Degradation in FPGAs
Monitoring, Modeling and Mitigation

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation

von

Abdulazim Amouri

aus Aleppo-Syrien

Tag der mündlichen Prüfung: **23.04.2015**

Erster Gutachter: **Prof. Dr. Mehdi Baradaran Tahoori**,
CDNC, Karlsruher Institut für Technologie (KIT), Deutschland
Zweiter Gutachter: **Prof. Dr. Pascal Benoit**,
LIRMM, Universität Montpellier 2, Frankreich

Ich versichere hiermit wahrheitsgemäß, die Dissertation bis auf die dort angegebenen Hilfen selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und als kenntlich gemacht zu haben, was aus Arbeiten anderer und eigenen Veröffentlichungen unverändert oder mit Änderungen entnommen wurde.

Karlsruhe, 05.03.2015

_____

Abdulazim Amouri

Dedication

To whom, who taught me my first letters...
To whom, who planted in me the love of learning and discovering...
To whom, who opened my eyes to the unknowns of this world and guided me to discover...
To whom, God ordered me to acknowledge and respect them...
To my parents ... I love you too much

***

To whom, who change my life...
To whom, who taught me the patience...
To whom, who loved me, helped me, encouraged me and was always beside me...
To my wife ... I love you

***

To whom, who bring the happiness into my world...
To whom, who make me feel the responsibility...
To whom, who make me look into the future...
To my children ... I love you

***

I dedicate this work to all of my family members and to the people of Syria, who are still suffering the war for 4 years.

Acknowledgment

My cordial gratitude is presented to my advisor Prof. Dr. Mehdi Tahoori, this great person from whom I learned a lot, who guided me through my research and give me the flexibility to try and discover, who unleashed my creativity to deal with the most challenging tasks. This great person, who kept a high level of patience in spite of all my nonsense ... I'm very thankful.

I'm also grateful to my co-advisor Prof. Dr. Pascal Benoit and his assistant Dr. Florent Bruguier for the nice and fruitful collaboration and for their warm welcome during my visit to Montpellier. Furthermore, my special thanks to Prof. Dr. Jörg Henkel and his assistants Mr. Hussam Amrouch and Dr. Thomas Ebi for the fruitful collaboration.

I would like to present my heartfelt thanks to all of my colleagues and project partners at the Chair of Dependable Nano Computing (CDNC) for the nice period I spent between them and the family atmosphere they have created at our chair and also for all their valuable feedback and discussions. In particular I want to thank (in alphabetical order): Matthias Beste, Liang Chen, Mojtaba Ebrahimi, Farshad Firouzi, Saman Kiamehr, Fabian Oboril and Parthasarathy Murali Baskar Rao. I also want to thank Jochen Hepp, whom I supervised in the scope of his Master thesis for his nice work. Furthermore, my special thanks to my colleagues Fabian Oboril, Artjom Grudnitsky and Gabriel Cadilha Marques for correcting my German translation of the abstract of this dissertation. Finally, my thanks to all the secretaries and technicians at our institute in particular to Mrs. Iris Schrder-Piepka, Mrs. Renate Murr-Grobe and Mr. Martin Buchty for all of their efforts in providing a good working environment.

List of own publications included in this dissertation

Book Chapters

[*B*.1] **A. Amouri**, M. Tahoori, *"Lifetime Reliability Sensing in Modern FPGAs"*, In P. Athanas, D. Pnevmatikatos, N. Sklavos, editors, *"Embedded Systems Design with FPGAs"*, Springer, ISBN 978-1-4614-1361-5, 2013.

Conference Papers

[*C*.1] **A. Amouri**, F. Bruguier, S. Kiamehr, P. Benoit, L. Torres and M. Tahoori, *"Aging Effects in FPGAs: an Experimental Analysis"*, Proceedings of the 24th International Conference on Field Programmable Logic and Applications (**FPL**), IEEE, 2014, Munich, Germany.

[*C*.2] **A. Amouri**, J. Hepp, and M. Tahoori, *"Self-Heating Thermal-Aware Testing of FPGAs"*, Proceedings of the 32nd IEEE VLSI Test Symposium (**VTS**), pages 1-6, 2014, Napa, California, USA.

[*C*.3] **A. Amouri** and M. Tahoori, *"Degradation in FPGAs: Monitoring, Modeling and Mitigation (PHD forum paper: Thesis broad overview)"*, Proceedings of the 23rd International Conference on Field Programmable Logic and Applications (**FPL**), pages 12, IEEE, 2013, Porto, Portugal.

[*C*.4] P.M.B. Rao, **A. Amouri**, S. Kiamehr, and M. Tahoori, *"Altering LUT Configuration for Wear-out Mitigation of FPGA-Mapped Designs"*, Proceedings of the 23rd International Conference on Field Programmable Logic and Applications (**FPL**), pages 18, IEEE, 2013, Porto, Portugal.

[*C*.5] **A. Amouri**, H. Amrouch, T. Ebi, J. Henkel and M. Tahoori, *"Accurate Thermal-Profile Estimation and Validation for FPGA-Mapped Circuits"*, Proceedings of the 21st IEEE International Symposium on Field-Programmable Custom Computing Machines (**FCCM**), pages 57-60, 2013, Seattle, Washington, USA.

[*C*.6] **A. Amouri**, S. Kiamehr and M. Tahoori, *"Investigation of Aging Effects in Different Implementations and Structures of Programmable Routing Resources of FPGAs"*, Proceedings of the 2012 International Conference of Field-Programmable Technology (**FPT**), pages 215-219, IEEE, 2012, Seoul, South Korea.

[*C*.7] **A. Amouri** and M. Tahoori, *"High-Level Aging Estimation For FPGA-Mapped Designs"*, Proceedings of the 22nd International Conference on Field Programmable Logic and Applications (**FPL**), pages 284-291, IEEE, 2012, Oslo, Norway.

[*C*.8] S. Kiamehr, **A. Amouri** and M. Tahoori, *"Investigation of NBTI and PBTI Induced Aging in Different LUT Implementations"*, Proceedings of the 2011 International Conference of Field-Programmable Technology (**FPT**), pages 1-8, IEEE, 2011, New Delhi, India

[*C*.9] **A. Amouri** and M. Tahoori, *"A Low-Cost Sensor for Aging and Late Transitions Detection in Modern FPGAs"*, Proceedings of the 21st International Conference on Field Programmable Logic and Applications (**FPL**), pages 329-335, IEEE, 2011, Chania, Greece.

Journal Papers

[*J*.1] **A. Amouri**, J. Hepp, and M. Tahoori, *"Built-in Self-Heating Thermal Testing of FPGAs"*, Transactions on Computer-Aided Design of Integrated Circuits and Systems (**TCAD**) (Accepted for publication, November 2015)

Abstract

The quest for high performance circuits with low power consumption, large scale integration and low cost, is the main reason behind the continuous shrinking of transistors' dimensions at nano-scale. This shrinking has already reached critical limits posing various manufacturing and reliability challenges to *Very Large Scale Integration* (VLSI) chips. Nowadays, the transistor degradation mechanisms (a.k.a. transistor aging) and the thermal-related issues (heat dissipation, hot spots, etc.) are among the most important challenges that affect the reliability of the VLSI chips.

Field Programmable Gate Arrays (FPGAs) are general purpose reconfigurable chips providing many unique features and capabilities over *Application-Specific Integrated Circuits* (ASICs). The FPGA design philosophy as compared to that of ASIC is based on both the concept of rapid development (faster time to market) and the in-field programming concept. Hence, unlike ASICs, the circuit can be deployed directly onto the FPGA without the need to worry about the manufacturing process and the FPGA chips can be easily reconfigured in the field as desired to accomplish different functions. For these reasons, FPGAs are used in a wide range of applications, where faster time to market and/or in-field reconfigurations are required. For many of these applications the reliability is very critical and comes as a first priority. State-of-the-art FPGAs, pushed by the ever-increasing demands on higher performance and lower power, use the latest advancements in VLSI technology, and thus they share most of the reliability challenges associated with nano-scale technology nodes. Therefore, to guarantee the correct functionality during the lifetime of FPGA-mapped systems in the field, proper techniques at various levels should be devised.

Transistor aging, as an important reliability threat at nano-scale technologies, causes changes in the transistor characteristics, which slow down the switching speed of the transistor over time and in turn leads to timing failures in the chip. This issue has been addressed in literature mainly for ASICs and at the device-level alone. In order to be able to fortify against the aging effects in FPGA, it is necessary to estimate the amount of degradation for the whole FPGA-mapped circuit early at design phase. However, it is infeasible to estimate the aging for a complete design using directly the models and techniques proposed at device-level. This is because of the simulation overhead at device-level and its complexity. That is why only experimental methods have been followed in literature to investigate the aging of FPGA. Based on that, there is a need for fast and accurate aging estimation at higher level, even if it is inherently less accurate than the device-level correspondent. In fact, to properly deal with aging in FPGAs, it requires monitoring, modeling and mitigation at device and architecture levels and this must be integrated in the tool-chain used for developing the FPGA designs at user side.

This dissertation targets the transistor aging degradation as well as the associated thermal challenges in FPGAs (since there is an exponential relation between aging and chip temperature). The main objectives are to perform experimentation, analysis and device-

level model abstraction for modeling the degradation in FPGAs, then to monitor the FPGA to keep track of aging rates and ultimately to propose an aging-aware FPGA design flow to mitigate the aging. The contributions presented in this dissertation can be divided into three main parts: modeling, monitoring and mitigation. For modeling the aging effects in FPGA, device-level investigations are carried out on fundamental FPGA building blocks (look-up tables and routing switch matrices) to understand the effects of aging on FPGA architecture. Furthermore, FPGA stress experiments with controlled stress conditions are made to measure the extent of degradation and the role of different parameters in the FPGA degradation. Additionally, a method for estimating the thermal-profile of FPGA-mapped designs is proposed, which is verified by real thermal measurements. Based on the results of these real experiments, an aging modeling framework for FPGA is proposed by combining both the accurate device-level models and the implicit high-level details in the FPGA design tools. Using this framework, several mitigation strategies are proposed to reduce the effect of aging on the total lifetime of the FPGA-mapped design. These strategies are based on an aging-aware logic mapping that targets directly the look-up tables of the FPGA.

As the effects of aging appear on a long time-scale and also they are process- and usage-dependent, a tracking of these effects is important to maintain the correct functionality of the mapped design throughout the FPGA lifetime. For monitoring the aging effects in FPGAs, logic-based sensors exploiting the existing programmable FPGA resources, are presented to monitor the effect of aging on the most critical paths of FPGA-mapped designs. Additionally, different techniques are proposed for thermal-aware functional testing of the FPGA using self-heating modules that are built on top of FPGA programmable resources.

The results of these contributions have been integrated in an aging-analysis-and-mitigation toolset, which takes as input the information available to the FPGA-users about their designs in order to estimate the total expected lifetime and performs various modifications to meet the lifetime reliability requirements. These results can be even integrated in the FPGA tools provided by the manufacturer with more accurate models for aging-aware placement and routing. Additionally the investigation results can be used to build an aging-aware FPGA chip, in which the FPGA building blocks are designed with aging effects in mind from the transistor-level upwards to the whole FPGA.

Zusammenfassung

Die Suche nach hochleistungsfähigen Schaltungen mit niedrigem Stromverbrauch, hoher Integrationsdichte und geringen Kosten, ist der Hauptgrund für das kontinuierliche Schrumpfen der Transistordimensionen bis in den Nanometer-Bereich. Das Schrumpfen hat jedoch bereits kritische Grenzen erreicht, die verschiedene Herstellung- und Zuverlässigkeitsherausforderungen an *Very Large Scale Integration* (VLSI)-Chips stellen. Heutzutage gehören die Transistorabbaumechanismen (auch bekannt als Transistoralterung) und die thermischen Probleme (Wärmeableitung, Hot Spots, etc.) zu den wichtigsten Herausforderungen, die die Zuverlässigkeit der VLSI-Chips beeinflussen.

Im-Feld programmierbare Logik-Gatter-Anordnungen (FPGAs) sind rekonfigurierbare Chips für allgemeine Anwendungen, die viele einzigartige Funktionen und Fähigkeiten gegenüber *anwendungsspezifische integrierte Schaltungen* (ASICs) besizen. Die FPGA-Entwurfsphilosophie basiert dabei, im Vergleich zu jener der ASICs, auf zwei wesentlichen Konzepten: Schnelle Prototypen-Entwicklung (und damit schnellere Markt-einführung) und der Rekonfigurierbarkeit während der Nutzung. Deshalb können, anders als in ASICs, Schaltkreise auf einem FPGA implementiert werden, ohne sich um den Herstellungsprozess zu kümmern. Die FPGA-Chips können im Betrieb nach Wunsch neu konfiguriert werden, um verschiedene Funktionen auszuführen. Aus diesen Gründen werden FPGAs in einer Vielzahl von Anwendungen eingesetzt, in denen eine schnellere Markteinführung und/oder im Feld Neukonfigurationen erforderlich sind. Für viele dieser Anwendungen ist die Zuverlässigkeit sehr kritisch und hat daher erste Priorität. Moderne FPGAs nutzen die neuesten Fortschritte der VLSI-Technologie, durch die ständig steigende Anforderungen an die höhere Leistung und geringerem Stromverbrauch, und teilen sich damit die meisten Zuverlässigkeitsherausforderungen, die mit Nanometer-technologien verbunden sind. Deshalb müssen sachgemäße Techniken auf verschiedenen Abstraktionsebenen entwickelt werden, um die korrekte Funktion im Betrieb der FPGA-basierten Systeme zu gewährleisten.

Transistoralterung, als eine wichtige Zuverlässigkeitsherausforderung bei Strukturgrössen im Nanobereich, führt zu Veränderungen in den Transistoreigenschaften, die die Schaltgeschwindigkeit der Transistorn mit der Zeit verlangsamen und damit wiederum zu Chipausfällen führen können. Dieses Problem wurde in der Literatur vor allem für ASICs und auf Transistoren-Ebene angegangen. Um in der Lage zu sein die Alterungseffekte in FPGAs verlangsamen zu können, ist es notwendig die Alterung früh in der Entwurfsphase abzuschätzen. Es ist jedoch unmöglich, die Alterung für einen kompletten Entwurf direkt mit Modellen und Techniken auf Transistor-Ebene abzuschätzen, aufgrund des Simulationsaufwands auf Transistor-Ebene und der verbundenen Komplexität. Deshalb sind nur experimentelle Methoden in der Literatur verfolgt worden, um die Alterung von FPGAs zu untersuchen. Somit gibt es einen großen Bedarf für schnelle und genaue Alterungsschätzungen auf höheren Abstraktionsebenen, auch wenn diese weniger exakt sind,

als die Transistor-Ebene-Abschätzung. Für den richtigen Umgang mit der Alterung in FPGAs werden Überwachung, Modellierung und Minderung der Alterung auf Transistor- und Architekturebenen benötigt. Darüber hinaus muss all dies in die Werkzeugkette, die für die Entwicklung der FPGA-Designs auf Anwenderseite verwendet wird, integriert werden.

Diese Dissertation befasst sich mit der Alterung von FPGAs, sowie den damit verbundenen thermischen Probleme in FPGAs (da es eine exponentielle Beziehung zwischen Alterung und Chiptemperatur gibt). Die in dieser Arbeit vorgestellten Beiträge lassen sich in drei Hauptteile gliedern: Modellierung, Überwachung und Minderung. Für die Modellierung der Alterungseffekte in FPGAs werden Untersuchungen auf Transistor-Ebene für die Grund-FPGA-Bausteine durchgeführt (Look-up Tables und Routing Schaltmatrizen), um die Auswirkungen der Alterung auf die FPGA-Architektur zu verstehen. Weiterhin werden FPGA-Stressexperimente mit kontrollierten Belastungsbedingungen durchgeführt, um das Ausmaß der Alterung und den Einfluss der verschiedenen Parameter auf die Alterung von FPGAs zu messen. Zusätzlich wird ein Verfahren zum Abschätzen des Wärmeprofils vorgeschlagen, dass durch reale thermische Messungen verifiziert/validiert wird. Basierend auf den Ergebnissen dieser realen Experimente wird eine Platform zur Alterungsmodellierung für FPGAs vorgeschlagen. Mit dieser Platform werden verschiedene Abwehrstrategien entwickelt, die die Auswirkungen der Alterung auf die Gesamtlaufzeit des FPGAs reduzieren. Diese Strategien basieren auf einer alterungsbewusste Logik-Zuordnung, und arbeiten direkt auf den Look-up Tables des FPGAs.

Da die Auswirkungen der Alterungseffekte über einen langen Zeitraum erkennbar werden und sie auch Prozess- und Nutzungsabhängig sind, ist eine Überwachung dieser Effekte wichtig, um die richtige Funktionalität über die gesamte FPGA Lebensdauer zu halten. Zur Überwachung der Alterungseffekte in FPGAs werden Logik-Sensoren, die die bestehenden programmierbaren FPGA-Ressourcen nutzen, entwickelt, um den Effekt der Alterung auf den kritischen Pfaden der FPGA-basierten Entwurfen zu überwachen. Zusätzlich werden verschiedene Techniken für die thermische sensitive Funktionalitätsprüfung des FPGAs durch sich selbst aufheizende Module vorgeschlagen, die auf den Resourcen des FPGAs aufgebaut sind.

Die Ergebnisse dieser Beiträge wurden in ein Toolset für Alterungsanalyse und Minderung integriert, dass als Eingabe die Information, die dem FPGA-Nutzer über seine Entwürfe zur Verfügung stehen, nimmt, um die gesamte erwartete Lebensdauer zu schätzen und dann verschiedene Modifikationen durchführt, um die Anforderungen bzgl. der Zuverlässigkeit über die gesamten Lebensdauer zu erfüllen. Diese Ergebnisse können auch mit den FPGA-Tools von kommerziellen Herstellern mit genaueren Alterungsmodellen für alterungsbewusste Platzierung und Verdrahtung benutzt werden. Zusätzlich können die Untersuchungsergebnisse verwendet werden, um einen alterungsbewusstes FPGA-Chip zu bauen, in dem die FPGA-Bausteine mit Berücksichtigung von Alterungseffekten von der Transistor-Ebene aufwärts entwickelt werden.

Contents

Chapter 1.

Introduction

The invention of *Field Programmable Gate Arrays* (FPGAs) in the late 1980's has revolutionized the modern life in many domains. First, it enabled the concept of hardware emulation, which helps in accelerating the design process of *Integrated Circuits* (ICs) in the means of allowing the design to be early prototyped on real hardware, with minimal cost. This allows debugging and verifying the design in the field of operation before the final fabrication of ICs takes place, which is normally a very costly process. Hence, the introduction of FPGAs in the design process of ICs saves a lot of development efforts and reduces the cost and time significantly.

In addition to hardware emulation, the unique feature of FPGAs of being general purpose reconfigurable chips, has opened the doors for new domain of systems that can be adapted in the field according to the environmental changes or user requirements. Examples of such systems include remote-area systems, space applications, automobile systems, telecommunication systems, etc. The in-field adaptation is usually very hard to be implemented using *Application Specific Integrated Circuits* (ASICs)-based systems, which makes the FPGAs a preferable choice for these systems. The increasing need for such systems in many application domains was and still behind the rapid development of FPGAs during the last two decades.

Nowadays, the state-of-the-art FPGAs, due to their exploitation for the latest advancements in transistor technology, have many advanced properties in terms of performance, power consumption and logic densities[1], making the FPGAs practical alternatives or even more preferable than the ASICs for low and medium-volume applications in several domains. In the following, the relation between the FPGAs and the latest advancement in the transistor technology is highlighted.

1.1. FPGAs and technology scaling

Since the introduction of ICs for the first time more than 60 years ago, their development has witnessed a huge number of leaps. For example, the integration density measured by the number of transistors per chip has been increased dramatically from few thousand transistors in the 1970s to more than 20 billion transistors in 2014 (Virtex® UltraScale™XCVU440 FPGA from Xilinx [1]). This rapid development is actually pushed by the non-stoppable demands for circuits with higher performance, lower power consumption, lower cost and easier portability. These circuit requirements are explained in the following: 1) **Higher performance**: for many applications, the performance is the first

[1]The number of logic gates that can be implemented on the chip

distinguishing aspect of the circuit. Circuits with higher performance allow more tasks to be completed in shorter time periods. Following the main rule of "performance is never enough", there are continuous demands for higher performance circuits. 2) **Lower power consumption**: the importance of this requirement comes from its relation to two other factors; the power consumption cost and/or the limited availability of power sources. The later factor is dominant in portable battery-running devices and in remote-area applications (e.g., mobile phones, laptops, heart-beat regulators, satellites, wireless sensors). The first factor however, is related to most of the applications and it represents both the operating costs (i.e., the electricity bill) and/or the infrastructure price (e.g., battery or power harvesting devices). These important factors raise the demands for lower-power circuits. 3) **Lower cost**: the cost of the circuit plays a role mostly in the marketing and directly affects the manufacturer revenue. For many applications, the circuit cost should not exceed a certain budget. 4) **Easier portability**: the smaller the device is, the more mobile it will be. This fact pushes toward smaller circuits with higher integration densities.

Although these four circuit requirements are the main motivation for the IC development, in most cases it is hard to satisfy them all at the same time. This is because of their strong interdependence that makes changing any of them affecting the rest significantly. For example, enhancing the performance of the circuit could increase its power consumption and raise its cost or vice versa. Therefore, a trade-off between these requirements is usually made to fulfill the application needs.

In order to satisfy the above requirements and continue coping with all market demands, the chip manufacturers used to follow a "magical solution" represented in downscaling the device technology. This is because smaller transistors are cheaper and also tend to be faster and consume less power. This solution was successful for a long period. Nowadays, the downscaling has already reached to a level where the performance and power consumption cannot be enhanced at the same rate as before. In spite of that, the downscaling continues, because it can still enhance the circuit cost and portability or it can be used to achieve higher level of integration, which means more and different functionality can be added to the chip. In other words, with the downscaling, the same number of transistors can be packed in a smaller chip area (i.e. more chips can be produced from a single wafer) or more transistors can be packed in the same chip area (i.e. adding additional functionality to the chip). Since the main cost represents in wafer production, smaller chips are cheaper and vice versa.

FPGAs are in the front line to benefit from the most advanced technology nodes, not only to meet the increasing demands of high performance and low power digital and mixed-signal applications, but also because FPGAs have regular and scalable structure and they are produced in high-volume. These reasons make the downscaling of FPGAs a more straightforward process than it is for the other types of chips.

The state-of-the-art FPGA series from the major FPGA manufacturers use the advanced 20 nm technology together with 3D stacking IC technology [2] and with tri-gate transistor technology [3]. Furthermore, FPGAs based on 16 nm and 14 nm technologies are also in plan [4, 5]. This excessive scaling allows the FPGA to have a very high integration density, which reaches to $\approx$6.8 billion transistors in the largest commercially-available FPGA [6]. In one of the upcoming FPGAs, this number of transistors is 2-3 times larger [2].

Having these integration capabilities, the state-of-the-art FPGAs provide very advanced

features. The upcoming Virtex® UltraScale™XCVU440 FPGA from Xilinx [7], as an example, can provide up to 4.4 million logic cells (which is equivalent to ≈50 million ASIC gates), 88.6 Mbit of internal BlockRAMs, 2880 of internal *Digital Signal Processor* (DSP) blocks and 1456 I/O pins. Additionally, it offers several hardened *Intellectual Properties* (IPs) and interfaces, such as 16 Gb/s transceivers, 100 Gbit Ethernet units, *Peripheral Component Interconnect* (PCI) Express blocks, etc. Similar features are also announced for the upcoming Stratix®10 FPGA from Altera [8] the main competitor of Xilinx.

