 4A.1.1–4A.1.11, 2011.

[119] C. Hu, S. C.Tam, F.-C. Hsu, P.-K. Ko, T.-Y. Chan, and K. Terrill, "Hot-electron-induced mosfet degradation - model, monitor, and improvement," *Solid-State Circuits, IEEE Journal of*, vol. 20, no. 1, pp. 295–305, Feb 1985.

[120] E. Maricau and G. Gielen, "Computer-aided analog circuit design for reliability in nanometer cmos," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 1, no. 1, pp. 50–58, March 2011.

[121] S. C. Sun and J. D. Plummer, "Electron mobility in inversion and accumulation layers on thermally oxidized silicon surfaces," *Solid-State Circuits, IEEE Journal of*, vol. 15, no. 4, pp. 562–573, Aug 1980.

[122] "Bsim4 manual," http://www-device.eecs.berkeley.edu/bsim/Files/BSIM4/BSIM480/BSIM480_Manual.pdf.

[123] H. Yamauchi, "A Discussion on SRAM Circuit Design Trend in Deeper Nanometer-Scale Technologies," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 5, pp. 763–774, 2010.

[124] T. T.-H. Kim and Z. H. Kong, "Impact analysis of nbti/pbti on sram v_{MIN} and design techniques for improved sram v_{MIN}," *JSTS: Journal of Semiconductor Technology and Science*, vol. 13, no. 2, pp. 87–97, 2013.

[125] W. Zhao and Y. Cao, "Predictive technology model for nano-CMOS design exploration," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 3, no. 1, 2007.

[126] ——, "New generation of predictive technology model for sub-45 nm early design exploration," *Electron Devices, IEEE Transactions on*, vol. 53, no. 11, pp. 2816–2823, Nov 2006.

[127] "Predictive Technology Model (PTM) Website," http://ptm.asu.edu/.

[128] Y. Singh Chauhan, S. Venugopalan, N. Paydavosi, P. Kushwaha, S. Jandhyala, J. P. Duarte, S. Agnihotri, C. Yadav, H. Agarwal, A. Niknejad *et al.*, "BSIM Compact MOSFET Models for SPICE Simulation," in *Mixed Design of Integrated Circuits and Systems (MIXDES), 2013 Proceedings of the 20th International Conference*, 2013, pp. 23–28.

[129] S. Agostinelli, J. Allison, K. a. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand *et al.*, "GEANT4a simulation toolkit," *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, 2003.

[130] "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices (JEDEC STANDARD)," http://www.jedec.org/sites/default/files/docs/jesd89a.pdf.

[131] K. Mistry, C. Allen, C. Auth, B. Beattie, D. Bergstrom, M. Bost, M. Brazier, M. Buehler, A. Cappellani, R. Chau *et al.*, "A 45nm logic technology with high-k+metal gate transistors, strained silicon, 9 cu interconnect layers, 193nm dry patterning, and 100% pb-free packaging," in *Electron Devices Meeting, 2007. IEDM 2007. IEEE International*. IEEE, 2007, pp. 247–250.

[132] "Discussions with Volker Wenzel with Chair for Embedded Systems (CES), KIT."

[133] "Ngspice," http://ngspice.sourceforge.net/.

[134] R. Naseer, Y. Boulghassoul, J. Draper, S. DasGupta, and A. Witulski, "Critical charge characterization for soft error rate modeling in 90nm sram," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, May 2007, pp. 1879–1882.

[135] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *Micro, IEEE*, vol. 25, no. 6, pp. 10–16, Nov 2005.

[136] G. Bronevetsky and B. de Supinski, "Soft error vulnerability of iterative linear algebra methods," *Proceedings of the 22Nd Annual International Conference on Supercomputing*, pp. 155–164, 2008.

[137] J. Autran, P. Roche, S. Sauze, G. Gasiot, D. Munteanu, P. Loaiza, M. Zampaolo, and J. Borel, "Real-time neutron and alpha soft-error rate testing of cmos 130nm SRAM: Altitude versus underground measurements," *Integrated Circuit Design and Technology and Tutorial, IEEE International Conference on*, pp. 233–236, June 2008.