These advanced features and the continuous development of modern FPGAs has also extended the range of their application domains to include even applications with few or no reconfiguration requirements. The relatively short time-to-market that the FPGAs can offer and their lower cost for low and medium-volume application in comparison to ASIC, play the main role in this extension. Currently, FPGAs are used in a wide range of domains including both critical and non-critical applications such as aerospace and defense, wired and wireless communications, smart networks, ASIC prototyping, video and imaging applications, consumer electronics, high-performance computing, factory automation, data centers, automotive applications, security systems, financial acceleration, high-end instrumentation and in many medical solutions [9].

Unfortunately, the advanced features of modern FPGAs, as a result of the excessive downscaling and the very high integration density, do not come free of cost. There are many manufacturing and reliability challenges accompanied with them. In the next section, the main reliability challenges facing the modern FPGAs are pointed out.

1.2. Degradation and reliability challenges of FPGAs

The reliability, which is defined as the consistent performing according to the specifications, means that the manufactured chip must continue functioning as specified during the whole expected lifetime. For many applications, especially critical ones, the reliability of the circuit is of the highest priority. However, as the downscaling of device technology reaches at nano-scale, many challenges rise up quickly and manifest themselves as manufacturing and reliability obstacles. Some of these challenges include manufacturing variability, sub-threshold leakage, heat dissipation, thermal hot spots, increased circuit noise sensitivity, and reliability concerns due to transient (e.g. radiation-induced soft errors) and permanent (e.g. transistor aging) failures [10, 11, 12, 13]. The importance of these challenges comes from the fact that they lead to more failures both during the manufacturing phase and at runtime. This is represented in lower manufacturing yield, more runtime failures and shorter lifetime of the chip than expected. All of these may lead to catastrophes in field of operation if no counter-measures are used. As FPGAs are utilized in many critical domains, in which the reliability is of a great importance, an appropriate addressing for each of these challenges is required through all the design phases. This is to assure the correct operation through the expected lifetime.

Transistor aging in particular, is one of the most important reliability challenges at nano-scale [14, 13]. It happens on a relatively long time period (i.e. several months to years). In this period, and under the effect of different factors, the transistor threshold voltage increases slowly, which reduces the switching speed and in turn increases the delay of the

circuit. This continues until the delay of the circuit reaches a critical limit after which timing failures start to happen. In other words, the degradation effect that results from transistor aging, appears first after having the chip deployed in the operational environment, which raises many concerns about the reliability. Usually, large design margins are considered to make up for the delay increase due to transistor aging and to extend the useful lifetime of the circuit. However, these margins prevent the exploitation of the maximum performance of the chip.

Actually, transistor aging is a result of several degradation mechanisms. These are namely *Bias Temperature Instability* (BTI) [15, 16], which consists of two corresponding mechanisms; the *Negative BTI* (NBTI) and the *Positive BTI* (PBTI), *Hot Carrier Injection* (HCI) [17, 18] and *Time-Dependent Dielectric Breakdown* (TDDB) [19]. The amount of degradation caused by these mechanisms is related to several circuit and environmental parameters (e.g. chip temperature, supply voltage, circuit switching activities, design signal probabilities, etc.). Some of these parameters have a larger impact on accelerating the degradation than other parameters. With adequate investigation and analysis, these parameters can be controlled to mitigate the degradation and prolong the useful lifetime of the chip. To achieve this in FPGAs, a deep understanding for the origin of the degradation and how it propagates from the device-level up to the system-level is required.

In addition to transistor aging, thermal-related issues such as heat dissipation and hot spots are also among the most important reliability challenges in modern VLSI chips [20]. In fact, increasing chip temperature raises the power consumption and it may results in cracks and permanent damage to the chip and/or nearby components. Furthermore, chip temperature has an exponential effect on the amount of degradation caused by transistor aging mechanisms. Thus, increasing chip temperature accelerates the aging process [21]. Therefore, addressing the thermal-related issues in FPGAs is a precondition for addressing the transistor aging challenge. This requires a deep understanding for the FPGA architecture and the FPGA power distribution.

The challenge of transistor aging has been addressed in literature predominantly for ASICs and with a main focus on the device-level alone. Because of the simulation overhead at device-level and its complexity, it is infeasible to apply the models and techniques proposed at device-level to estimate the aging for a complete design. Additionally, some of the main factors that influence the aging are originated from higher-levels (e.g., it is extremely usage, workload and temperature dependent), which requires access to higher-level details for correct estimation at device-level. Aside from that, in FPGAs most of the device-level details are proprietary to the manufacturer and the users have access only to the final fabricated chip. Therefore, only few works have investigated the aging of FPGA and this is done experimentally. However, there exists no work for estimating the aging for a complete FPGA-mapped design as required by FPGA designers in order to be able to fortify against its effects. Based on that, there is a need for fast and accurate aging estimation at higher level, even if it is inherently less accurate than the device-level correspondent. In fact, to properly deal with aging in FPGAs, it requires monitoring, modeling and mitigation at device and architecture levels as well as the tool-chain used for developing the FPGA designs at user side.

1.3. Dissertation focus and goals

Having the main goal of enhancing the reliability of FPGAs, this dissertation targets mainly the degradation in FPGAs due to transistor aging as well as the related thermal challenges. Novel techniques and methods are proposed to handle these challenges at different levels. The main objectives are to model the degradation in FPGAs by performing experimentation, analysis and higher level abstraction for the device-level models, then to keep track of aging rates in the FPGA by proposing aging monitors and ultimately to propose an aging-aware FPGA design flow to mitigate the aging. The proposed aging analysis and mitigation techniques are integrated on top of traditional (commercial) FPGA design toolchain, which shows the applicability of these techniques for real designs. The work accomplished in the scope of this dissertation can be divided into three main phases; modeling, monitoring and mitigation. The relation between these phases is depicted in Figure 1.1

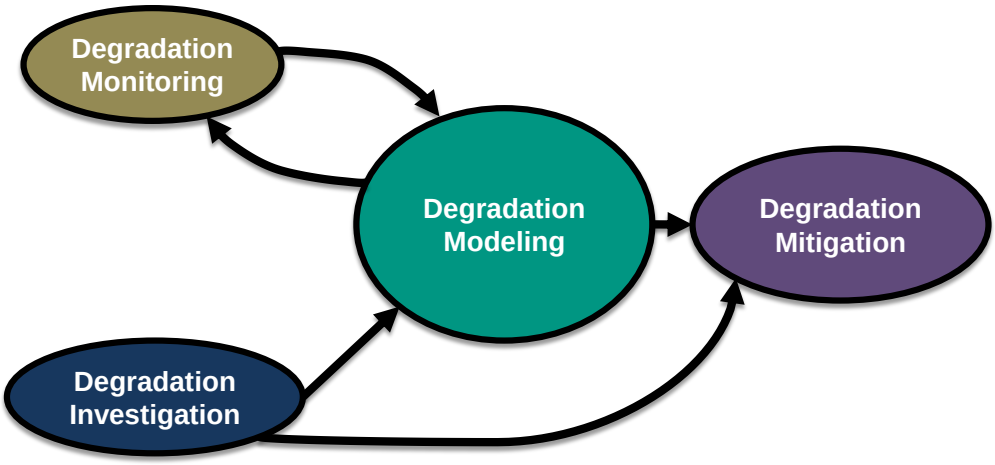


Figure 1.1.: The relation between the different phases of work accomplished in the scope of this dissertation

1.3.1. Investigation and Modeling

In this phase, the first purpose is to have a deep understanding for the FPGA architecture and to investigate the effects of the degradation mechanisms and the related thermal challenges in FPGAs. To achieve this, device-level investigations are carried out on fundamental FPGA building blocks (e.g. *look-up tables* (LUTs), switch matrices, etc.) to understand the effects of aging on FPGA architecture. Furthermore, FPGA stress experiments with controlled stress conditions are made to measure the extent of degradation and the role of different parameters in the FPGA degradation. Additionally, a method for estimating the thermal-profile of FPGA-mapped designs is proposed, which is verified by real thermal measurements. The results of these investigations are exploited for abstracting the

transistor-level models of the aging mechanisms to *register-transfer-level* (RTL), such that they can be applied during the mapping of RTL designs to the FPGA. Based on these abstracted models, an aging analysis tool is implemented on top of the FPGA synthesis and mapping toolchain for estimating the effects of degradation and thermal challenges early in the design phase, such that appropriate strategies for the mitigation of these challenges can be applied.

1.3.2. Monitoring

Since aging is process, environment and workload dependent, it is necessary to continuously monitor the amount of circuit degradation to gauge the remaining lifetime and to maintain the correct functionality of the FPGA. The work done in the scope of the monitoring phase focuses on developing methods and techniques that enable both the FPGA designer at the design phase, and the FPGA end-user in the field of operation, to monitor when the effects of transistor aging and thermal issues start to cause malfunctions in the circuit operation and even to warn before these errors are near to happen. This monitoring ability gives the possibility to consider suitable countermeasures to reduce the effect of these challenges and to enhance the FPGA reliability before any failure happens. The monitors are implemented using native programmable resources of FPGAs, which eliminates the need for any external equipment. Furthermore, the sensitivity of these monitors can be calibrated to work either as aging sensors or as warning sensors before the aging start to cause errors. In addition to the aging monitors, a thermal-aware testing for FPGAs using self-heating method is proposed, which enables testing the FPGA at different thermal corners without the need for external equipment.

1.3.3. Mitigation

Based on the results of the modeling and monitoring efforts, the work in this phase proposes a set of strategies to mitigate the effect of aging in FPGAs. The strategies come in form of aging-aware logic mapping techniques that target directly the look-up tables of the FPGA. Additionally several counter-measures and recommendations are discussed through the whole dissertation to avoid critical degradation effects.

The main contributions of this dissertation are summarized in the following:

- Investigating the effect of BTI mechanism in different LUT structures inside the FPGA. It is found out that the delay degradation due to transistor aging depends on the mapped configuration, usage (input signal probability) as well as the specific LUT structure. Moreover, it is shown that the configuration history of an LUT has a considerable effect on the delay degradation of the currently-used configuration of that LUT.

- Investigating the effect of BTI mechanism in different types of programmable routing switches in the FPGA using device-level simulations. The effects of different

parameters on the aging of routing resources are studied. It is found out that different parameters affect the amount of degradation differently. Furthermore, it is shown which structure is the best in terms of aging degradation under different aging mechanisms arising from different technology nodes.

- Introducing a flow to accurately estimate the thermal profile of FPGA-mapped designs validated by thermal-camera measurements. The flow is based on a method that distributes the leakage power properly across the FPGA chip. This method uses a temperature-leakage loop estimation model for distributing and adapting the leakage power for more accurate thermal simulation. The results of testing several designs, with different sizes and frequencies, show that the presented approach can achieve accurate thermal-profile estimation with average absolute estimation error of around 1°C across the chip when compared to the camera measurements.

- Analyzing the aging effects in FPGA experimentally using accelerated lifetime conditions. The focus of this analysis is to identify the main parameters and phenomena influencing the performance degradation of FPGAs. This is done using a set of controlled ring-oscillator-based sensors with different lengths and tunable activity control that are implemented on a Spartan-6 FPGA. The FPGA is exposed to accelerated-lifetime conditions using elevated temperatures and voltages in a controlled environment. The results show the extent of performance degradations, the impact of usage, the correlation of process variation and aging, the aging of unused resources, and the relative impacts of BTI and HCI aging factors.

- Abstracting both the BTI and HCI device level models to RTL models. This allows faster aging estimation for the FPGA-mapped designs than the device-level models. These abstracted models, in addition to implicit device-level information existed in the power and timing reports provided from the FPGA's vendor tools, are used to present a tool for high-level aging estimation in FPGA. The tool is used to predict the amount of aging-induced degradation for designs mapped to FPGA devices. This helps the designers, in an early phase of the design flow, to choose the appropriate mapping and/or optimization efforts, to prolong the lifetime of their FPGA-mapped designs.

- Presenting the design and mapping of a low-cost logic-level aging sensor for FPGA-based designs. The sensor is designed to provide controlled sensitivity, ranging from a warning sensor to warn before the degradation starts to cause failures, to late transition detector that detect when the failures happen. The Area, delay, and power overhead of a set of sensors mapped for most aging-critical paths of representative designs prove to be very modest.

- Introducing self-heating integration techniques for thermal-aware testing of FPGAs chips, in which the internal resources of FPGA are used to build controlled self-heating elements (SHEs). These controlled SHEs are distributed across the FPGA and integrated with the test scheme to generate the required temperature profile for

testing, and thus no external devices for heating up the FPGA are needed. Two different categories of SHEs integration techniques are presented to fit different testing purposes; the first category is for built-in self-test (BIST), and the second one is for application-dependent testing.

- Proposing two methods to mitigate the BTI-induced aging in LUTs of the FPGA. The mitigation is performed by manipulating the configuration of the used LUTs and their input signal probabilities, while maintaining the functionality of the mapped design.

1.4. Dissertation outline

The outline of the reminder of this dissertation is as follows:

Chapter 2 gives an overview about the general FPGA structure and the main building blocks that form its resources. This overview is followed by a background about transistor aging and the degradation mechanisms behind it. An elaborated discussion is then given on how these degradation mechanisms affect the transistors from both physical and device-level points of view.

The contributions of the dissertation are divided into three parts as mentioned before in Section 1.3:

Part I (the investigation and modeling part) contains the following chapters: Chapter 3 describes the investigation process of the BTI mechanism in three different LUT structures of the FPGA, reviews the related work in this area and presents the main conclusions. Afterwards, Chapter 4 describes the investigation of BTI in routing resources of FPGA and how different parameters play different roles in the amount of the resulted degradation. Chapter 5 presents the results and the main conclusions of analyzing the aging effects in FPGA experimentally using accelerated lifetime conditions. Chapter 6 then, introduces the thermal-profile estimation flow of the FPGA-mapped designs together with a review for the related work and the experimental results. At the end, Chapter 7 presents the structure and the implementation details of the high-level aging estimation tool, which can be used for aging analysis.

Part II (the monitoring part) contains the following chapters: Chapter 8 presents the design and mapping of the proposed aging sensors, which can be calibrated to be warning or aging sensors. The chapter also introduces the concept of aging-critical paths, reviews the related work and presents a method for mapping the sensors to monitor the aging-critical paths. Chapter 9 introduces the concept and the design of self-heating in FPGA, reviews the related work and presents an example for integrating the SHEs in an available BIST structure for FPGA.

Part III (the mitigation part) contains Chapter 10 which describes two methods for mitigating the BTI effect in FPGA LUTs. The chapter also reviews the related work and gives the implementation detail of these methods on an open-source FPGA synthesis and mapping tool.

At the end, Chapter 11 concludes this dissertation and highlights the possible future work.

Chapter 2.

Background

In order to monitor, model and mitigate the degradation in FPGAs, it is necessary to understand the architecture of modern FPGAs and the degradation mechanisms at nano-scale. The main focus in this dissertation will be on *Static Random Access Memory* (SRAM)-based FPGAs, which use SRAM cells to store the configuration bits. This is because SRAM-based FPGAs are the most common type available on markets these days and they are also expected to continue to be the fastest growing ones over the next few years [22]. Other types of FPGAs that use non-volatile storage such as Flash-based FPGAs are less common. In this chapter, the FPGA architecture and its main components are explained. This is followed by a review for the main degradation mechanisms behind circuit aging.

2.1. FPGA architecture

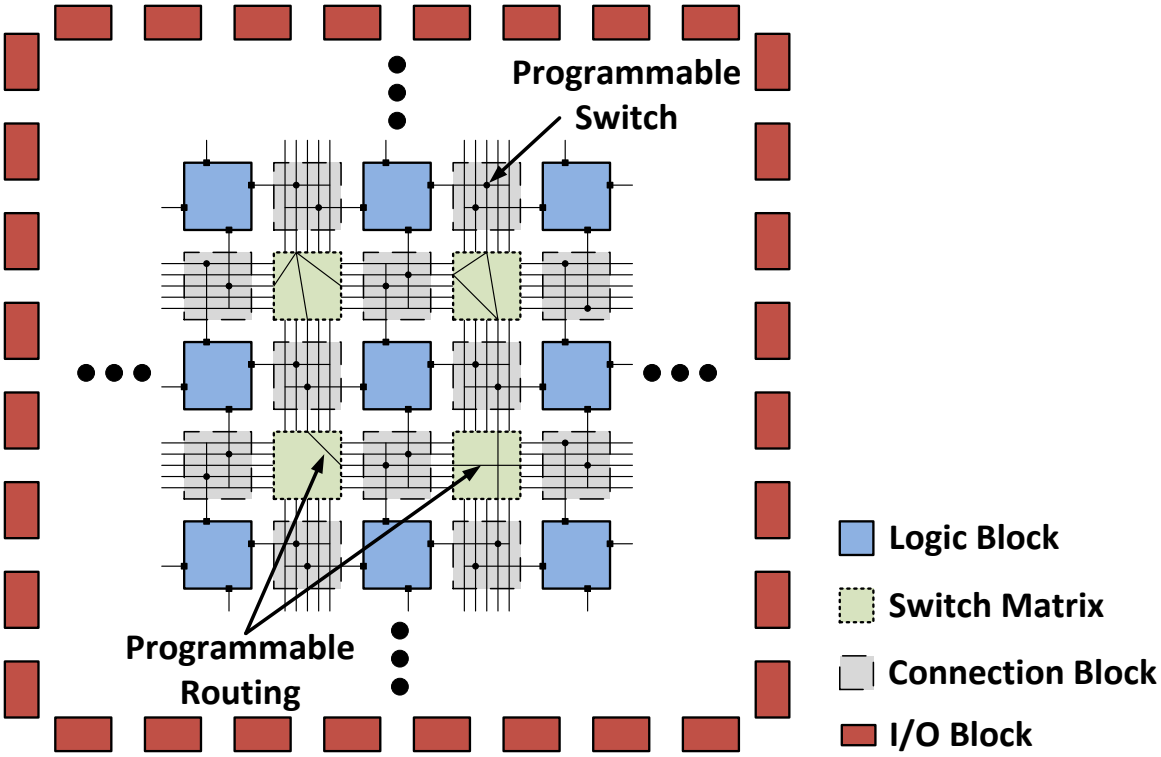


Figure 2.1.: An abstracted FPGA architecture (island-style)

Being general purpose chips, FPGAs usually offer a large number of configurable logic components for mapping the logic operations. Additionally, a large variety of other components to fulfill different purposes (e.g., communication, acceleration, storage, etc.) are also available in different versions of FPGA series. However, most of modern commercial SRAM-based FPGAs share almost the same architecture depicted in Figure 2.1. This architecture is usually called the *island-style* [23]. It consists of a set of logic blocks (also known as configurable logic blocks), which represent the core of the programmable fabric. These blocks are connected together via a set of switch matrices, and surrounded by a set of programmable I/O blocks located at the periphery of the chip. The I/O blocks provide different programmable I/O connections to the FPGA with different standards. The following subsections give an overview about each of these blocks and the additional components that might exist in an FPGA.

2.1.1. Configurable Logic Block (CLB)

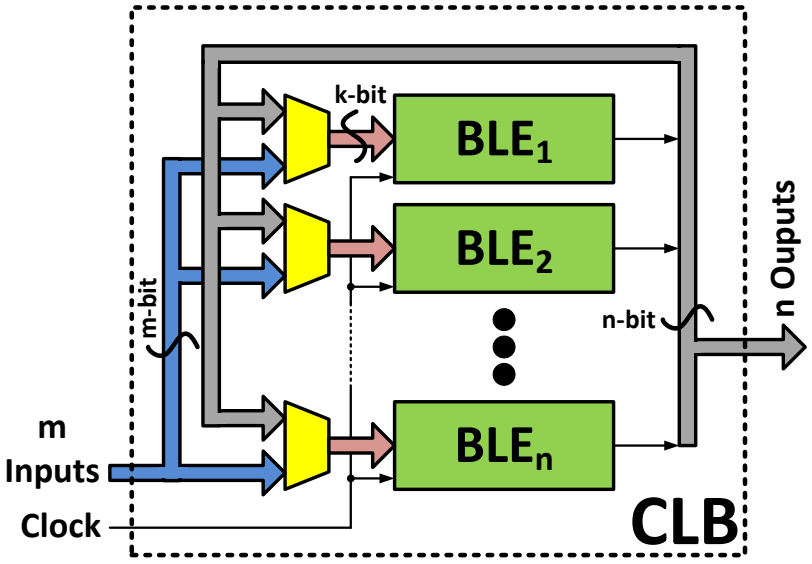
The CLB is the main place to map the logic operations in FPGA. That is why it is available in large quantity inside the FPGA. Many commercial FPGAs share a common structure for the CLB. This structure consists of a set of *Basic Logic Elements* (BLEs), which are connected together via local interconnects [24] (see Figure 2.2(a)). Each BLE in turn, contains a k-input *look-up table* (LUT) and a register (D-type *flip-flop* (D-FF)). A multiplexer is used to select whether the LUT output is registered or not (see Figure 2.2(b)).

The LUT forms the basic building block of FPGA. It consists of a k-input multiplexer; whose job is to select one of the 2^k inputs programmed in SRAM memory cells (see Figure 2.3(a)). With this structure, a k-input LUT can be programmed to perform any k-input logical function. Figure 2.3(b) shows an example of 2-input LUT programmed to perform the XOR function.

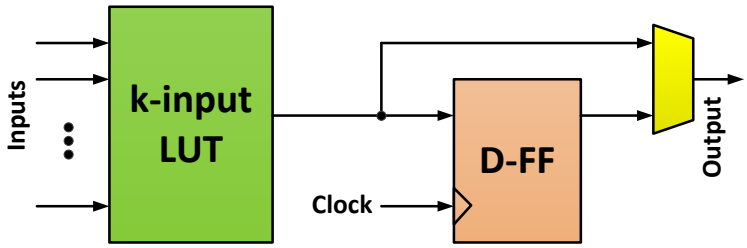
The optimal size of an LUT (i.e. the number of inputs it has) is basically a trade-off between the area it occupy and the performance it can provide [23]. Nowadays, commercial FPGAs contain LUTs with up to 8 inputs [8]. In addition to LUTs and registers, modern FPGAs contain additional components packed inside the CLB to improve the performance such as arithmetic circuitry to perform carry addition operations, and register shift operations.

2.1.2. Switch Matrix

The purpose of the switch matrices is to set the connections between different CLBs as required by the circuit mapped onto the FPGA. This is done using a set of programmable switches that activate/deactivate the connections as needed to provide the required routing. In the island-style FPGA architecture (Figure 2.1), these matrices exist on the sides of each CLB, which provide a high connection flexibility. It should be noted that these matrices are identical; the variation of connections shown in the figure is just examples of the possible connections. The connection blocks shown in the figure, provide an interface between the CLBs and the wiring around them. In many modern FPGA architectures, the connection blocks and the switch matrices are merged together for each cluster of CLBs.