[138] B. J. A., S. Gupta, S. Feng, and S. Mahlke, "Cost-efficient soft error protection for embedded microprocessors," *Proceedings of the international conference on Compilers, architecture and synthesis for embedded systems*, pp. 421–431, 2006.

[139] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks RAID," *Proceedings of the ACM SIGMOD international conference on Management of data*, pp. 109–116, March 1988.

[140] R. Azevedo, S. Rigo, M. Bartholomeu, G. Araujo, C. Araujo, and E. Barros, "The ArchC architecture description language and tools," *International Journal of Parallel Programming*, pp. 453–484, 2005.

[141] "Open source vhdl for the hamming code."

[142] "Taiwan Semiconductor Manufacturing Company (TSMC)," http://www.tsmc.com/english/default.htm.

[143] "Xilinx Synthesis Design Guide," http://www.xilinx.com.

[144] "DIAS Infrared Camera," http://www.dias-infrared.com/pdf/pyroview380lcompact_eng.pdf.

[145] S. Ghosh, S. Basu, and N. Touba, "SEU characterization of digital circuits using weighted test programs," *Technical Report, Center for Reliable Computing, Stanford University.*, pp. 1322–1331, 2001.

[146] N. Gong, S. Jiang, J. Wang, B. Aravamudhan, K. Sekar, and R. Sridhar, "Hybrid-cell register files design for improving nbti reliability," *Microelectronics Reliability*, vol. 52, no. 9-10, pp. 1865–1869, 2012.

[147] M. Rebaudengo, M. Reorda, and M. Violante, "An accurate analysis of the effects of soft errors in the instruction and data caches of a pipelined microprocessor," *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pp. 602–607, 2003.

[148] E. Touloupis, J. Flint, V. Chouliaras, and D. Ward, "Efficient protection of the pipeline core for safety-critical processor-based systems," *Signal Processing Systems Design and Implementation, IEEE Workshop on*, pp. 188–192, 2005.

[149] "International technology roadmap for semiconductors (ITRS)," http://www.faraday-tech.com/index.html.

[150] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The Gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, 2011.

[151] A. Bhattacharjee, G. Contreras, and M. Martonosi, "Full-system chip multiprocessor power evaluations using FPGA-based emulation," in *Low Power Electronics and Design (ISLPED), ACM/IEEE International Symposium on*, 2008, pp. 335–340.

[152] "Intel SCC," www.intel.com/info/scc.

[153] T. Ebi, M. Faruque, and J. Henkel, "TAPE: Thermal-aware agent-based power econom multi/many-core architectures," in *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, Nov 2009, pp. 302–309.

[154] B. Griffith, D. Trler, and H. Goudey, "Emissivity," *Infrared Thermography. Encyclopedia of Imaging Science and Technology*, 2002.

[155] "PowerPC 440: A Hard-Core Microprocessor," https://www-01.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_Cores.

[156] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," *Proceedings of the IEEE/ACM international conference on Computer-aided design*, pp. 721–725, 2002.

[157] "Infrared Windows," http://iriss.com/.

[158] F. J. DiSalvo, "Thermoelectric cooling and power generation," *Science*, pp. 703–706, 1999.

[159] A. Coskun, T. Rosing, K. Whisnant, and K. Gross, "Temperature-aware mpsoc scheduling for reducing hot spots and gradients," *Design Automation Conference, Asia and South Pacific*, pp. 49–54, 2008.

[160] J. B. Bernstein, M. Gurfinkel, X. Li, J. Walters, Y. Shapira, and M. Talmor, "Electronic circuit reliability modeling," *Microelectronics Reliability*, vol. 46, no. 12, pp. 1957–1979, 2006.

[161] V. Traag, https://github.com/vtraag/tikz-colorbrewer.

[162] M. Thoma, https://github.com/MartinThoma/LaTeX-examples/.

[163] M. Wibrow, http://www.texample.net/tikz/examples/author/mark-wibrow/.

Acknowledgments: This product includes color specifications and designs developed by Cynthia Brewer (http://colorbrewer2.org/). The use of color schemes in TikZ is based on [161]. The TikZ codes of plotting Figures (1.2(a), 2.3) and Figure 1.13 are based on [162, 163], respectively, after modifications.