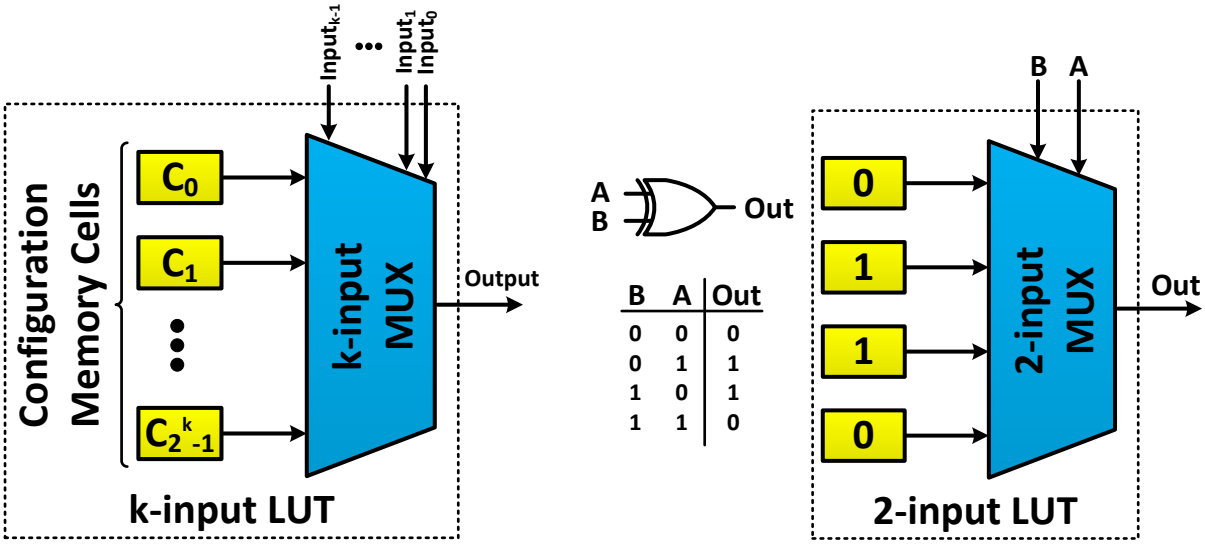


(a) CLB



(b) BLE

Figure 2.2.: The CLB structure



(a) (b)

Figure 2.3.: The LUT structure: (a) a general k-input LUT and (b) 2-input LUT implementing the XOR function

The number of possible connections between the wires in each matrix is usually too large. This means, implementing a switch for each possible connection inside the switch matrix is infeasible in terms of area. Therefore, the number of switches inside the switch matrix is usually limited to a certain number that provide a good trade-off between area, rout-ability and performance (which is affected by the length of the routes). Usually, the routing structure of the FPGA is defined by; i) the channel width (W): which represents the number of wires in each channel or track, ii) the switch matrix flexibility (F_s): which represents the number of wires to which each incoming wire can connect in a switch matrix, iii) the connection block flexibility (F_c): which represents the number of wires in each channel to which a CLB input or output pin can connect, and iv) the segmented wire length: which represents the number of CLBs a wire segment spans [23]. Figure 2.4 shows an example of a switch matrix with W= 5, F_s = 3 and F_c= 3.

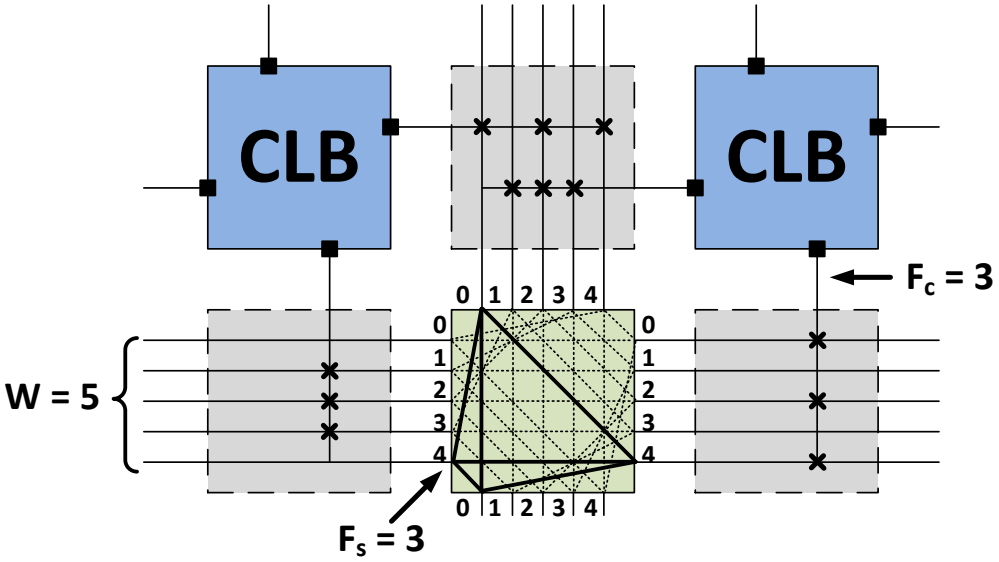


Figure 2.4.: A switch matrix example

In order to improve the performance, modern FPGAs provide additional direct point-to-point wiring with different lengths to connect non-adjacent CLBs as well as adjacent CLBs.

2.1.3. Other components

In addition to CLBs, modern FPGAs integrate other components to increase the performance and also to support different types of interfaces. The following list includes the most common components:

- **BlockRAMs:** These are RAM blocks, each with a size of few Kbits, integrated between the CLBs and distributed across the FPGA. The purpose of them is to provide a faster access/storage to the data processed by the mapped circuit.

- **Digital Signal Processors (DSPs):** As the name implies, these are a set of small processors that are designed to perform the "multiply-and-add" operation, which is required by most digital signal processing algorithms. Similar to the BlockRAMs, the DSPs are integrated between the CLBs and distributed across the FPGA.

- **General purpose processors:** Some FPGAs include one or two instances of hardened general purpose processors such as PowerPc and ARM processors. These are made to facilitate the hardware and software co-design of modern systems.

- **Digital Clock Managers (DCMs):** Each of these modules includes a *phase-locked loop* (PLL) with additional circuitry to perform several operations on the system clock, such as clock multiplication and division. Also they can be used to generate clock signal with certain duty cycles or to generate several synchronous clock signals, etc.

- **Intellectual Properties (IPs):** There exist different versions of FPGAs, each with different sets of hardened IPs optimized to target a specific domain of applications. Examples of hardened IPs in FPGA include multi Gbit/s transceivers, Ethernet units, sensors, different types of controllers, different type of I/O standard interfaces, etc.

With this short background about the FPGA architecture, it becomes clear that the CLBs and the switch matrices form the main components of the FPGA. Therefore, studying the degradation effects on FPGA should begin by these basic building blocks. In Chapter 3 and 4, several transistor-level architectures representing the LUTs and the routing switches are investigated under the effect of transistor aging. In the following section, the main degradation mechanisms that cause the circuit aging are reviewed.

2.2. Circuit aging

Aging (also known as wearout) is one of the most important reliability challenges facing the VLSI chips at nano-scale. It happens gradually on a long time scale, in which the delay of the circuit increases until the point that the timing requirements are not met. After that the circuit starts to fail, i.e. generating timing failures.

The aging effects can be observed in the transistors as well as the interconnects (i.e., the wires) inside the chip. However, the degradation mechanisms that affect the transistors are different from those affecting the interconnects. In this section, both types of mechanisms are reviewed with a focus on the main mechanisms targeted in this dissertation.

2.2.1. Transistor aging

The transistor aging causes the magnitude of the threshold voltage of transistors to increase, which is accompanied with a degradation of the mobility, drain current, and transconductance of the transistor. Hence, the switching delay of the transistor increases, and as a result, the delay of the circuit functional paths can exceed the timing specifications. Once

this happens, the circuit starts to fail. This can greatly reduce the operational lifetime of FPGA chips.

There are several degradation mechanisms behind the transistor aging namely BTI, HCI and TDDB [14, 13]. In the following each of these mechanisms is explained.

2.2.1.1. Bias Temperature Instability (BTI)

The BTI mechanism [25, 26, 27, 28] consists of two corresponding phenomena: NBTI, which has effect on *P-type Metal-Oxide-Semiconductor* (PMOS) transistors, and PBTI, affecting *N-type Metal-Oxide-Semiconductor* (NMOS) transistors. In previous technology nodes and fabrication schemes, the PBTI effect was negligible in comparison to NBTI and was mostly ignored. However, since the introduction of high-κ / metal gates transistors in sub 45 nm technology, the PBTI effect becomes comparable to the NBTI one [16, 29, 30] (see Figure 2.5). Consequently, both effects should be considered in new technologies.

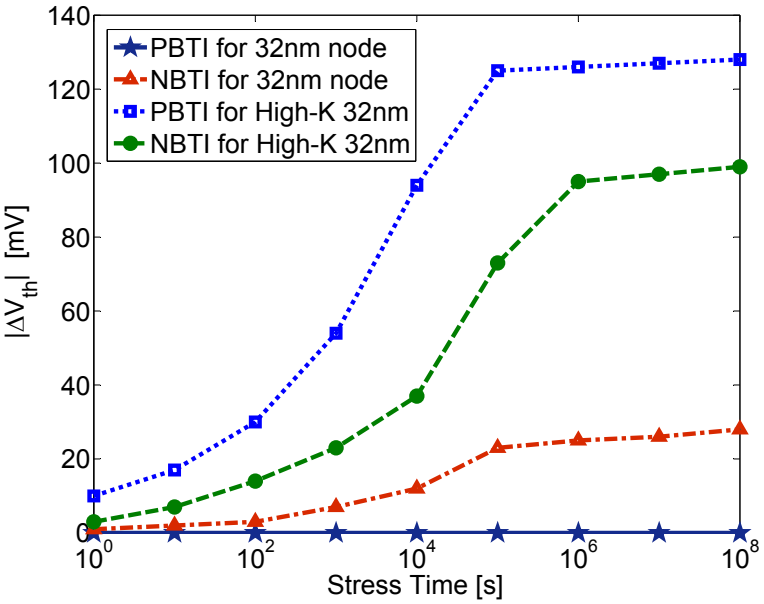


Figure 2.5.: V_{th} shift induced by NBTI and PBTI [16]

The NBTI (PBTI) can be distinguished from the other degradation mechanisms through their two phases of influence, in which their effect is observable on the magnitude of threshold voltage (V_{th}) of PMOS (NMOS) transistors (see Figure 2.6):

- Stress phase: In this phase, the magnitude of V_{th} increases, which slows down the switching speed of the transistor. The transistor enters the stress phase when it is switched ON. i.e., when the gate-source voltage is reversely biased ($V_{gs} = -V_{DD}$) for

PMOS under NBTI, or when the gate-source voltage is positively biased ($V_{gs} = V_{DD}$) for NMOS under PBTI.

- Recovery phase: The transistor enters the recovery phase when its gate-source voltage is removed ($V_{gs} \approx 0$). In this phase, the magnitude of V_{th} of the transistor decreases back toward its initial value [15]. However, this recovery cannot completely compensate the effect of the stress phase. Consequently, the overall effect of BTI is an increase in the magnitude of threshold voltage over the time (see Figure 2.6).

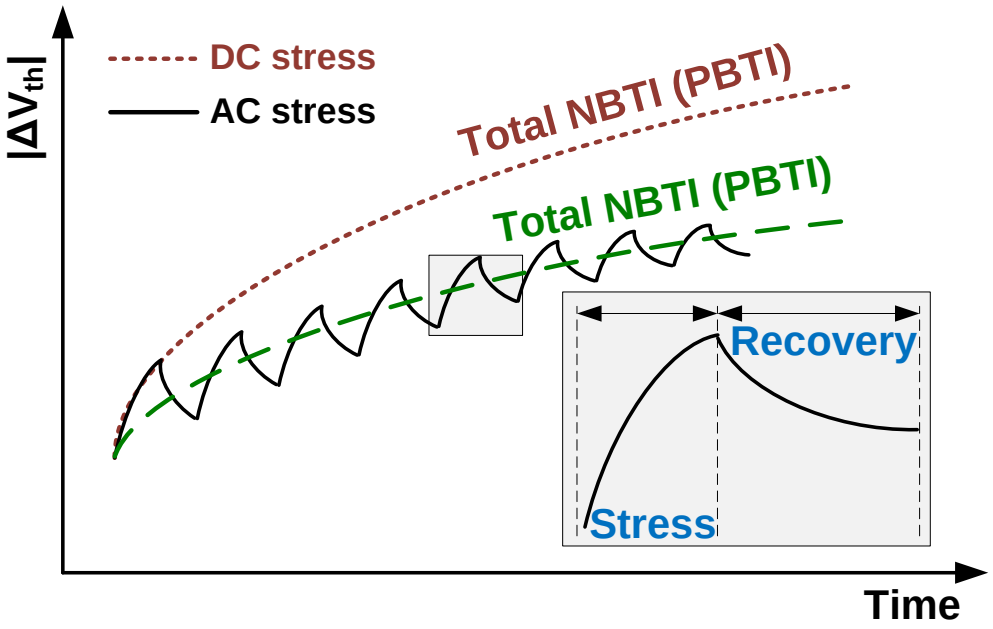


Figure 2.6.: BTI-induced V_{th} change during stress and recovery for both AC and DC type of stress

The transistor can face two types of stress; continuous (DC) and interrupted (AC) stress. If a DC type of stress is applied (i.e., when the transistor is always ON), there will be no recovery phase and the effect on V_{th} will be much severe than in the AC case (see Figure 2.6). The ratio between the stress-time to the total-time is designated as the duty cycle. At 50% duty cycle, the net ΔV_{th} is less than one-half of its DC value, with little or no frequency dependence up to 500 kHz [31].

The physical mechanism behind BTI is still not completely understood [26, 27, 28]. However, there exist two different theories that can be used to explain and model the stress and recovery behavior of BTI: i) Reaction-Diffusion (RD) theory and ii) Charge Trapping/Detrapping (TD) theory. Some believe that both of these theories are tightly coupled and responsible for the BTI effect [32]. Some also believes that TD has the dominant effect in earlier lifetime [27] and RD is the dominant one later on [28]. In fact, there are

many versions and modifications for these two theories to match the experimental measurements [28]. The following paragraphs briefly highlight these two theories.

Reaction-Diffusion (RD) theory This theory is proposed initially to explain the NBTI mechanism. It is based on the dissociation of the interface bonds between *Silicon* (Si) and *Hydrogen* (H) atoms at the SiO2/Si substrate interface under the combinatorial effect of electric field and elevated temperature [33, 34] (see Figure 2.7). When the gate of the PMOS transistor is negatively biased, the holes from the channel are attracted to the Si-H bonds, which weaken these bonds (this is the reaction part). At elevated temperature, the Si-H bonds are broken and the H diffuses either into the oxide or the Si substrate (this is the diffusion part), which result in dangling bonds that are called interface traps (D_{it} or N_{it}) at the SiO2/Si substrate interface. These traps cause a positive oxide charge that increases the threshold voltage of the transistor. The model presented by this theory faced many criticisms, because it fails to explain and match the experimental measurements of the recovery phase at short time scale [27]. This made many of the researchers to believe that the TD theory, which will be explained in the next paragraph, is the main contributor to the BTI effect, especially because it is also in line with the explanation of PBTI in the high-κ dielectric [27].

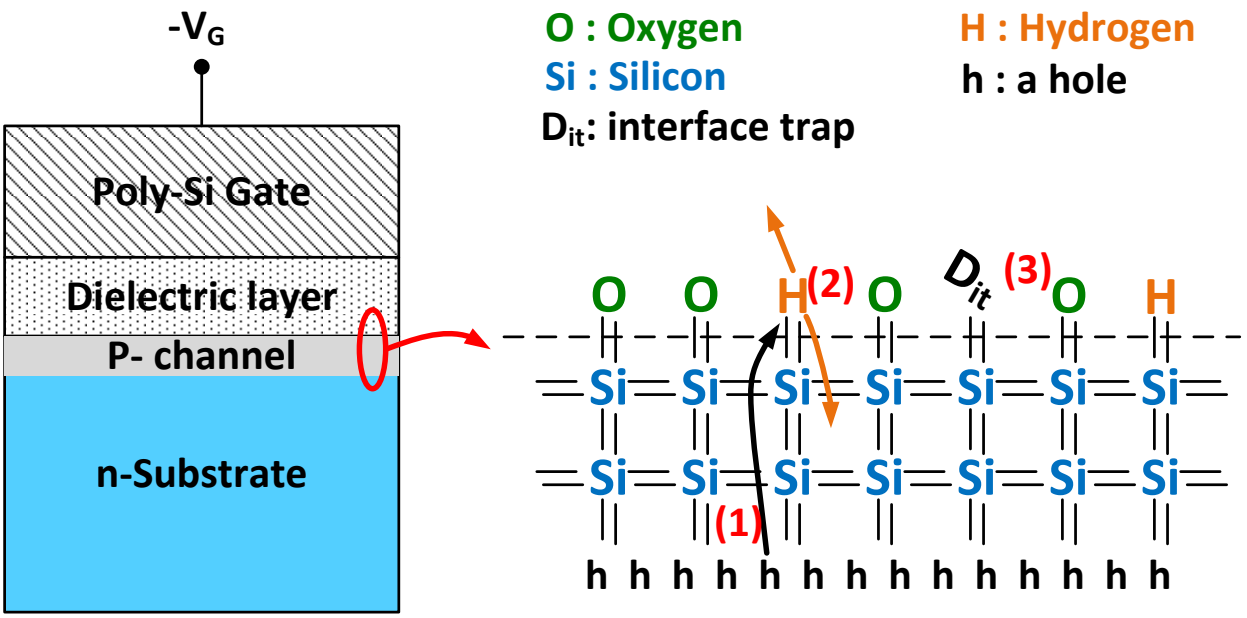


Figure 2.7.: The RD physical mechanism (NBTI) [14]

Charge Trapping/Detrapping (TD) theory This theory is based on the assumption that there are process-related preexisting defects at the SiO2/Si substrate interface [35, 36]. These defects include bulk traps that cause the charge (holes for PMOS in NBTI or

electrons for NMOS in PBTI) trapping when the transistor is ON (i.e., under stress), and the charge detrapping when the transistor is OFF (i.e., in the recovery phase). Although the model provided for this theory can explains the recovery at short time scales, it has been contradicted by many experimental measurements and proven to be not the only physical mechanism behind BTI [28, 36].

Transistor-level BTI model As the main purpose of this dissertation is to study the lifetime of the FPGA on long time scale, the focus will be on the BTI model that can best fit the measurement data on the long-term. The BTI model provided for the RD theory is reported to successfully achieve that [28]. Therefore, it is adopted in this dissertation for all the simulations. Since the methodologies for all the simulations are also provided, it should be easy to adopt any other model in any future investigation.

The degradation caused by BTI is usually modeled as a change in the V_{th} of the transistor. The adopted model of this change is introduced in [25, 15] for NBTI effect. This model handles the long-term upper-bound V_{th} change (i.e., worst case ΔV_{th}):

$$\Delta V_{th} = \left(\frac{\sqrt{{K_v}^2 Y T_{clk}}}{1 - \beta_m^{1/2n}} \right)^{2n} \tag{2.1}$$

where n is a constant that can be either 1/6 or 1/4 depending on the fabrication process. Y is the duty cycle, which represents the ratio of stress time (i.e., when the transistor is ON) to total time and T_{clk} is the time period of one stress-recovery cycle. The supply voltage (V_{dd}) is a part of the technology dependent factor (K_v), which can be found together with the other parameters and their values in [25, 15].

If the device related parameters are gathered in one constant, the model in Equation (2.1) can then be abstracted to the following simple form:

$$\Delta V_{th} = A_{BTI1} \times Y^n \times t^n \times e^{(\frac{-E_a}{kT})} \tag{2.2}$$

where A_{BTI1} is a technology dependent factor that is also a function of supply voltage (V_{dd}), t is the time (transistor's age) and T is the temperature in Kelvin. k is Boltzmann's constant, and E_a is a fitting parameter.

Using the alpha-power law [37], the degradation can be also presented as a change in the switching delay [38, 39]:

$$\Delta d = K \times \Delta V_{th} \times d_0 \tag{2.3}$$

where K is a constant, d_0 is the fresh (i.e., non-aged) switching delay, Δd is the increase in the switching delay due to NBTI.

Substituting ΔV_{th} from Equation (2.2) in Equation (2.3) gives:

$$\Delta d = A_{BTI2} \times Y^n \times t^n \times e^{(\frac{-E_a}{kT})} \times d_0 \tag{2.4}$$

where A_{BTI2} is the result of multiplying the technology dependent factor A_{BTI1} from Equation (2.2) by the constant K from Equation (2.3).

As the long-term PBTI effect is dual to the NBTI one, the same model in Equation (2.1) can be used for estimating the PBTI effect on NMOS transistors as well [40, 41].

2.2.1.2. Hot Carrier Injection (HCI)

Similar to the BTI, the HCI phenomenon (also known as *Hot Carrier Effect* (HCE)), causes also the threshold voltage of the transistors to increase, which in turn leads to similar consequences, like BTI, increasing the delay of the functional paths, and reducing the lifetime of the chips. However, unlike the BTI, there is no recovery phase for the HCI effect and the resulted increase in the V_{th} is permanent (see Figure 2.8). Additionally, the HCI is accelerated by temperature and the electric field [14], the same as BTI, but the electric field that causes the HCI effect is usually larger than the one which results in the BTI effect [26]. According to [42, 43, 14], there are four mechanisms staying behind the HCI phenomenon. These are explained in the following paragraphs.

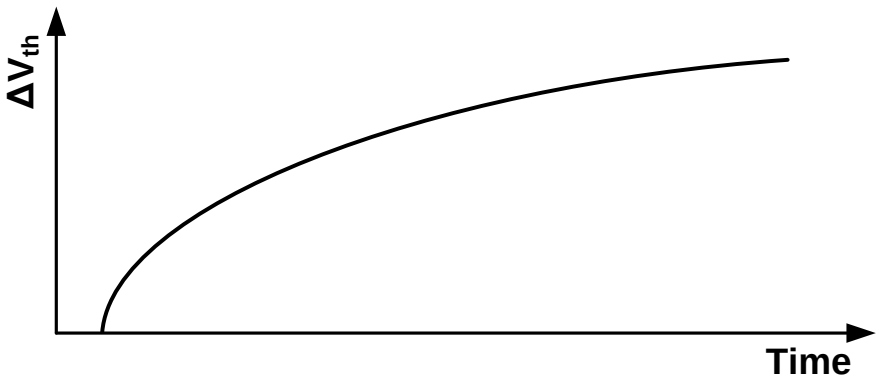


Figure 2.8.: HCI-induced V_{th} change

Drain Avalanche Hot Carrier (DAHC) The DAHC injection is considered to cause the worst degradation among the four HCI mechanisms. It happens when a high voltage is applied at the drain ($V_D > V_G$) causing the channel carriers (electrons for NMOS and holes for PMOS) to be accelerated into the depletion region of the drain. These accelerated carriers will collide with the silicon atoms. The effect is that some of them will gain a little more energy than the average, which makes them able to overcome the electric potential barrier between the silicon substrate and the gate oxide, and will get injected into the gate oxide layer where they are sometimes trapped (see Fig. 2.9). The worst case is when $V_D = 2V_G$. Over the time, these trapped carriers will eventually build up electric charge within the dielectric layer, which will increase the threshold voltage needed to turn the transistor on.

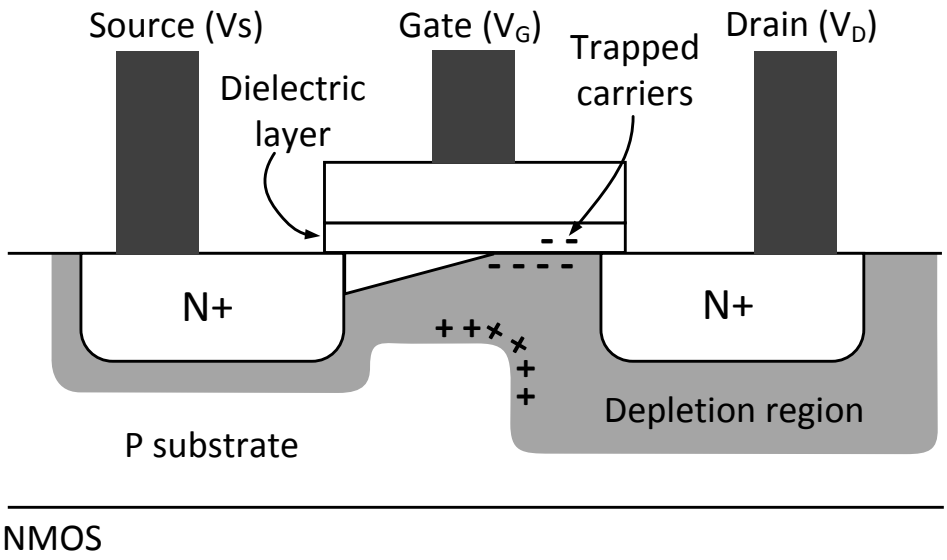


Figure 2.9.: HCI- DAHC mechanism [43]

Channel Hot Electron (CHE) In the CHE injection, channel carriers that travel from the source to the drain are sometimes, because of the high gate voltage, directed to the gate oxide layer and they get injected there. This happens when the source voltage (V_S) is significantly lower than both the drain voltage (V_D) and the gate voltage (V_G), while $V_D \approx V_G$.

Substrate Hot Electron (SHEl) In the SHEl injection, carriers in the substrate are directed under the effect of the substrate field to the substrate-oxide interface. While they move there, their kinetic energy increases due to the effect of the high field in surface depletion region. These carriers eventually overcome the energy barrier of the surface and get injected into the gate oxide; some of them get trapped there. This happens when the absolute value of the substrate back bias voltage (V_B) is very large (i.e., $|V_B| >> 0$).

Secondary Generated Hot Electron (SGHE) In the SGHE injection, carriers are generated from impact ionization including a secondary carrier that was created similarly by a previous incident of impact ionization. The conditions of the SGHE are similar to the DAHC. The main difference between them is the influence of the V_B in the generation of the hot carriers, where a field is created under the effect of V_B that direct the hot carriers generated by the secondary carriers to the surface region.

Although these four mechanisms are used to explain the HCI since the eighties, the recent studies (e.g., [18, 44]) shows that the HCI is still a main reliability problem for modern devices.

Transistor-level HCI model The effect of HCI on NMOS transistor is empirically found and presented as a change in V_{th} [45, 18, 17]. If the device related parameters are

gathered in one constant and the alpha-power law is used, the degradation can be presented as a change in the switching delay (Δd) as follows:

$$\Delta d = A_{HCI} \times \alpha \times f \times t^{0.5} \times e^{(\frac{-E_b}{kT})} \times d_0 \tag{2.5}$$

where d_0 is the fresh (i.e., non-aged) switching delay, A_{HCI} is a technology dependent factor that is also a function of supply voltage (V_{dd}), t is the time, E_b is a fitting parameter, α is the activity factor of the transistor, and f is the frequency. The factor $\alpha \times f$ is known as the activity rate, which represents the switching activity of the transistor (i.e., the rate of transitions in Hz) and is referred to by AR in this dissertation.

2.2.1.3. The main influencing parameters for BTI and HCI

The transistor-level models of both BTI and HCI show that there is a set of main parameters that influence the BTI and HCI degradation. Table 2.1 summarizes these parameters and shows their relation with the HCI and BTI effect.

Table 2.1.: Aging mechanisms and their relations with the main parameters

Factor	T	Vdd	AR	Y
BTI	exponential	exponential	-	sub-linear
HCI	exponential	exponential	sub-linear	-

2.2.1.4. Time Dependent Dielectric Breakdown (TDDB)

The TDDB (also known as gate oxide breakdown), as the name indicates, is a breakdown in the gate oxide layer of the transistor that happens gradually over the time. It causes a slowdown in the transistor switching speed and an increase in the leakage current, and ends by losing the dielectric layer to its insulating properties, which cause a permanent failure in the chip [46, 13]. This degradation process goes through three stages (see Figure 2.10):

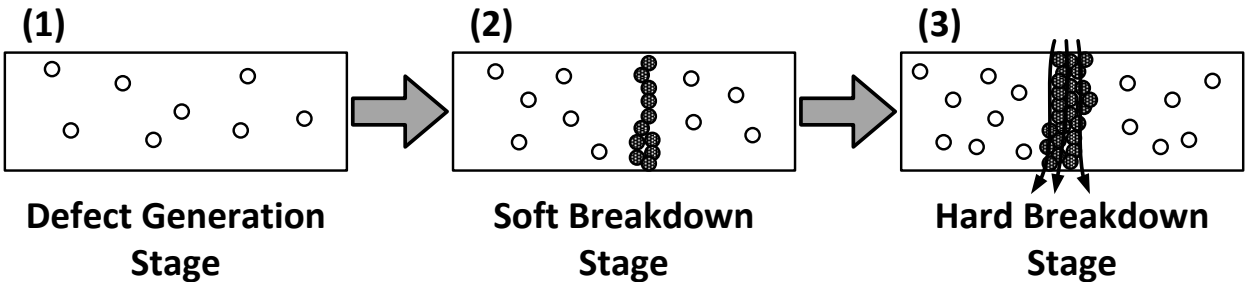


Figure 2.10.: The three stages of the TDDB in a cross-sectional view of the gate oxide [13]

1. **Defect Generation Stage**: The application of high electric field at the gate oxide layer results in defects (traps) generation in this layer. At the beginning, these defects are non-overlapping and thus do not conduct.

2. **Soft Breakdown Stage**: If the stress continues, more defects are generated and they start to overlap, which may form a resistive conducting path from the gate to the channel of the transistor. In this stage, the transistor continues to operate, however with a slowdown in the switching speed and an increase in the leakage current.

3. **Hard Breakdown Stage**: Once the conducting path is formed, more traps appear due to thermal damage causing the conducting path to become wider and hence more leakage current flows through it. The increasing leakage current rises the temperature and results in thermal runaway that finally leads to a breakdown in the dielectric layer.

As it can be noticed, the TDDB can be distinguished from the BTI and HCI effects by the increase in the transistor's leakage current that it causes. One test method for the TDDB effect depends on monitoring this leakage increase after applying a constant stress to calculate the *Time-To-Failure* (TTF) of the transistor [47]. Although the TDDB effect is important, it is not targeted in this dissertation and left as a possible future extension.

2.2.2. Interconnects aging

As mentioned at the beginning of Section 2.2, both the transistors and the interconnects inside the chip are susceptible to degradation. There are two main degradation mechanisms that affect the interconnects inside the chip. These are namely the *Electro-Migration* (EM) and the *Stress Migration* (SM) [43]. Both of them cause the metal atoms to migrate from one side of the wire to the other side, which slowly cause voids at the side where the atoms migrated from, that ultimately result in open failures. Also hillocks are created at the side to which the atoms migrated, which lead to short circuit failures.

For the EM, the migration is a result of the current flows in the wire, where the atoms migrate in the direction of the electrons. In the SM, the migration happens due to the stress resulted from the thermal difference alone at the sides of the wires without applying any current, and the atoms migrate from the highly stressed areas to the less stressed ones. The interconnects aging is not targeted in this dissertation.

Part I.

Investigation and Modeling

Chapter 3.

Investigating The Degradation of LUTs

3.1. Introduction

As mentioned in Chapter 2, LUTs are the basic blocks for mapping Boolean functions. Hence, studying the effect of aging on these basic blocks is necessary to understand the behavior of FPGA device and its mapped circuit under transistor aging. The fact that FPGAs allow modifications to the mapped function of LUT through reconfiguration, which can be partial or full, online or offline [48, 49], necessitates an extended investigation for the aging effect under different LUT configurations.

In this chapter, the effect of BTI-induced aging (both NBTI and PBTI) is investigated on three different LUT structures using accurate SPICE simulations. The first considered LUT structure is based on typical *Complementary Metal-Oxide-Semiconductor* (CMOS) logic gates that form together a multiplexer. The second one is based on a set of pass transistors (this one has the smallest area among the three structures) and the last one is based on a set of transmission gates (this one has the smallest delay among the three structures).

Two-input LUTs are investigated at the beginning, since larger LUTs are built from 2-input LUTs. The aging-induced degradation of the three structures using is obtained accurate SPICE simulations. For each structure, the effect of input signal probabilities (usage), LUT configuration, and FPGA reuse scenarios are investigated. This choice of small LUTs allows us to comprehensively investigate all possible scenarios for conclusive analysis, which would be impossible for larger LUTs. Afterwards, logic simulation is used to extend the results of 2-input LUTs to 4-input LUTs to analyze the aging effects in multi-level LUTs as used in modern FPGAs (such as 5-input and 6-input LUTs). For the sake of simplicity, results for 4-input LUTs are presented. However, the overall methodology of this analysis as well as the major conclusions are still hold for larger multi-level LUTs.

The main goals of this investigation are summarized in the following:

- Identify the best LUT structure in terms of aging.
- Determine best run-time LUT configuration for minimum aging effect.
- Determine best run-time LUT configuration in terms of aging if the reconfiguration history (i.e., stress history) of the LUT is known.
- Determine the best configuration for the unused LUTs for minimum aging effect.
- Find the role of usage on the amount of aging-induced degradation.

It is found that different parameters such as LUT structure, the current configuration, the reconfiguration history, and the inputs signal probabilities, have considerable effect

on the NBTI and PBTI induced aging rate of the LUT. Furthermore, the structure with the smallest area among the three structures is the most susceptible to aging, particularly in reconfiguration cases. Finally, it is found that, "all-zero" configuration, which is commonly used in the bitstream for the unused LUTs [50], is not always the best choice in terms of aging degradation.

The results of the investigation in this chapter are exploited in Chapter 10 for mitigating the aging effects in LUTs.

The rest of this chapter is organized as follows. Section 3.2 reviews the related work. Afterwards, Section 3.3 introduces the three different LUT structures under investigation. The evaluation methodology is presented in Section 3.4. The results are described and analyzed in Section 3.5. Finally, Section 3.6 concludes this chapter.

3.2. Related Work

There has been previous work on analyzing the effect of transistor aging in FPGAs [51, 52], where only the NBTI effect is investigated. However, to comprehensively investigate the aging, PBTI has to be also considered. Additionally, "pass transistor"-based LUT is the only structure that the authors considered in their investigations, although there are other LUT structures used in FPGA (such as transmission-gate-based LUT). Moreover, the authors assumed that the input signal probabilities were fixed to 50%, which means that the effect of signal probabilities, i.e. usage, was not considered. However, the input signal probabilities have a direct effect on BTI-induced aging. Another study, introduced in [53], describes exposing real FPGA chips to stress conditions to emulate aging, and reports delays of both fresh and aged circuit for different benchmarks, with suggestions for countermeasures.

Recently and similar to the results of this chapter, the authors in [54] show that the "pass transistor"-based FPGA structures are worse than the "transmission gate"-based alternatives. Besides, the authors in [55] use the "pass transistor"-based LUT to build a model for accelerated healing from degradation, in which controlling the signal probabilities plays a main role.

3.3. LUT structures

This section briefly introduces the three different LUT structures that are investigated in this work: 1) *Logic Gate* (LG)-based structure, 2) *Pass Transistor* (PT)-based structure and 3) *Transmission Gate* (TG)-based structure. For each structure, the implementation methodology of 2-input LUTs and 4-input LUTs are described. However, the same implementation principles are also applicable for larger multi-level LUTs (5 and 6-inputs).

3.3.1. Two-input LUT structures

For the first step, the implementation of 2-input LUTs are introduced for different structures. Comprehensive investigation of these small LUTs, helps us to understand the behav-

ior of larger LUTs since larger LUTs are made by 2-input LUTs as their building blocks.

3.3.1.1. Logic Gate (LG)-based structure

The most straight forward structure for LUT is based on traditional CMOS gates as shown in Figure 3.1. This structure consists of (see Figure 3.1(a)):

- Two inputs (In0 and In1) to select the required configuration SRAM cell
- Four configuration lines (C0-C3) that come from the SRAM cells
- Three two input multiplexers

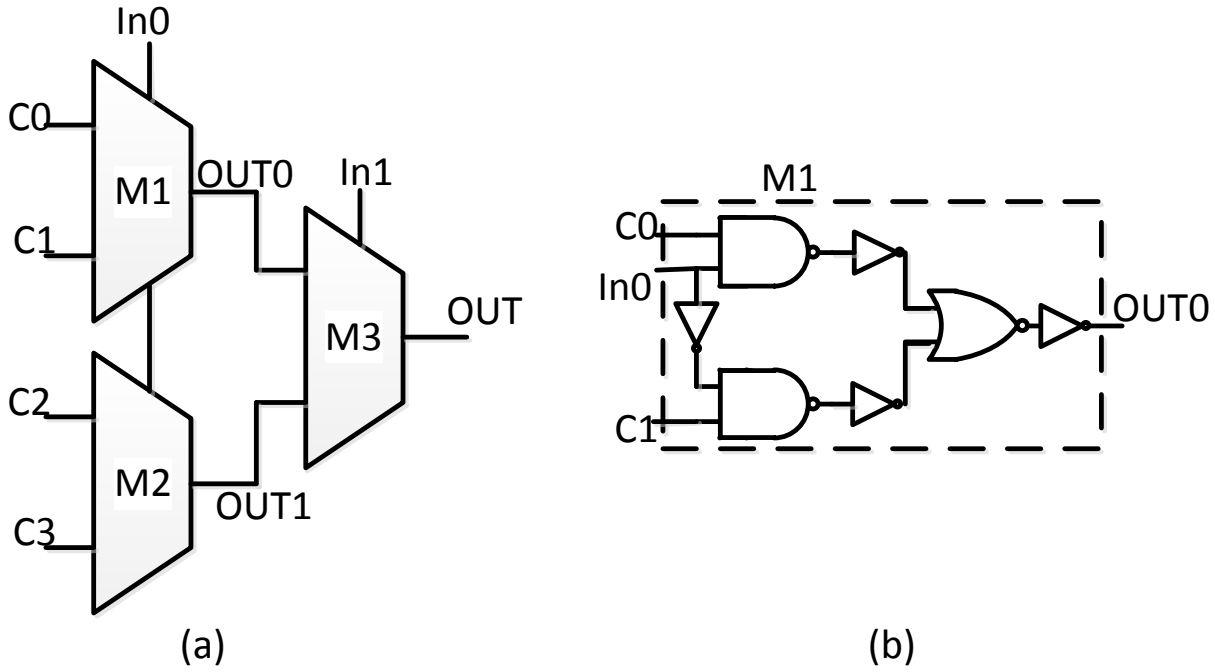


Figure 3.1.: LG-based structure of two input LUT

Each of the multiplexers consists of two NAND gates, four inverters, and one NOR gates, as shown in Figure 3.1(b). The main disadvantage of this structure is the large number of transistors. However, this simple LG structure is a good reference to compare with other structures in terms of delay, area, power and aging.

3.3.1.2. Pass Transistor (PT)-based structure

As shown in Figure 3.2, this structure consists of [56] :

- Two inputs (In0 and In1) to select the required pass transistor

- Four configurations (C0-C3) that come from the SRAM cells
- Six pass transistors used to form the multiplexer
- A half-latch consists of an inverter and a keeper PMOS transistor

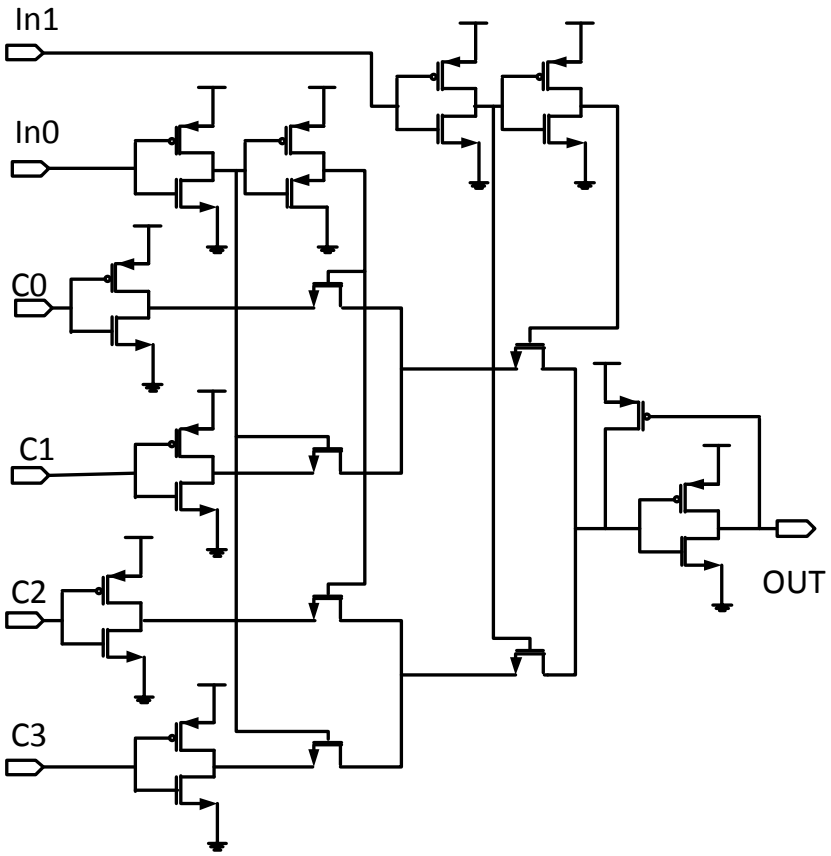


Figure 3.2.: PT-based structure of two input LUT

The main feature of this circuit is that NMOS pass transistors are used to form the multiplexer. Since NMOS pass transistors are smaller than PMOS transistors (designed to operate under the same conditions), this structure has the smallest area among the three structures. The half-latch is needed to improve the high-level logic at the output. This is because the NMOS transistor passes a strong zero but a weak one (i.e., less than V_{dd}). However, as the difference between V_{dd} and V_{th} decreases over time (as a result of both technology downscaling and aging), this method is no longer suitable.

3.3.1.3. Transmission Gate (TG)-based structure

As mentioned in the previous section, the PT-based LUT is not reliable at low voltage levels. Therefore, a transmission-gate-based LUT has been designed to operate reliably

at low voltage levels, but at the expense of higher transistor count [56]. In this structure, instead of unpaired NMOS pass transistor, transmission gate (a pair of parallel NMOS and PMOS transistors) is used to select one memory cell. This structure consists of (see Figure 3.3):

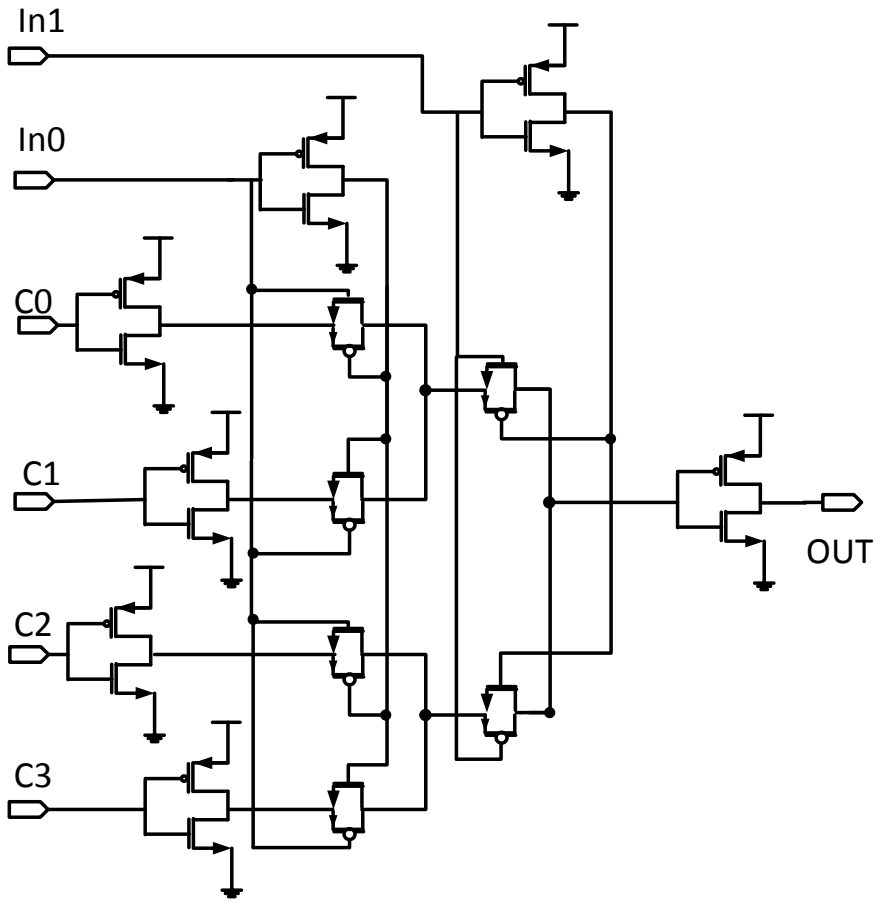


Figure 3.3.: TG-based structure of two input LUT

- Two inputs (In0 and In1) to select the corresponding transmission gates
- Four configurations (C0-C3) that come from the SRAM cells
- Six transmission gates used to form the multiplexer
- An inverter at the output

The TG-based structure has the minimum delay among the three structures. Also, unlike pass transistors, transmission gates can pass both logic zero and logic one with no degradation in the voltage level and thus it is suitable for low voltage levels. However, it requires more area in comparison with the PT structure (because of using a large PMOS

transistor in addition to the NMOS transistor to form the TG structure). This disadvantage can be partially relieved by removing the half-latch and any initialization circuitry, which are unnecessary in this structure.

3.3.2. Four-input LUT structures

Nowadays, state-of-the-art FPGAs contain 6-input LUTs. For the sake of simplicity, 4-input LUT is adopted as an example of larger LUTs since the same principles are also applied for 6-input LUTs (see for example [56] and [57]).

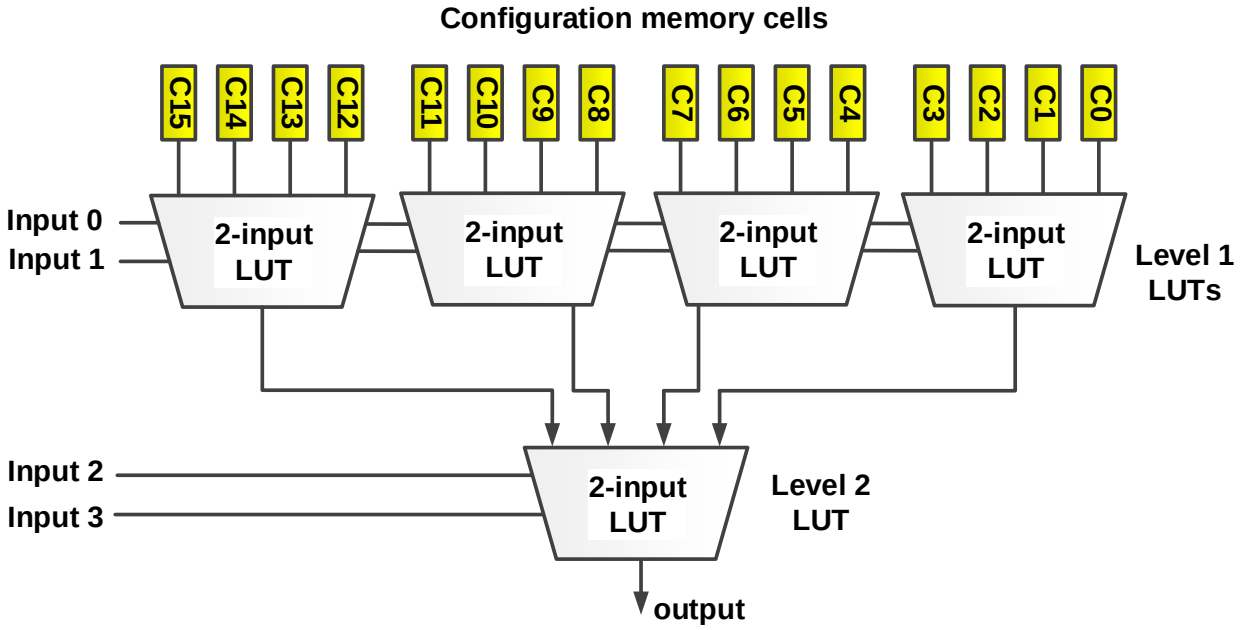


Figure 3.4.: Four-input LUT out of a set of five 2-input LUTs

Figure 3.4 shows how a 4-input LUT can be implemented out of a set of 2-input LUTs. As shown in this figure, the implementation consists of five 2-input LUTs in two levels. The outputs of the first-level LUTs are fed to the configuration lines of the second-level LUT. This formation is applied for each of the three structure described in the previous subsection to build the 4-input LUT version out of each of them. To build larger LUTs (e.g. 6-input LUT), more levels of this tree structure can be added using the same principle.

3.4. Evaluation methodology

The proposed evaluation methodology focuses firstly on 2-input LUTs and then it is extended to include 4-input LUTs as well.

3.4.1. Two-input LUTs

First, each LUT implementation, introduced in section 3.3, is converted to H-SPICE netlist in order to execute the desired simulations. H-SPICE simulator is then used to calculate the rise and fall delays between the *inputs* and the *output* of the LUTs. A 22-nm *Predictive Technology Model* (PTM) for metal-gate/high-κ CMOS [58] is used to simulate all the three implementations.

In order to investigate the effects of NBTI and PBTI on each LUT implementation introduced in Section 3.3, the accurate ΔV_{th} of each transistor in the implementation has to be calculated. As discussed in Chapter 2, the overall BTI effect on V_{th} over time can be calculated using Equation (2.2). For the sake of simplicity, the temperature is considered as a constant in the simulation. This simplifies Equation (2.2) as follows:

$$\Delta V_{th} = A \times Y^n \times t^n \tag{3.1}$$

where A is a technology dependent factor, which is a function of temperature. The rest of the parameters are the same as in Equation (2.2).

The duty cycle (Y), which is the time ratio between the stress time to the total time, can be represented by the *signal probability* (SP) of the gate input of the transistor. The SP is the time ratio in which the signal is 1 to the total time. Particularly, $Y = 1 - SP$ for PMOS transistors under NBTI and $Y = SP$ for NMOS transistors under PBTI. Moreover, A and n can be different for NBTI and PBTI for a given technology. Consequently, to calculate the ΔV_{th} for each transistor based on Equation 3.1, the duty cycle for each transistor (which is a function of the SPs of the internal nodes) in each LUT implementation must be first found.

In general, the 2-input LUT has 4 configuration lines (C0 - C3) to map the required function, and 2 input lines (In0 and In1) that are used as (primary) inputs for the mapped function. The configuration lines are fixed (e.g. fixed during the operation of the circuit unless a reconfiguration is done). However, the input lines change according to the operation of the circuit. In order to determine the behavior at these inputs, SPs are used. The signal probabilities of In0 and In1 are denoted by SP0 and SP1 respectively. Then the SPs of the internal nodes are obtained as a function of both the SPs of the inputs (SP0 and SP1) and the configurations (C0 - C3). The SPs of the internal nodes for LG-based, PT-based and TG-based LUTs are depicted in Figures 3.5, 3.6 and 3.7, respectively.

To find the duty cycle of each transistor, the SPs of internal nodes are used. For example in Figure 3.6, transistor M1 is under stress whenever $V_g(M1) = V_{dd}$ and $V_s(M1) = 0$, the duty cycle of transistor M1 can be calculated as $DC_{M1} = SP0 \times C0$, where DC_{M1} is the duty cycle of transistor M1. The duty cycle for other transistors can be calculated in a similar way. Based on the aforementioned observations, different SPs of inputs and different configurations result in different SPs of the internal nodes, different duty cycles of the transistors, and consequently, different NBTI and PBTI induced aging.

To cover all the effective parameters that have influence on NBTI and PBTI, all possible configurations (2^4 for the 2-inputs LUT) are investigated, and for each configuration, the SPs of the inputs (SP0 and SP1) are independently swept from 0% to 100% with a step of 10%. This makes the total number of required simulation cases for each LUT implementation equal to $2^4 * 11^2 = 1936$. Afterwards, for each case, the duty cycle of individual tran-

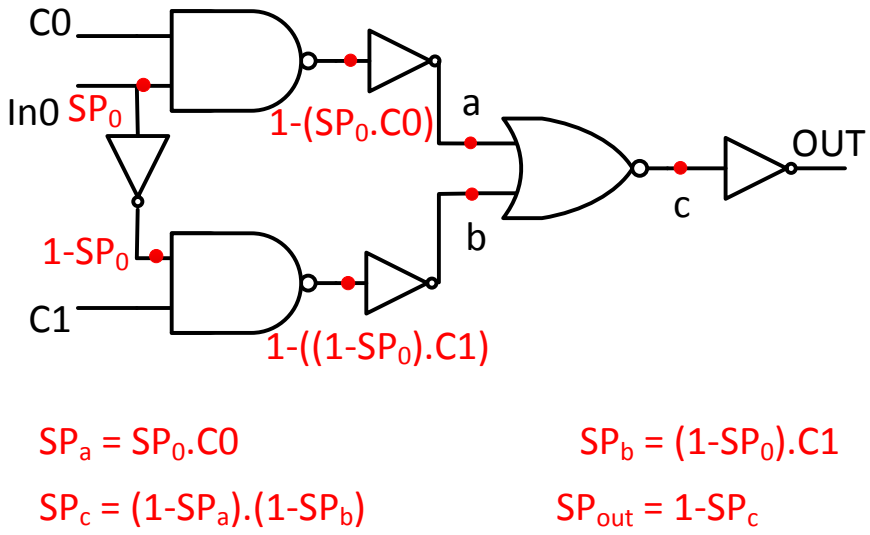


Figure 3.5.: SPs of the internal nodes of LG-based 2-input LUT

sistors are calculated and the accurate BTI-induced ΔV_{th} is found using Equation (3.1). For calculating the threshold voltage based on Equation (3.1), the H-diffusion type ($n = 1/4$) is assumed. Moreover, the constant A is chosen in a way that the NBTI/PBTI induced $\Delta delay/delay$ (for both rise and fall time) of a normal inverter with $SP = 50\%$ will be 10% in $t = 3$ years.

The calculated ΔV_{th} for each transistor is given to the H-SPICE netlist using the `DELVTO` parameter available in H-SPICE. The `DELVTO` parameter simply shifts the threshold value of each transistor in the netlist.

3.4.2. Four-input LUTs

As shown in Figure 3.4, a 4-input LUT has 16 configuration lines (C0 - C15) and 4 input lines (In0 - In3). In order to comprehensively investigate the effects of NBTI and PBTI on 4-input LUTs with the same resolution for SP of primary inputs, which is used for 2-input LUTs (0% to 100% with a step of 10%), a total number of ($n = 2^{16} \times 11^4$) different HSPICE simulations have to be run. The huge number of simulations makes the investigation very difficult for this type of LUTs. Therefore, in order to make the investigation possible for 4-input LUTs (i.e., reduce required runtime), in this section a method to obtain the BTI-induced delay degradation of 4-input LUTs based on the results of 2-input LUTs is proposed. It should be noted that since larger LUTs (such as 6-input LUTs) are also built by 2-input LUTs, the same methodology can be used to obtain the BTI-induced delay degradation of larger LUTs.

This method is based on the following facts (see Figure 3.8): i) The 4-input LUT consists of two levels of 2-input LUTs (level-1 and level-2). ii) The delay of level-1 LUTs is from their inputs (In0 and In1) to their outputs, because the configuration lines are fixed inputs from SRAM cells. However, iii) for level-2, it can be assumed that its inputs (In2 and In3) are already there by the time the outputs of level-1 LUTs are arriving. This makes the delay of level-2 LUT to be from its configuration lines to the output. iv) Since the paths from

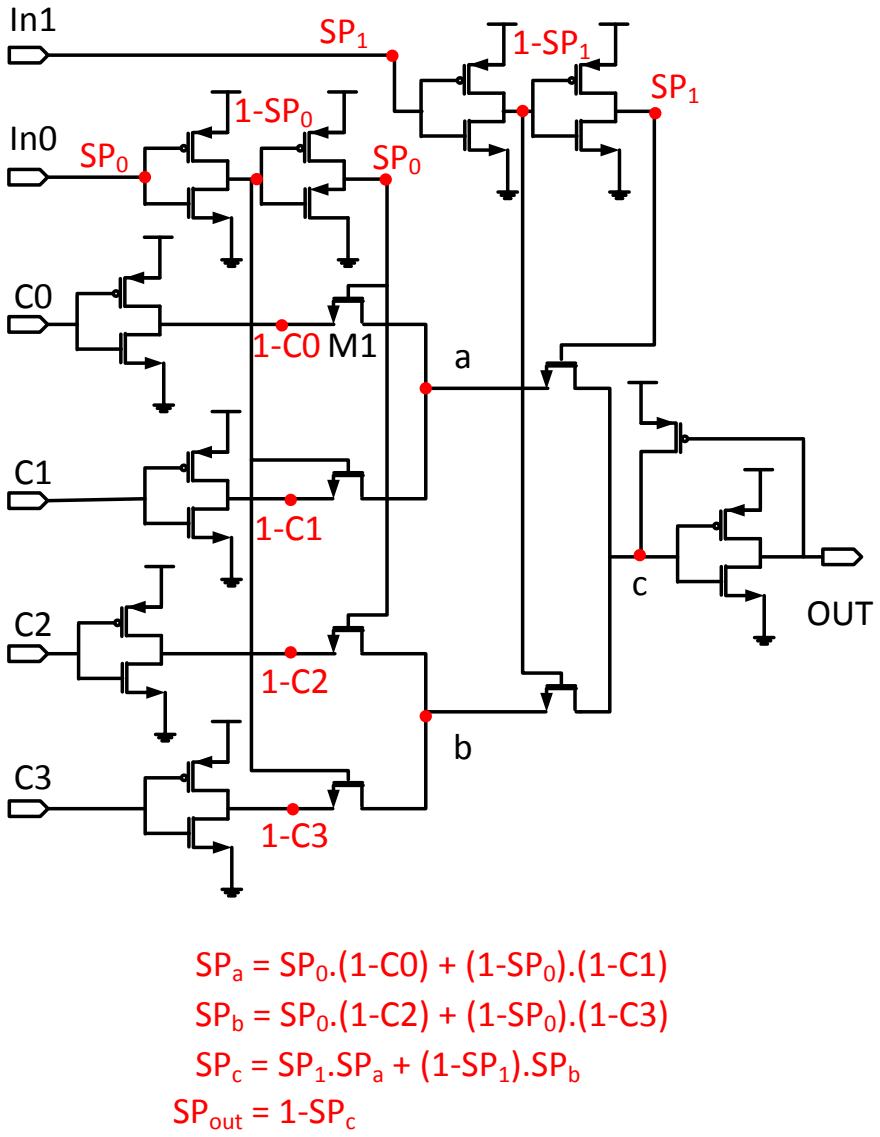


Figure 3.6.: SPs of the internal nodes of PT-based 2-input LUT

the configuration lines to the output at level-2 are symmetric, their delays are identical, and thus the load on the output drivers from level-1 is the same for any path on level-2. These facts allow us to write the total delay of the 4-input LUT as a summation of the independent delays of level-1 and level-2 as obtained by the following equation:

$$\begin{aligned} D_{4inp} &= D_{level1} + D_{level2} \\ \text{and} \quad D_{level1} &= \max_{LUT(i) \in level1} (D_{LUT(i)}) \end{aligned} \tag{3.2}$$

where D_{4inp} is the delay of 4-input LUT. D_{level1} is the maximum delay among all 2-input LUTs in level-1. The delays of 2-input LUTs in level-1 are the ones which are obtained by the methodology introduced in the previous subsection. D_{level2} is the delay of 2-input LUT in level-2, which is different from the first level delays as explained before.

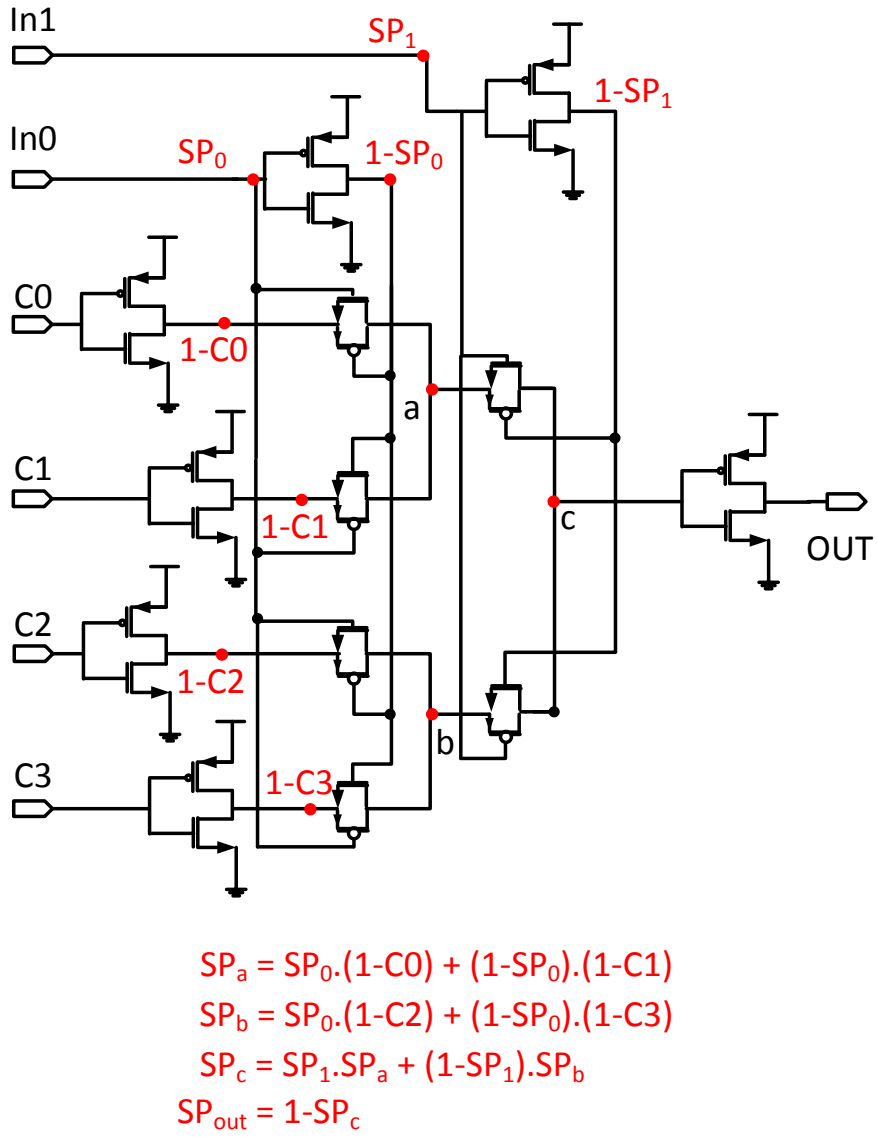


Figure 3.7.: SPs of the internal nodes of TG-based 2-input LUT

Therefore, there is a need to run additional SPICE simulations to obtain the BTI-induced delay degradation of level-2 LUT. For that, a separate BTI analysis for the SPs of each different configuration line at level-2 (11 different signal probabilities from 0% to 100% with a step of 10% for each of the 4 configuration lines = 11^4) and each input SPs (11 different signal probabilities from 0% to 100% for each of the 2 inputs = 11^2) is generated. This makes the total number of additional SPICE simulations equal to 11^6. It should be noted that a tradeoff between the number of SPICE simulations and the accuracy has to be made. For example, the number of SPICE simulations can be reduced using less accurate SPs for configuration lines (e.g. with a step of 20% instead of 10%).

After obtaining the BTI-induced delay results of level-2 LUT, Equation (3.2) is used to obtain the BTI-induced delay of the 4-input LUT. For each set of specific configuration and input SPs, the D_{level1} is obtained by the methodology introduced in the previous subsection. Besides, the SPs are obtained for the output of each 2-input LUT in level-1. Since

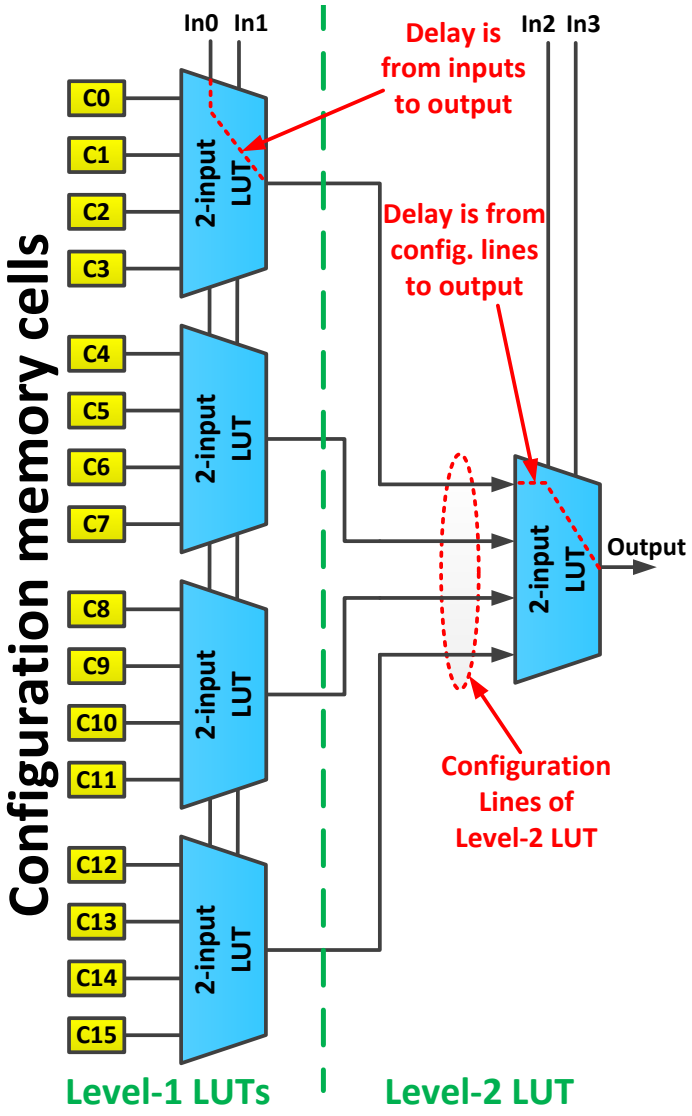


Figure 3.8.: Delay definitions of first and second level LUTs

the configuration lines of level-2 LUT are connected to the outputs of level-1 LUTs, the obtained SPs as well as SPs of third and fourth inputs (In2 and In3) can be used to obtain the BTI-induced delay degradation of second level LUT (D_{level2}). Interpolation is used to convert the SPs obtained from the outputs of LUTs at level-1 to the exact SPs (0% - 100% with a step of 10%) at the configuration lines input of the level-2 LUT.

3.5. Experimental results and analysis

The analysis presented in this section is divided into two main parts. In the first part, the aging of 2-input LUTs is studied, then in the second part the effect of aging on 4-input LUTs is investigated. Two aging effects are considered separately for all the simulations: i) NBTI effect alone, because before the introduction of high-κ/metal gate transistors, the PBTI effect was negligible, and ii) The combined effect of both NBTI and PBTI, to con-

sider the LUTs fabricated using the new high-κ/metal gate transistor technology, in which both effects are comparatively important. A 22 nm technology library is used for SPICE simulations as described in Section 3.4. The BTI related coefficients are set in a way that the delay of an inverter increases by 10% in 3 years for both NBTI and PBTI phenomena. Please note that in all the following results only the maximum delay between rise and fall delays are considered.

3.5.1. Two-input LUTs results

The first focus of this section is on studying the LUT aging under different configurations with respect to different input SPs. Then, the second subsection focuses on studying the effect of "previous" configuration (The configuration that was previously loaded on the LUT in "reconfigurable" applications, or the configuration which is loaded when the LUT is unused) on the aging of the LUT with respect to its "current" configuration (the configuration which is loaded when the LUT is used).

3.5.1.1. Aging Effects for Different Configurations

All possible configurations with all possible input SPs are examined in order to find their influence on the aging of the three LUT structures. Figure 3.9 shows the BTI-induced aging on the three LUT structures. It consists of two parts: on the left side; the aging effect is due to NBTI-alone and on the right side; the aging effect is due to the combined effect of both NBTI and PBTI. The y-axis is the normalized LUT delay increase. The x-axis reflects the 16 possible configurations of the 2-input LUT. The effect of SPs can be seen with the vertical lines (range) for each configuration. The average over all combinations of input SPs for each configuration is represented by the x sign on the lines as shown in the figure. Since the relation between SPs and the NBTI/PBTI effects is not linear, the average value is not always at the middle of the range.

It can be seen in Figure 3.9 that the post-aging delay depends on the configuration, signal probability, and the structure. Therefore, all these three aspects must be considered for accurate aging modeling of LUTs. Also, the extent of the effect of each of these three factors (LUT structure, configuration, and input SPs) on aging-induced delay heavily depends on the other two parameters.

The other interesting observation is that in a given LUT structure, different configurations have different BTI sensitivity. In the delay models used for LUTs, it is widely accepted that the propagation delay of an LUT is independent of the mapped function (configuration) [59, 60]. However, these results suggest that the post-aging delay actually depends on the mapped function (configuration) to the LUT.

In terms of NBTI-induced post-aging Δdelays, the three structures are comparable. However, there are minor differences that make the LG-based structure the best choice.

In PT-based structure, NBTI-induced Δdelays are negative for some SPs (meaning that the LUT becomes faster due to NBTI effect) which can be explained by the following: In Section 3.3, it is mentioned that NMOS transistor passes normally a weak one, and therefore a keeper is needed to improve the high level logic at node c and low level logic

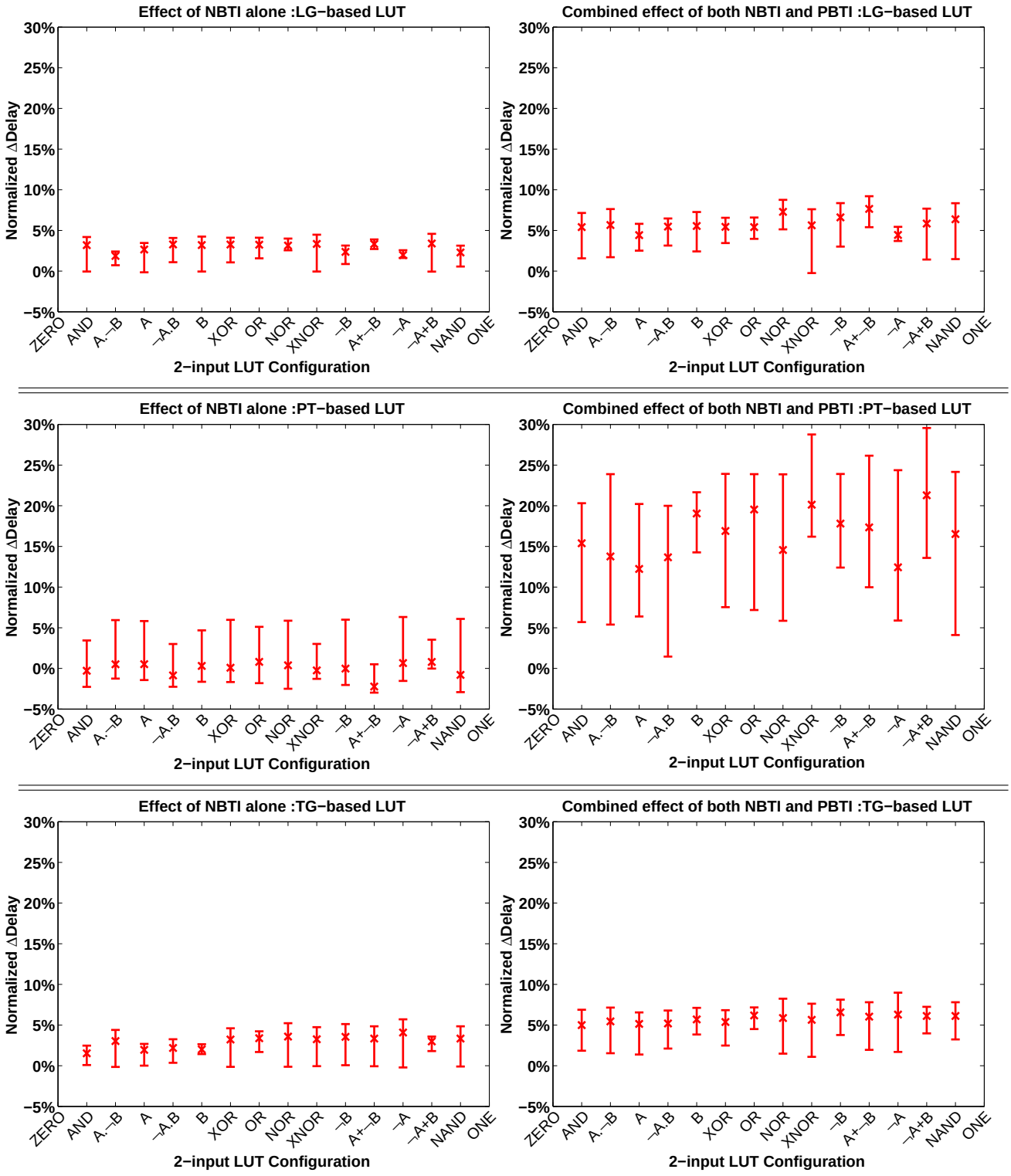
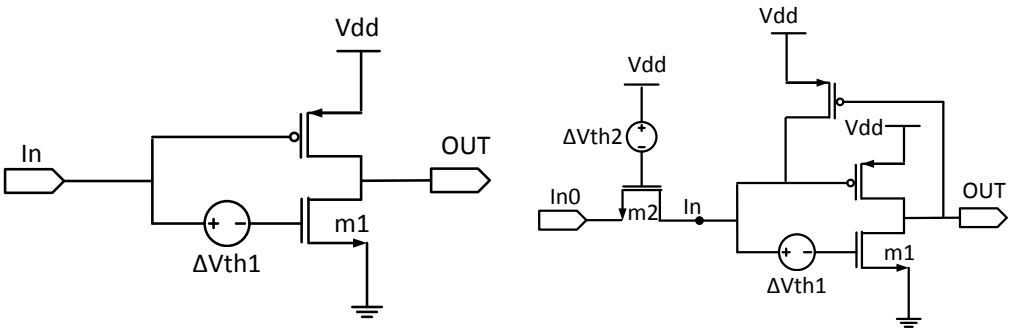


Figure 3.9.: The BTI effect of different configurations with different input SPs on the three LUT structures (2-input LUTs)

at node OUT (see Figure 3.6) in PT-based LUT. However, this keeper makes the rising transition at the output slower. Moreover, the NBTI effect increases the keeper's threshold

voltage and as a result it becomes weaker due to NBTI. Putting altogether, NBTI may result in faster rising transition at the output by making the keeper weaker. This effect can be seen in Figure 3.9 (negative $\Delta delay$).



(a) Circuit1 (b) Circiut2

Figure 3.10.: A normal inverter (Circuit1) and a pass transistor connected to a half latch (Circuit2)

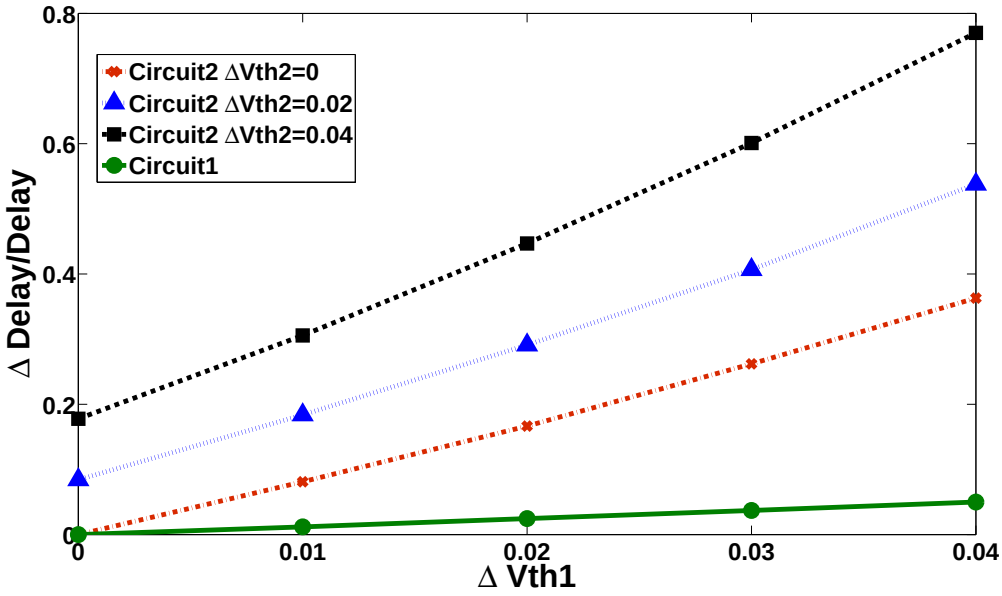


Figure 3.11.: Delay sensitivity of the two circuits in Figure 3.10 to threshold voltage of NMOS transistor

In terms of BTI-induced post-aging Δdelays (i.e., the combined effect of both NBTI and PBTI), the LG-based and TG-based structures are comparable, while the PT-based

structure is the worst one. The larger range of Δdelay in the PT-based structure is mainly due to the high impact of PBTI on the fall time. To clarify this effect, two circuits shown in Figure 3.10 are considered. The first circuit consists of a normal inverter, and the second one consists of a half latch with a pass transistor connected to its input. Figure 3.11 shows the sensitivity of fall time (from node In to node OUT) to the threshold voltage change of transistor $m1$. As shown in this figure, the normalized Δdelay of circuit2 is much higher than circuit1 for the same amount of ΔV_{th} of transistor $m1$. Moreover, the normalized Δdelay of circuit2 increases when the threshold voltage of pass transistor increases. As a result, since PBTI increases the threshold voltage of both the NMOS pass transistor and the NMOS transistor in the pull-down network of the half-latch, it has relatively high impact on PT-based structure compared to the LG-based structure. It should be noted that the effect of PBTI is less in TG-based structure compared to PT-based structure because of two reasons:

- A normal inverter, which is, as explained in the previous paragraph, less sensitive to PBTI, is used at the output of the TG-based structure instead of the half-latch in the PT-based structure
- The structure of the transmission gate, in which a PMOS transistor is parallel to an NMOS transistor, alleviates the effect of PBTI on the NMOS transistor.

Table 3.1 shows a ranking of the five best configurations that have the minimum aging (NBTI and NBTI/PBTI) in each LUT structure. In addition, for each configuration the *delay degradation reduction* (DR) compared with worst-case configuration is shown. The DR is calculated as follows:

$$DR = \frac{\max_i(Conf_i^{max}) - \min_i(Conf_i^{max})}{\max_i(Conf_i^{max})} \tag{3.3}$$

where $Conf_i^{max}$ is the maximum value of normalized Δdelay for all possible SPs in a certain configuration (i). The information in this table can be used for aging-aware logic mapping.

These results show that when only NBTI is considered, the range of aging-induced delay-increase for different configurations is quite high (up to 92% for PT-based structure). This range is reduced when both NBTI and PBTI are considered.

3.5.1.2. Aging Effects Considering Previous Configuration

Here the effect of the "previous" configuration on the aging of the LUT when it is loaded with the "current" configuration is studied. In this scenario, the LUT is already aged based on a previous configuration (for a long period of time) and the delay degradation of the newly loaded configuration due to transistor aging under the previous configuration is investigated. The purpose of this study is to find out:

- If a pre-knowledge about the user configuration is available, then the suitable configuration to be loaded on unused LUTs can be determined.
- In reconfiguration systems in which the set of possible configurations are known, an accurate delay of the newly loaded configuration can be obtained and optimized,

Table 3.1.: Best NBTI and PBTI induced configurations for the three 2-input LUT structures and the improvement of $\Delta delay$ in comparison to the worst case configuration

	LG		**PT**		**TG**	
Ranking	**NBTI alone**	**NBTI and PBTI**	**NBTI alone**	**NBTI and PBTI**	**NBTI alone**	**NBTI and PBTI**
1	$A.\neg B$ (47.51%)	$\neg A$ (40.92%)	$A+\neg B$ (92.00%)	$\neg A.B$ (32.37%)	AND (56.58%)	A (27.02%)
2	$\neg A$ (43.93%)	A (36.82%)	$\neg A.B$ (52.49%)	A (31.60%)	B (53.64%)	$\neg A.B$ (24.52%)
3	$NAND$ (31.62%)	$\neg A.B$ (29.71%)	$XNOR$ (52.30%)	AND (31.29%)	A (52.92%)	XOR (23.99%)
4	$\neg B$ (31.44%)	XOR (28.79%)	AND (45.59%)	B (26.74%)	$\neg A.B$ (42.83%)	AND (23.45%)
5	A (24.61%)	OR (28.42%)	$\neg A+B$ (44.19%)	NOR (19.25%)	$\neg A+B$ (37.05%)	B (20.81%)

given the LUT configurations, usage (SPs), and runtime of the previous configurations.

- What is the best configuration, in terms of minimum aging delay degradation, to be loaded in unused LUTs, when the user configuration is unknown.

In order to perform this analysis, all pairs of configurations (as the "previous" configuration and "current" configuration) have been simulated. The best "previous" configuration for each "current" configuration is reported in Table 3.2. For instance, if the "current" configuration is *NAND*, the best "previous" configurations for the NBTI-induced aging are *ONE*, *XNOR* and *ZERO* for LG, PT, and TG-based structures respectively. For the BTI-induced aging the best "previous" configuration is *ONE* for both LG and PT-based structures, and $\neg A.B$ for TG-based structure.

These results show that the best "previous" configuration depends on the "current" configuration and the particular structure. Moreover, the choice of all-zero as the standby configuration, as dominantly used to fill unused LUTs in configuration bitstream in the FPGAs, may result in maximum delay degradation for some user configurations.

To have an overview of the effect of the "previous" configurations on the BTI-induced aging of the "current" configurations, Figure 3.12 shows the range of normalized post aging $\Delta delay$ for different "previous" configurations. Unlike Figure 3.9, the range in Figure 3.12 represents the effect of different SPs as well as different "current" configurations. It can be seen from the ranges in Figure 3.12 that if the FPGA is reused, the PT-based structure will have the worst BTI-induced post aging delay in comparison with the two other structures. This is mainly due to the high sensitivity of PT-based structure to the PBTI effect as described before.

When the user configuration is unknown, the "previous" configuration which has in average the minimum post-aging Δdelay among all "current" configurations is chosen according to Equation (3.4).

$$min_i(mean_j(Conf^{mean}_{current(j)}|Conf^{max}_{previous(i)})) \tag{3.4}$$

where $Conf^{mean}_{current(j)}$ is the mean value of normalized Δdelay for all possible SPs in a certain "current" configuration (j) and $Conf^{max}_{previous(i)}$ is the maximum value of normalized Δdelay for all possible SPs in a certain "previous" configuration (i). The results for the three LUT structures are shown in Table 3.3. Also, the percentages of DR compared to the worst case "previous" configuration are shown. Note that in current FPGAs, unused LUTs in the configuration bitstream are filled with all-zero configuration. However, these results show that for both the LG-based and the PT-based structures, all-zero is not the best choice, and even does not lie in the top-five rank for the NBTI effect. Furthermore, PT-based structure is the most sensitive to the choice of "previous" configuration, followed by TG and LG.

3.5.2. Four-input LUTs results

Similar to 2-input LUTs, the effect of both NBTI and PBTI on each possible configuration of the 4-input LUT in each structure is investigated. Table 3.4 shows a ranking of

Table 3.2.: Best NBTI and PBTI induced "previous" configuration for different "current" configuration in 2-input LUTs

	C-C	*AND*	$A.\neg B$	*A*	$\neg A.B$	*B*	*XOR*	*OR*	*NOR*	*XNOR*	$\neg B$	$A+\neg B$	$\neg A$	$\neg A+B$	*NAND*
LG	**NBTI P-C**	*ONE*	*ONE*	$\neg A$	*ONE*	*ONE*	*ONE*	*ONE*	*B*	*ONE*	*ONE*	$\neg A.B$	*A*	*ONE*	*ONE*
	BTI P-C	*ONE*	*ONE*	$\neg A$	*ONE*	*ONE*	*ONE*	*ONE*	*OR*	*ONE*	*ONE*	*ONE*	*A*	*ONE*	*ONE*
PT	**NBTI P-C**	*ZERO*	*XNOR*	*XNOR*	*ZERO*	*XNOR*	*XNOR*	*XNOR*	*XNOR*	*XNOR*	*XNOR*	*XNOR*	*XNOR*	*XNOR*	*XNOR*
	BTI P-C	*A*	*OR*	$A+\neg B$	$\neg A$	*ONE*	*ONE*	*ONE*	$\neg A+B$	*ONE*	*ONE*	*ONE*	*ONE*	*ONE*	*ONE*
TG	**NBTI P-C**	*ZERO*	*ZERO*	*ZERO*	*ZERO*	$\neg B$	*ZERO*	*ONE*	*ZERO*	*ZERO*	*ZERO*	$A.\neg B$	*ZERO*	*ONE*	*ZERO*
	BTI P-C	*ZERO*	*ZERO*	*ZERO*	*ZERO*	*NOR*	*ZERO*	$\neg B$	*ZERO*	*ZERO*	*ZERO*	*AND*	*ZERO*	*NAND*	$\neg A.B$

C-C: Current Configuration P-C: Previous Configuration

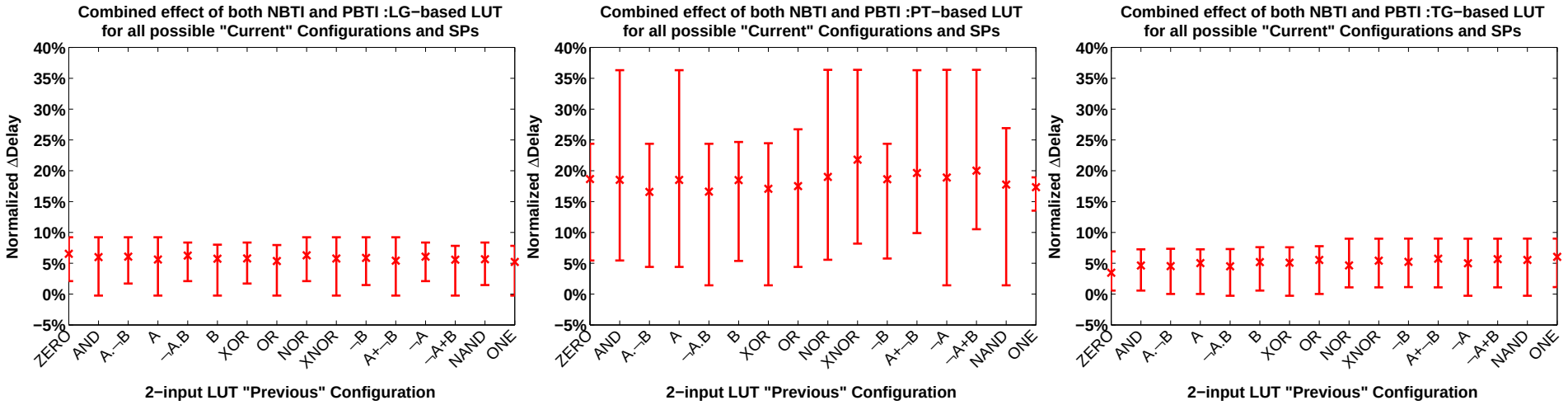


Figure 3.12.: BTI effects of different "current" configurations with different input SPs and different "previous" configuration for three 2-input LUT structures.

Table 3.3.: Best NBTI and PBTI induced "previous" configuration for the three structures and the improvement of $\Delta delay$ in comparison to the worst case when the "current" configuration is unknown (2-input LUTs)

	LG		**PT**		**TG**	
Ranking	**NBTI alone**	**NBTI and PBTI**	**NBTI alone**	**NBTI and PBTI**	**NBTI alone**	**NBTI and PBTI**
1	*ONE* (12.26%)	*ONE* (16.77%)	*XNOR* (45.57%)	*ONE* (46.84%)	*ZERO* (45.41%)	*ZERO* (41.03%)
2	*OR* (11.30%)	$A+\neg B$ (12.60%)	$A+\neg B$ (45.57%)	B (34.01%)	$\neg A.B$ (35.34%)	$\neg A.B$ (26.80%)
3	$A+\neg B$ (10.64%)	*OR* (12.20%)	$\neg A+B$ (45.57%)	*OR* (32.88%)	*AND* (35.19%)	$A.\neg B$ (25.17%)
4	A (10.22%)	*NAND* (10.12%)	*ONE* (45.57%)	$\neg A.B$ (32.88%)	B (26.35%)	*AND* (24.60%)
5	*NAND* (6.80%)	$\neg A+B$ (9.08%)	*AND* (4.58%)	*ZERO* (31.32%)	$A.\neg B$ (23.93%)	*NOR* (18.82%)

the top five configurations that have the minimum aging in comparison to the worst case configuration for each structure type. The 16-bit configurations are shown in hexadecimal format. In fact, due to the symmetry of the components of 4-input LUT (see Figure 3.4), there are several configurations that have the same NBTI/PBTI-induced delay degradation (see for example the column "NBTI alone" of both the PT and the TG structures). For that case, the first five configurations in sequence have been reported regardless of how many configuration share the same aging effect.

As the number of possible configurations and SPs for each LUT type is too large (2^{16} configurations $\times 11^4$ SPs), in Table 3.5 only the ranges of the delay degradation for each of the three LUT structures are reported. Also, in Figure 3.13, the probability density function of the normalized Δdelay is plotted for each aging effect (NBTI alone and both NBTI/PBTI) for all combinations of configurations and SPs. This is to show a summary of their total effect on aging. The normalized Δdelay is taken for the mean value over all SPs in each possible configuration. Several observations can be drawn from this figure:

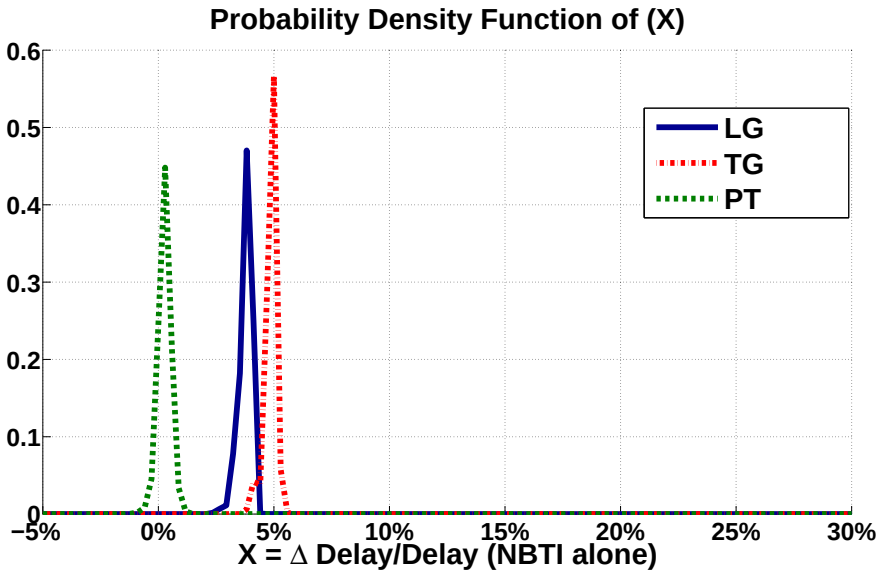
- The TG-based LUT is the worst among the three structures when the NBTI effect alone is considered.
- From the distribution of the three structures in Figure 3.13(a), it can be seen that the aging variation due to different configurations is comparable for the three structures in case of considering NBTI alone.
- In Figure 3.13(b), it can be seen that the PT-based LUT is much worse than the other two types in both degradation and variation when both NBTI and PBTI are considered together.

3.6. Summary

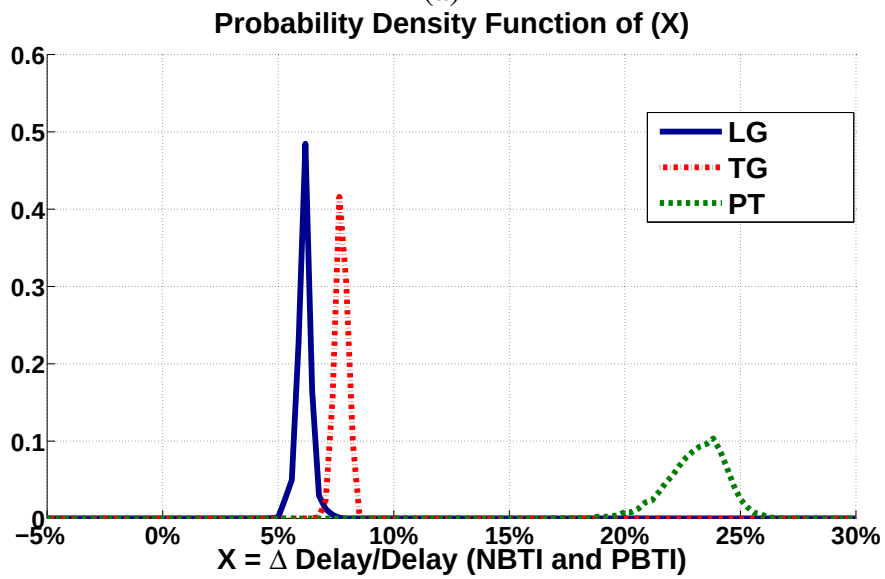
In this chapter, the effect of transistor aging, due to NBTI and PBTI, in look-up tables (LUTs) has been investigated by considering different structures through detailed SPICE simulations. The main conclusions of this analysis can be summarized as follows.

- The LUT structure, the current configuration, the previously run configuration, and the inputs signal probabilities are parameters that have considerable effect on the NBTI and PBTI induced aging of LUTs, and all of them must be considered for accurate delay degradation analysis.
- The PT-based structure (the one with least area overhead) is the worst choice for high-κ/metal gates technologies, because PBTI has a considerable effect on this structure.
- All-zero configuration in the bitstream for the unused LUTs is not the best choice in term of aging and may result in high delay degradation when the LUT is used in future. The best configuration for the unused LUTs depends on the runtime configuration and the LUT-structure.

The results of this analysis can be used for aging-aware logic mapping as well as reconfiguration scheme for FPGAs.



(a)



(b)

Figure 3.13.: Probability density function for the normalized Δdelay in the three 4-input LUT structures: (a) considering NBTI alone and (b) considering both NBTI and PBTI together

Table 3.4.: Best NBTI and PBTI induced configurations for the three 4-input LUT structures and the improvement of Δdelay in comparison to the worst case configuration

	LG		PT		TG	
Ranking	**NBTI alone**	**NBTI and PBTI**	**NBTI alone**	**NBTI and PBTI**	**NBTI alone**	**NBTI and PBTI**
1	F2FFh (46.95%)	FCFFh (33.21%)	B00Bh (63.95%)	FFFFh (68.40%)	DD1Dh (40.79%)	FD1Fh (24.51%)
2	2FFFh (46.35%)	FFFFh (32.81%)	B00Fh (63.95%)	B11Fh (52.68%)	DD1Fh (40.79%)	FF1Fh (24.51%)
3	02FFh (45.77%)	333Fh (32.31%)	B0BBh (63.95%)	F3FFh (52.67%)	DF1Dh (40.79%)	F1DFh (24.46%)
4	20FFh (45.77%)	33CFh (32.31%)	B0BFh (63.95%)	FF4Fh (52.44%)	DF1Fh (40.79%)	F1FFh (24.46%)
5	22FFh (45.77%)	33FFh (32.31%)	B0FBh (63.95%)	F4FFh (52.43%)	FD1Dh (40.79%)	F1FDh (24.41%)

Table 3.5.: Ranges of normalized Δdelay degradation for the three 4-input LUT structures

LG		PT		TG	
NBTI alone	**NBTI and PBTI**	**NBTI alone**	**NBTI and PBTI**	**NBTI alone**	**NBTI and PBTI**
-0.12% to 5.47%	0.37% to 8.52%	-2.37% to 5.43%	3.78% to 48.74%	1.81% to 8.02%	3.14% to 10.79%

Chapter 4.

Investigating The Degradation of Programmable Routing Resources

4.1. Introduction

In the previous chapter, the degradation of LUTs due to BTI effect is investigated. This chapter investigates the degradation of another basic element of FPGAs, which is the routing switch. As discussed in Chapter 2, the CLBs are connected together through programmable routing resources. These resources are designed to be very general in order to enable different routing possibilities among logic blocks. This results in large routing resources, which consequently consume most of the die area and power [61]. Additionally, in most cases, due to the large sets of switches that form a certain route, routing resources cause the dominant delay in the overall path delay [61]. For these reasons, studying the degradation of FPGA programmable routing resources is of great importance.

A routing switch, programmed through an SRAM cell, is the building block for the routing resources in FPGAs. For that reason, in this chapter, the effect of BTI-induced aging (both NBTI and PBTI) on four different implementations of programmable routing switches is investigated. For each implementation, detailed SPICE simulations using PTM [58] 32 nm technology library are done, considering a wide range of different parameters which affect the operation of the switch such as: load (wire length, number of cascaded switches and fan-out), input signal probability and supply voltage. The results show that these parameters affect the amount of degradation differently. Furthermore, the structures based on an NMOS transistor are the best in terms of NBTI while the transmission gate based structure is the best when both NBTI and PBTI are considered. In addition, contrary to typical CMOS logic structures, using a higher supply voltage causes less amount of degradation for the pass transistor based structures, which makes them benefit from both higher performance and less aging.

The rest of this chapter is organized as follows. Section 4.2 introduces the different routing switch implementations under investigation. Then, the evaluation methodology is presented in Section 4.3. Afterwards, the results are described and analyzed in Section 4.4. Finally, Section 4.5 points out the main conclusions of this investigation.

4.2. Programmable Routing Switches in FPGAs

As discussed in Chapter 2, the routing switches or *Programmable Interconnect Points* (PIPs) are used to define the desired routing among the logic blocks inside the FPGA. With the development of FPGAs, several designs and architectures have been proposed for the

routing elements, and various switch matrices appeared in state-of-the-art FPGA devices. However, the structure of a basic PIP did not change much. Here, four different implementations for routing switches are described, which are mainly used in different generations, categories, and variations of SRAM-based FPGAs. These implementations are the basis of the analysis in this chapter.

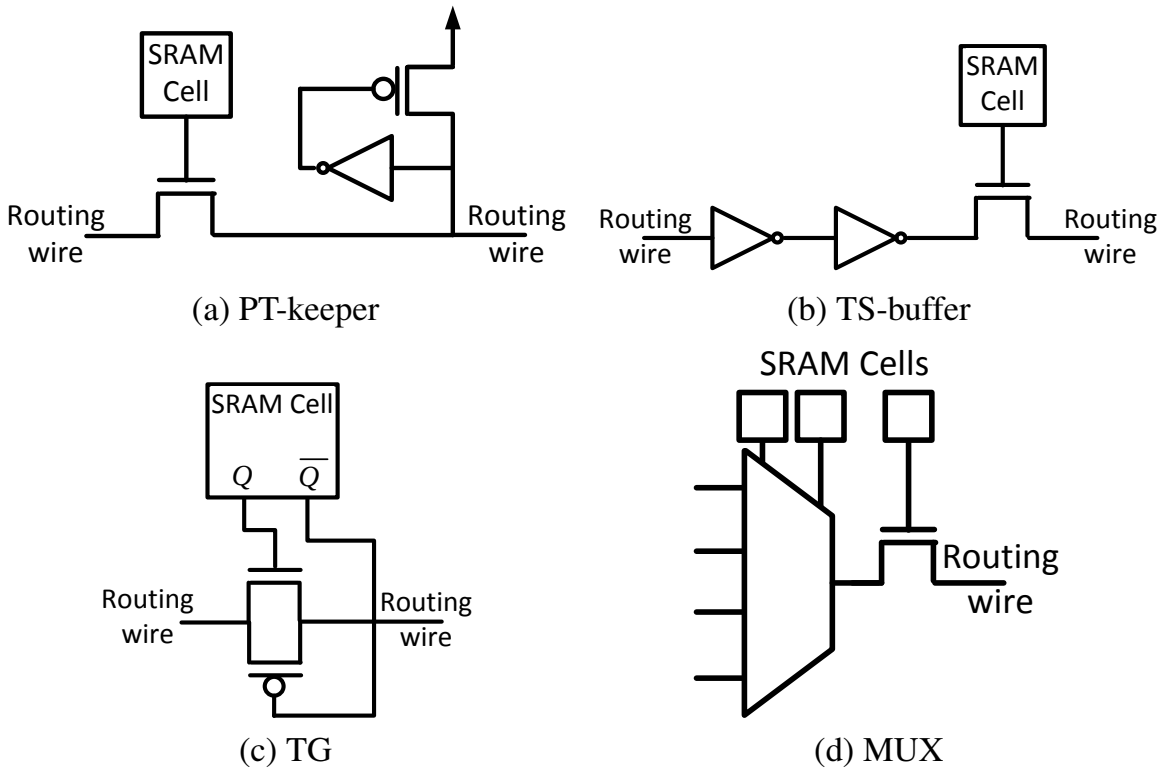


Figure 4.1.: Different routing switch structures.

4.2.1. Pass Transistor with keeper (PT-keeper)

A pass transistor, which consists of an NMOS transistor and a controlling SRAM cell, was used in the early FPGAs for routing switches, because it can be used as a bidirectional switch and needs only one SRAM cell to control it [62]. In addition, this type of routing switch has very small area. However, the delay of this type of switches grows quadratically when they are connected in series [62]. The other disadvantage of pass transistor switches is that they can only convey a *weak-1* ($V_{dd} - V_{th}$) instead of a *strong-1* (full V_{dd}). This increases the leakage current of the subsequent elements. With technology downscaling, the difference between V_{dd} and V_{th} decreases, and hence using this structure in newer technologies is more problematic.

To overcome the disadvantages of pass transistor structure, a keeper is added to the output of pass transistor (see Figure 4.1(a)) to pull a weak-1 into a strong-1 [62]. The

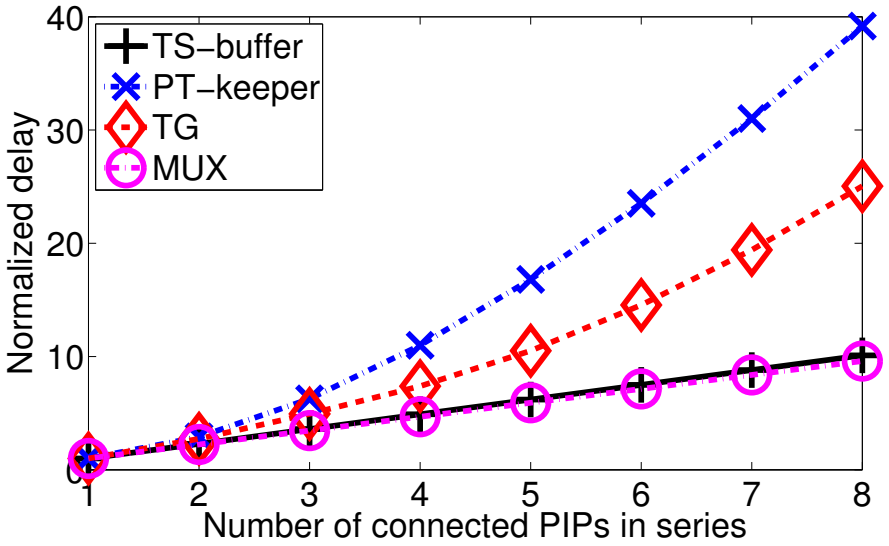


Figure 4.2.: Normalized Delay versus number of PIPs connected in series for different structures.

keeper consists of an inverter and a PMOS transistor. When the output of pass transistor is a weak-1, the PMOS transistor of the keeper switches on, and as a result, the output is pulled up to V_{dd}. Still, the delay of the PT-keeper increases quadratically with the number of cascaded switches (see Figure 4.2).

4.2.2. Tri-State buffer (TS-buffer)

Similar to the PT-keeper, a tri-state buffer consists of a pass transistor controlled by an SRAM cell (see Figure 4.1(b)). However, a buffer is used at its input to solve the problem of quadratic delay increase when multiple routing switches are connected in series. The buffer consists of two or more cascaded inverters. This type of routing switches needs more area in comparison to the pass transistors. However, it can provide a linear delay dependency to the number of cascaded routing switches (see Figure 4.2). That is why they are used in larger FPGAs, where many switches are connected in series.

4.2.3. Transmission Gate (TG)

The transmission gate based routing switch consists of a parallel structure of an NMOS and a PMOS transistor [63]. Because an SRAM cell has both Q and $\neg Q$, a single SRAM cell can be used for controlling the TG switch, where Q is connected to the gate of NMOS and $\neg Q$ is connected to the gate of PMOS (see Figure 4.1(c)). Although this structure has a larger area in comparison to the simple pass transistor structure, it can pass both strong-1 and strong-0. However, similar to PT-keeper structure, the delay of transmission gate switches increases quadratically when they are connected in series (see Figure 4.2).

4.2.4. Multiplexer (MUX)

The multiplexer based routing switch consists of an n-input multiplexer and a tri-state buffer (see Figure 4.1(d)). The tri-state buffer determines whether the input is connected to the output or not, through a programmable SRAM cell, and the multiplexer selects which input is connected to the output [62]. The multiplexer is built based on NMOS pass transistors (see the PT-based LUT structure in Figure 3.2). Similar to the TS-buffer structure, the delay of the MUX based routing switch increases linearly with the number of connected switches in series (see Figure 4.2).

4.3. Evaluation Methodology

In FPGAs, the circuit delay consists of two parts: the delay of the logic and the delay of the routing. Usually the routing delay is the dominant one [61], because the routing inside the FPGA goes through multiple switching elements to form the desired routing. On the other hand, as discussed in Chapter 2, aging increases the switching delay of the transistors inside the circuit, and the aging effects become more critical at smaller feature sizes. Putting all together, investigating the effect of aging on the FPGA switching elements is a necessity to assure the correct functionality of FPGAs. In this section, the details of the investigation of the BTI effects on basic routing switches are described. The four different types of programmable switches (PT-keeper, TS-buffer, TG and MUX) discussed in Section 4.2 are considered. In order to do this investigation on each of the structures above, HSPICE simulations are carried out, using PTM 32 nm technology libraries [58] to obtain fresh and aged routing switch delays. In the following, the different phases of the methodology are introduced:

4.3.1. Circuit-level details and assumptions

As the device level details of the commercial FPGAs are not publicly available, some of the information is obtained from available literature and the rest, particularly regarding the relative and actual sizing of transistors, are assumed, in order to obtain logically correct results. Regarding the size of transistors, the gate lengths of all transistors of the routing switches are set to the minimum length. In addition, the width of each transistor is set to the minimum size which results in a reasonable delay in the worst case (maximum load and maximum number of routing switches connected in series). In fact, in experimental results section (Section 4.4), normalized BTI-induced $\Delta delay$ ($\Delta delay/Fresh\ delay$) is reported. That is why the particular sizing has in general a minimal effect in this analysis; therefore, no detailed optimization on sizing in terms of delay is carried out.

4.3.2. NBTI/PBTI effect analysis

As the BTI effect represents itself as a change in transistor V_{th}, it is necessary to obtain NBTI/PBTI-induced ΔV_{th} of each transistor. Similar to the method in Section 3.4, Equation (3.1) is used. The suitable coefficients are obtained from PTM reliability models [58].

The aging effect is considered for a time period of three years. To get the duty cycle of each transistor, the *SP* of the routing switch input is swept from 0.1 to 0.9, and propagated through all the transistors inside the routing switch to calculate the local *SP* of each transistor (SP_l). Afterwards, for each transistor the duty cycle is obtained such that for PMOS $Y_{PMOS} = 1 - SP_l$ and for NMOS $Y_{NMOS} = SP_l$. Then, the calculated ΔV_{th} for each transistor is adjusted in the HSPICE netlist.

4.3.3. Delay measurements

Finally, three different sets of H-SPICE simulations are carried out to obtain the delay of fresh, aged switches considering only NBTI, and aged switches considering both NBTI and PBTI, respectively. All delay measurements are taken when the signal passes through $V_{DD}/2$. Then, the maximum of rise and fall delays is considered as the delay of the routing switch. To show the effect of BTI-induced aging on timing behavior of the switch, the normalized BTI-induced delay change ($\frac{D_{aged}-D_0}{D_0}$) is reported, where D_{aged} is the delay of aged switch (NBTI or BTI) and D_0 is the delay of fresh switch.

4.3.4. Load analysis

Load information is necessary to perform correct simulations. This information consists of: 1) the wire length between each two switches, 2) the number of connected routing switches in series, and 3) the fan-out of the switch.

1. **Wire Lengths (WL)**: To model the wires in HSPICE netlist, an RC model is used. The metal resistance and capacitance values of the routing wires are proportional to its length. To obtain the wire length, the distance between two CLBs in a basic FPGA structure (if an island-based FPGA structure is considered) is defined as the minimum wire length. For long wires that cross multiple CLBs (with no routing switching in between), the wire length can be obtained by multiplication of minimum wire length times the number of CLBs it passes (segment length). For this analysis, a maximum segment length of 16 is considered. Then PTM interconnect models [58] are used to obtain the suitable parameters of copper interconnect (such as minimum wire width, space, height, thickness and inter-layer dielectric), and accordingly, the appropriate resistance and capacitance values are extracted.

2. **Number of Cascaded Switches (NCS)**: The routing between two non-cascaded CLBs usually passes through multiple routing switches. During the investigation, the number of cascaded switches is swept to show the effect of NCS on the amount of BTI-induced aging of the routing. For simplicity, the maximum NCS is set to 4 in order to show the trend.

3. **Switch fan-out**: Switch fan-out indicates when a routing switch can drive multiple other switches at the same time. For the sake of simplicity, the maximum fan-out of each switch is set to 5.

4.4. Experimental Results

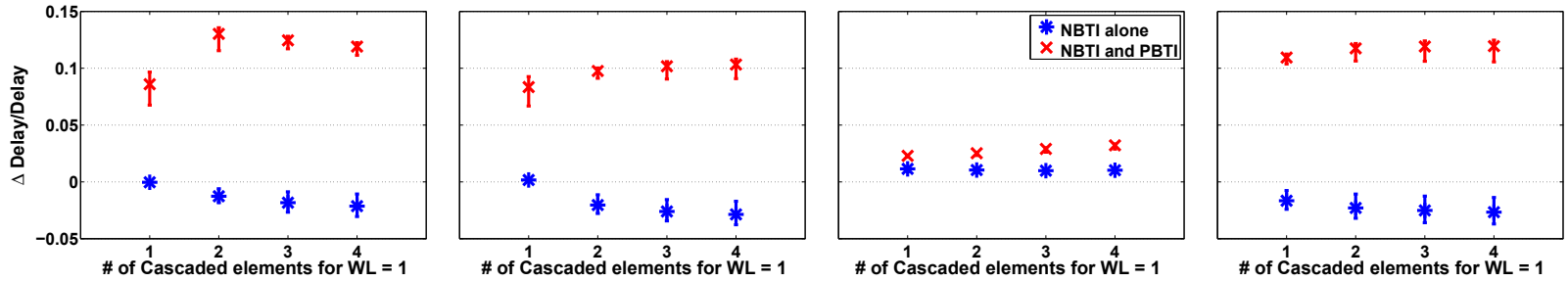
The main goal of this chapter is to investigate the aging effect on routing switches for two different cases: i) considering only NBTI effect and ii) considering BTI (both NBTI and PBTI effects). In order to have a comprehensive investigation, the four structures of routing switches discussed in Section 4.2 are considered. The parameters under investigation and the experimental setup are already discussed in Section 4.3.

4.4.1. Effect of NCS

For each cascade length, the delay from the input switch to the output of the last switch is measured. Figure 4.3 depicts the effect of NCS on both the NBTI and BTI cases for the four types of routing structures. The effect of the SP is expressed by the vertical lines, and the x and $\star$ symbols represent the average over all SPs.

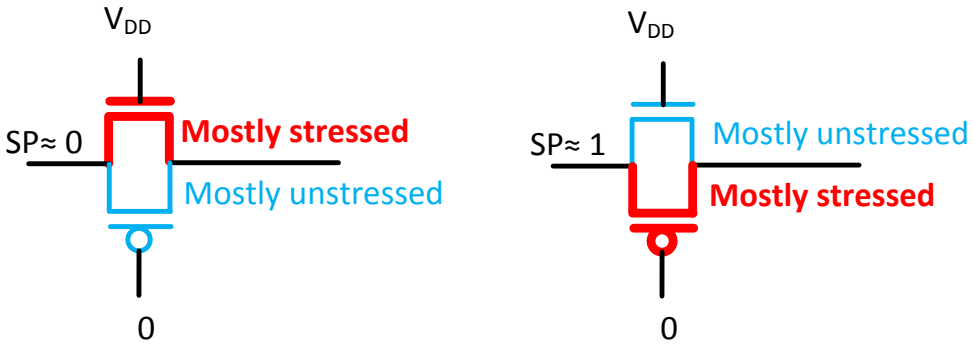
As shown in this figure, the NBTI effect is negligible for PT-keeper, TS-buffer and MUX structures. This is because the delay of the NMOS pass transistor is dominant in both structures, and as the NBTI affects only PMOS transistors, the effect of NBTI cannot be seen in these three structures. However, since the TG structure consists of parallel PMOS and NMOS transistors, as shown in Figure 4.1(c), the effect of NBTI appears in this structure. On the other side, the effect of BTI (NBTI+PBTI) on PT-keeper, TS-buffer and MUX structures is high, where the normalized $\Delta Delay$ can reach to more than 10%. Moreover, it can be seen that the normalized $\Delta Delay$ is highly dependent on the input SP. For the TG structure, the effect of BTI-induced aging is generally less than the other three structures. To better explain this, consider two different cases shown in Figure 4.4. In the first case (Figure 4.4(a)), the SP of input is near 0, in this case the NMOS transistor is under PBTI stress, while the PMOS transistor stays unstressed, which makes it able to compensate the overall aging of the TG structure. In the other case (Figure 4.4(b)), where SP is near 1, the PMOS transistor is stressed while the NMOS transistor experiences almost no stress, again resulting in a compensation for the total aging of the TG structure. This is also the reason for the very low sensitivity of BTI-induced aging to input SP in the TG structure in comparison to the other three structures.

In fact the NBTI effect makes PT-keeper, TS-buffer and MUX structures even faster, and with higher NCS, this effect gets amplified. The reason can be explained for each structure separately. For the PT-keeper structure (Figure 4.1(a)), the rise delay is the dominant delay. With NBTI effect, the inverter inside the keeper structure becomes faster at the falling edge and this causes the pull-up PMOS inside the keeper circuit to switch on faster, although its threshold voltage increases due to NBTI. For the TS-buffer structure (Figure 4.1(b)), the rise delay is also the dominant delay. With NBTI effect, the first inverter of the buffer becomes faster at falling edge, while the second one becomes slower at rising edge. Depending on the sizing of the transistors and the dominance of the delay, this effect may lead either to positive or to negative delay degradation. For the MUX structure, the same explanation for TS-buffer is applied as the tri-state buffer is a part of the MUX structure. For the TG structure, NBTI-induced aging has no sensitivity to the NCS.



(a) PT-keeper (b) TS-buffer (c) TG (d) MUX

Figure 4.3.: BTI-induced wear-out for different number of cascaded switches



(a) SP near 0 (b) SP near 1

Figure 4.4.: Effect of different SPs on the BTI-induced aging of TG switch

4.4.2. Effect of WL

To see the effect of WL on BTI-induced aging, several wire lengths, as discussed in Section 4.3, are investigated. The results show that, the BTI-induced wear-out decreases if WL increases. This is due to the fact that the WL has no effect on the BTI-induced ΔV_{th}. However, longer wires cause higher switching delays. As a result, the normalized ($\Delta Delay$) decreases when WL increases.

4.4.3. Combined influence of both WL and NCS

To connect two logic blocks inside the FPGA, there are two possibilities: the first one is to use long wires with minimum possible NCS, and the second one is to use shorter wires with more NCS. To see which one is better from the aging point of view, three different options are investigated to connect two logic blocks with segment distance of 4: i) using 1 switch with WL=4, ii) using 2 switches connected in series, each has a WL=2 and iii) using 4 cascaded switches each has a WL=1. The BTI-induced degradation of the three cases is shown in Figure 4.5. As shown in the figure, in terms of BTI-induced aging, it is better to use less number of cascaded switches with longer wires.

4.4.4. Effect of Fan-out

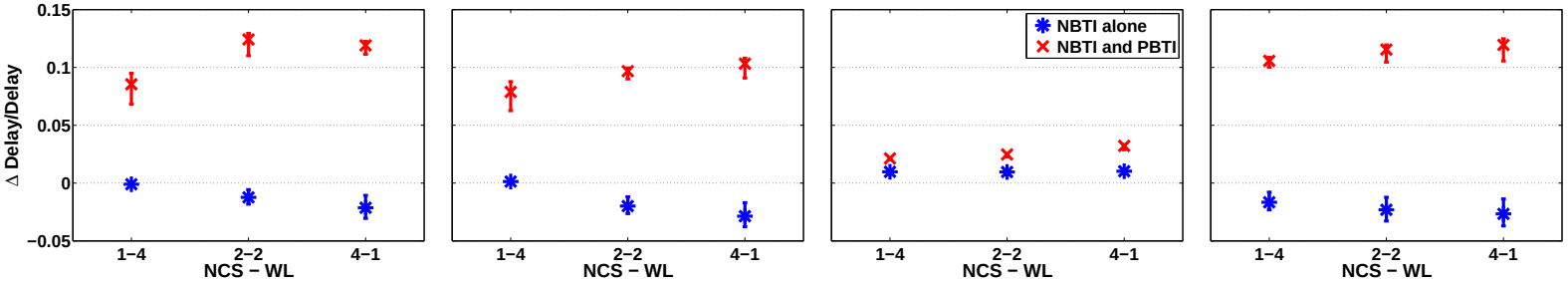
Generally, the delay of PT-keeper, TG and MUX routing switches are affected by the fan-out. However, the delay of TS-buffer without a buffer sharing (see Figure 4.6(a)) is not affected by fan-out [62]. This is due to the fact that the capacitance seen by node A is the same no matter how many of the TS-buffers are connected. Therefore, to consider the effect of fan-out on BTI-induced delay degradation of TS-buffer, the structure which is shown in Figure 4.6(b), i.e., TS-buffer with buffer sharing [62], is considered.

The effect of fan-out on BTI-induced delay degradation of routing switches is shown in Figure 4.7. The results show that the effect of fan-out is negligible for PT-keeper, TG and MUX structures, although their fresh delays are affected by fan-out. However, for TS-buffer the BTI-induced normalized $\Delta Delay$ decreases with higher fan-out.

4.4.5. Effect of Vdd

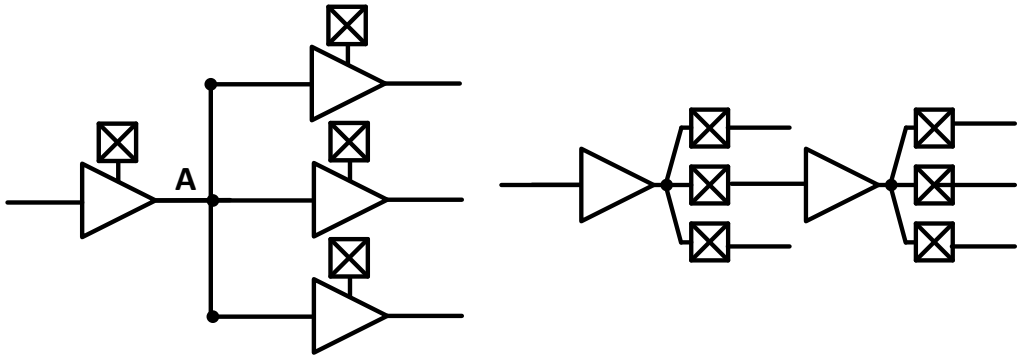
When performance is critical, higher V_{dd} is usually used at the cost of extra power consumption. On the other hand, when the power is critical, lower V_{dd} is used at the cost of performance. In this experiment, three values for V_{dd} are investigated: i) Typical V_{dd}, to represent typical circuit operation, ii) High V_{dd} (110% V_{dd}), to represent high performance case, and iii) Low V_{dd} (90% V_{dd}), to represent low power case. The effect is shown in Figure 4.8 for the four routing switches.

To explain this figure, let us consider a simple CMOS inverter. In typical CMOS gates, BTI-induced ΔV_{th} increases when supply voltage increases, although the delay of the fresh circuit decreases [25]. This leads to an increase in the normalized BTI-induced $\Delta Delay$ at higher supply voltages. However, the PT-keeper, TS-buffer and MUX structures do not follow the same trend in terms of supply voltage, and surprisingly the amount of normalized



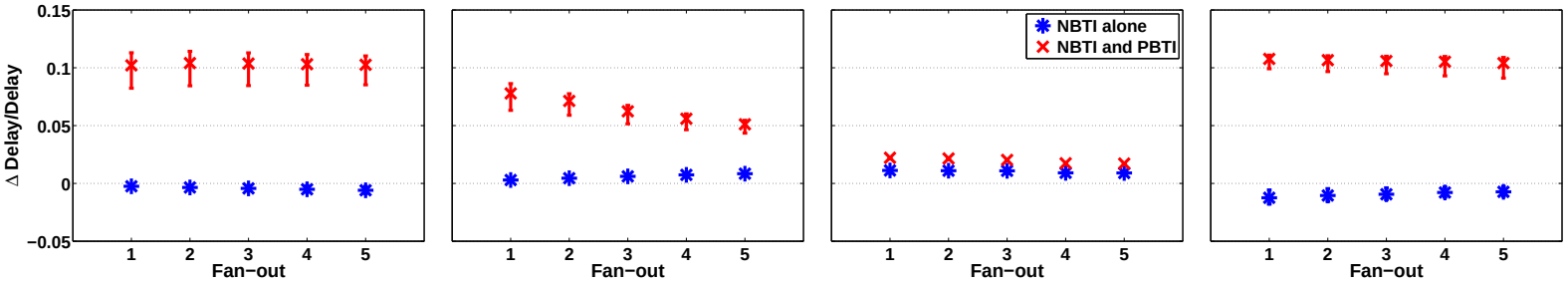
(a) PT-keeper (b) TS-buffer (c) TG (d) MUX

Figure 4.5.: The effect of wire length and number of cascaded elements together on BTI-induced wear-out



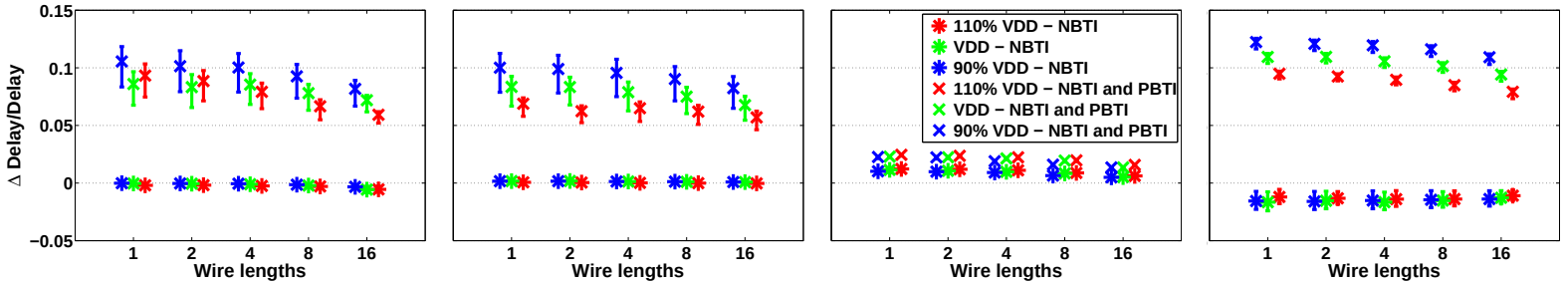
(a) Without Buffer-sharing (b) Buffer-sharing

Figure 4.6.: Two different switch type of TS-buffer



(a) PT-keeper (b) TS-buffer (c) TG (d) MUX

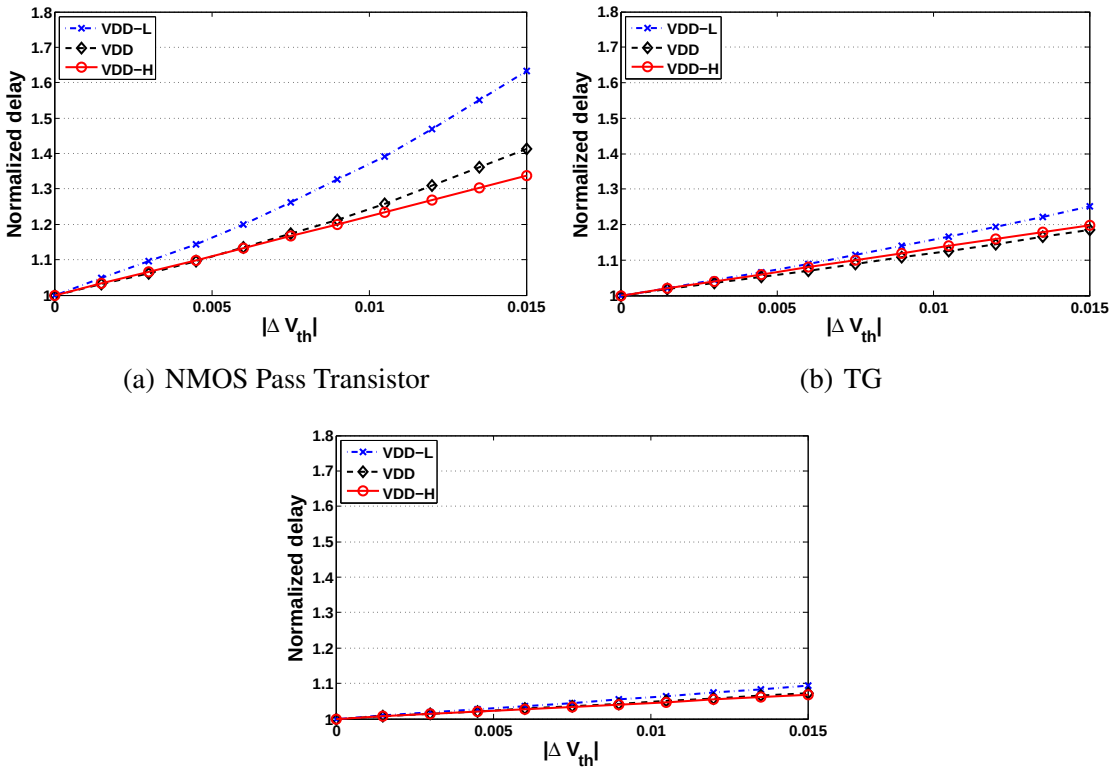
Figure 4.7.: BTI-induced wear-out for different Fan-out



(a) PT-keeper (b) TS-buffer (c) TG (d) MUX

Figure 4.8.: BTI-induced wear-out for different supply voltages

BTI-induced $\Delta Delay$ decreases for higher supply voltages in these three structures. This behavior can be explained by taking into consideration that these routing switches are not simple CMOS gates. Thus, they have different characterization and sensitivity to different parameters. Figure 4.9 shows the sensitivity of delay to ΔV_{th} for a pass transistor, a transmission gate and a simple CMOS inverter, for different supply voltages. As shown in this figure, the sensitivity of delay to ΔV_{th} is almost the same for different supply voltages in TG structure and CMOS inverter. However, the sensitivity of delay to ΔV_{th} decreases significantly when the supply voltage increases for NMOS pass transistor. Therefore, even with higher BTI-induced ΔV_{th} in higher supply voltages, the normalized BTI-induced $\Delta Delay$ decreases for the three structures that have NMOS pass transistor as a main part of the switch (i.e. PT-keeper, TS-buffer and MUX).



(a) NMOS Pass Transistor

(b) TG

(c) CMOS inverter

Figure 4.9.: Sensitivity of delay to ΔV_{th} in different circuits

However, the NBTI effect and its sensitivity to supply voltage are negligible for PT-keeper, TS-buffer and MUX structures. This is consistent with the observations of previous sections. For the TG structure, increasing the supply voltage results in a very small increase in the normalized NBTI and BTI-induced $\Delta Delay$.

4.5. Summary

In this chapter, the effect of transistor aging, due to NBTI and PBTI, on routing switches inside the FPGA has been investigated by considering different implementations through detailed SPICE simulations. The effect of different parameters has been studied on the aging of switches (i.e. wire length, number of cascaded switches, fan-out, signal probability and supply voltage). The main knobs that can be used by the FPGA manufacturers and designers to mitigate the aging, can be summarized according to the obtained results in the following:

- The NCS vs WL: less number of cascaded switches with longer wires has less aging effect than larger number of cascaded switches with shorter wires.
- Fan-out: for the TS-buffer structure, activating a higher fan-out leads to less aging.
- Supply voltage: for both PT-keeper and TS-buffer structures, using higher V_{dd} leads to less aging.
- The structures based on NMOS transistor are the best for technologies susceptible to only NBTI while the transmission gate based structure is the best when both NBTI and PBTI are important.

Chapter 5.

Experimental Analysis of Aging Effects

5.1. Introduction

The investigation of the aging effects in FPGA building blocks (LUTs and switch matrices), which has been done in the previous two chapters, is based on the results of simulating the aging of the individual transistors using the transistor-level BTI model (see Section 2.2). In this chapter, the purpose is to experimentally investigate the aging effects on commercially-available FPGA devices. For this purpose, an experimental analysis is performed to observe the impact of various parameters, such as temperature, voltage, usage (signal probabilities and switching activities) as well as process variation on performance degradation of FPGAs impacted by various aging mechanisms. The goal of this analysis is to observe the following:

- The extent of performance degradation.
- How the performance degradation correlates with intra-die process variation.
- The effect of resource utilization in FPGAs on relative aging, and
- The relative criticality of different aging mechanisms (BTI vs HCI).

The experiment[1] is done using a set of controlled ring-oscillator-based sensors with different lengths and tunable activity control, which are implemented on pre-specified locations in a Spartan-6 FPGA. These sensors are designed in such a way that the internal *Switching Activities* (SAs) and SPs of them can be varied and hence the effect of these parameters can be analyzed. Afterwards, accelerated-lifetime conditions using elevated temperatures and voltages are applied to stress the FPGA and emulate the aging process.

The performance of the sensors and hence their delay degradations are monitored throughout the experiment period (one week of stress with 80°C and V_{dd} = 150% of the nominal voltage, followed by one week of recovery). The influences of several factors such as input SPs, input SAs and *intra-die process variation* (PV) on the amount of performance degradation of the sensors are analyzed. The results provide some key insights regarding the relation between these factors and the resulted degradation. Additionally, the effect of aging on unused FPGA resources is investigated. It is shown that the unused FPGA resources age significantly as well, in some cases more than some used/active blocks.

[1] Here I acknowledge our collaborators at LIRMM, University of Montpellier 2, France. The measurement method and the experimental setup is designed, developed and performed by them [64].

The rest of the chapter is organized as follows: Section 5.2 reviews the related work. Then, Section 5.3 presents the design and the implementation of the sensors used in the analysis. Afterwards, Section 5.4 illustrates the experimental setup. Later, the experimental results and the analysis are discussed in Section 5.5. Finally, Section 5.6 concludes this chapter.

5.2. Related work

There is some prior work which uses accelerated-lifetime conditions for stressing FPGA chips and emulating the aging effects such as [52, 53] and [65]. Different modes of stress conditions are investigated in [52, 53] ranging from normal operating conditions (1.2V and 310°K) to high ones (2.2V and 420°K). Test circuits that utilize both the LUTs and the routing resources are used for measuring the performance degradation. Similar conditions are applied in [65] to analyze the effect of aging on *Physical Unclonable Functions* (PUFs). The PUFs were basically a set of ring oscillators mapped to FPGA LUTs.

The main differences between the work in this chapter and the aforementioned experiments are:

- Investigating the effect of different parameters that have direct relation with aging such as SP and SA and their combinations to identify the contribution of different degradation mechanisms from a system-level perspective.
- The uniform placement of the different sensors (i.e. the test circuits) across the FPGA chip, which capture the effect of intra-die variation and relate it to the degradation mechanisms, and also to assure a homogeneous thermal profile across the chip.
- The use of a novel non-intrusive performance monitoring method based on measuring the electromagnetic emissions of the FPGA [64], which is more accurate than the methods that have on-chip communication modules. This is because those methods may introduce biases in the measurements, which come as a result of intra-die variations, heat generation and voltage droops.

5.3. Sensors design and implementation

The main parameters that affect the aging process in FPGA, which previously discussed in Chapter 2 (see Table 2.1), must be controlled in such a way that their values can be varied in order to analyze their contribution to the total performance degradation. In this section, the details of the sensors used in the aging experiment to vary these parameters are discussed.

5.3.1. Sensors design

As discussed in Chapter 2, the BTI mechanism is distinguished by its sensitivity to signal duty cycle (Y) that can be represented by input SP as well (i.e., $Y = 1 - SP$ for PMOS transistors under NBTI and $Y = SP$ for NMOS transistors under PBTI), while the HCI

can be distinguished by the sensitivity to the amount of *AR*. It should be noted that *SA* (the input switching activity) and *AR* have the same value if the transistor gate is directly connected to the same input. Since the transistor-level details of the FPGA are proprietary, the input SAs and SPs of individual modules (e.g., LUTs and routing resources) will be used in the rest of this chapter. Although these values are different from the internal *AR* and *Y* of each transistor, they are still related with them and can give a general implication of the individual effect for each of BTI and HCI mechanisms.

Taking that into account, four controllable *Ring-Oscillator* (RO)-based sensors are used to vary the values of SPs and SAs (see Figure 5.1). The number of the stages in the RO can determine the generated frequency, and hence the amount of SA. Two of the sensors are utilized for this purpose; the first one (S1) with three inverter stages (reaching a frequency of $\approx$ 350 MHz on a Spartan-6 FPGA) and the second one (S2) with a single inverter stage (for a maximum possible frequency of $\approx$ 900 MHz on the same FPGA).

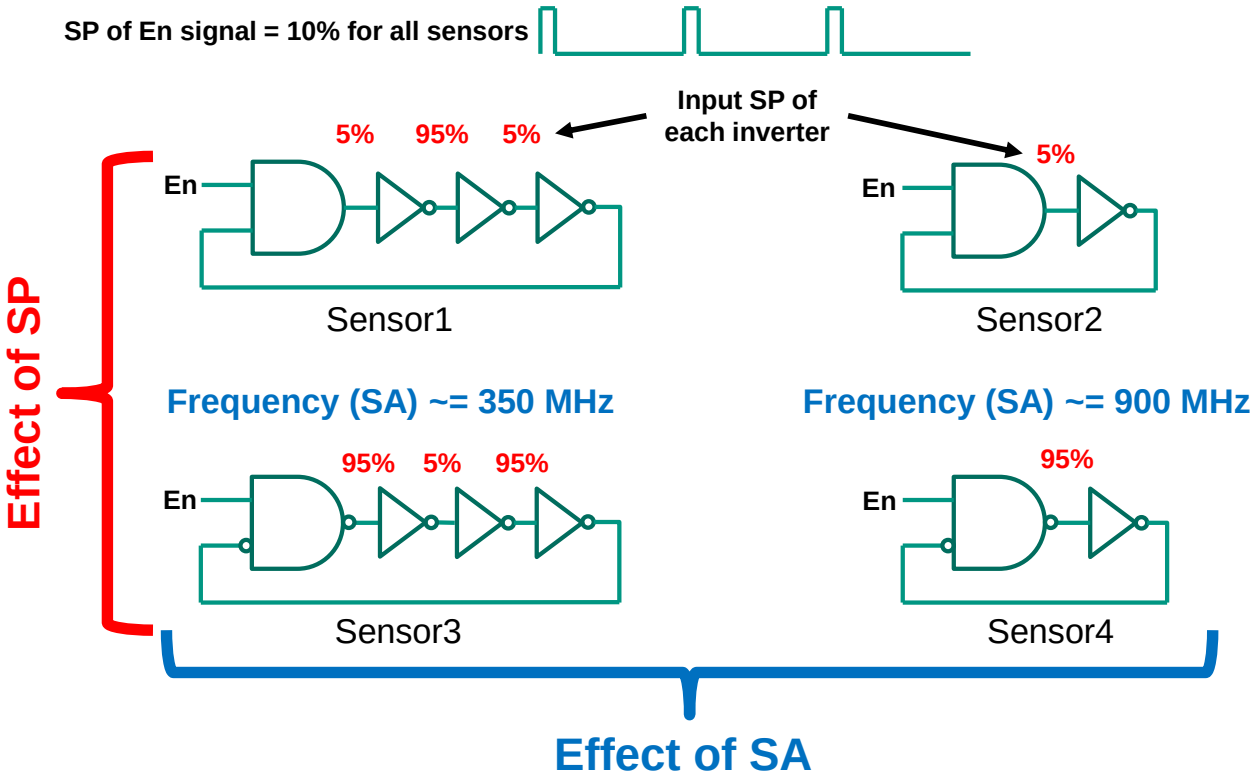


Figure 5.1.: Sensor versions to vary SPs and SAs

To vary the SP of the internal sensor stages, the switching of the RO itself can be controlled. Usually, all the internal stages of the RO have a fixed input SP of 50% because the inverters toggle continuously between logic-0 and logic-1. Adding an external enable signal (En) to enable/disable the switching of the RO can change the input SPs of all the inverters accordingly. Based on that, an enable signal is added to both S1 and S2 to specify their internal SPs. A clock signal with a duty cycle of 10% is fed to this enable signal,

which set the internal SPs of S1 to 5%, 95% and 5% respectively, and the internal SP of S2 to 5% as shown in Figure 5.2. It should be noted that this enable signal is very low frequency (i.e., 10 kHz) compared to the frequency of RO such that it does not interfere with the functionality of RO. In order to get the effect of the reverse combinations of input SPs, two other sensors are used; S3 as a counterpart to S1 with internal SPs of 95%, 5% and 95%, and S4 as a counterpart to S2 with input SP of 95% as shown in Figure 5.1.

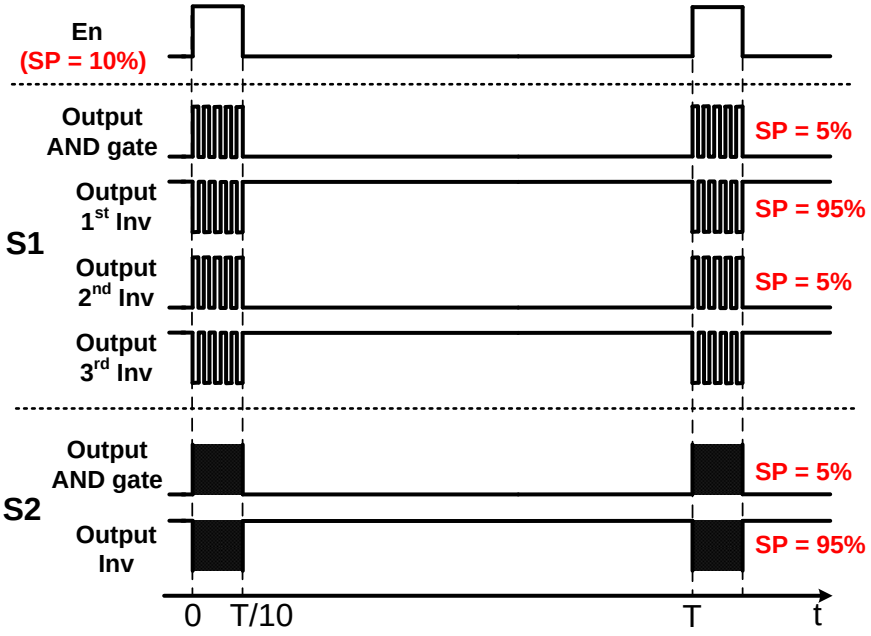


Figure 5.2.: The internal SPs for sensors S1 and S2

5.3.2. Implementation

With the aim of realizing the aging measurement[2], a Nexys 3 board that offers a xc6slx16-2csg324 Spartan 6 Xilinx FPGA is used [66]. This FPGA is manufactured with a 45-nm process technology. The slices inside the CLBs of this device can be divided into 3 different types: SliceX, SliceL and SliceM. SliceXs are the basic slices and are composed of LUTs and FFs. SliceLs include in addition an arithmetic carry structure and wide multiplexers. SliceMs, which are the most complex ones, allow using the LUTs as distributed RAM and shift registers. Since these different types do not have the same resources, one can assume that they do not present exactly the same timing performances. In order to compare the measured frequencies among different locations, the sensors have to be implemented on the same type of slices. SliceXs were chosen since they represent one half of the available FPGA slices.

[2]The implementation is done in collaboration with our collaborators at LIRMM, University of Montpellier 2, France

Similarly, the exact same configuration of LUT inputs is chosen, which means that the routing nets structure is the same for all the sensors. This is also to be sure that the same internal paths inside each LUT are used so a comparison between them is then possible. Figure 5.3 depicts the resources used in each CLB for each sensor type. The sensors are implemented using *Xilinx Design Language* (XDL) description, then converted into *Hard Macro* (HM) to be sure that the resources used in the final design are those defined in the specifications.

Figure 5.3.: Implementation of the sensors

In fact, ROs are PVT sensors (i.e. they are sensitive to Process, Voltage and Temperature variations). For this reason, 40 sensors of each type were implemented. First, the different types of sensors are placed homogeneously to guarantee a homogeneous thermal profile

across the chip. Secondly, depending on their location on the floorplan, random and systematic variations of the process may affect the actual frequencies of ROs. The different types of sensors are hence interleaved to average process variations. Figure 5.4 shows an overview of a part of the FPGA floorplan (Block RAMs, DSPs and other specific blocks are not depicted).

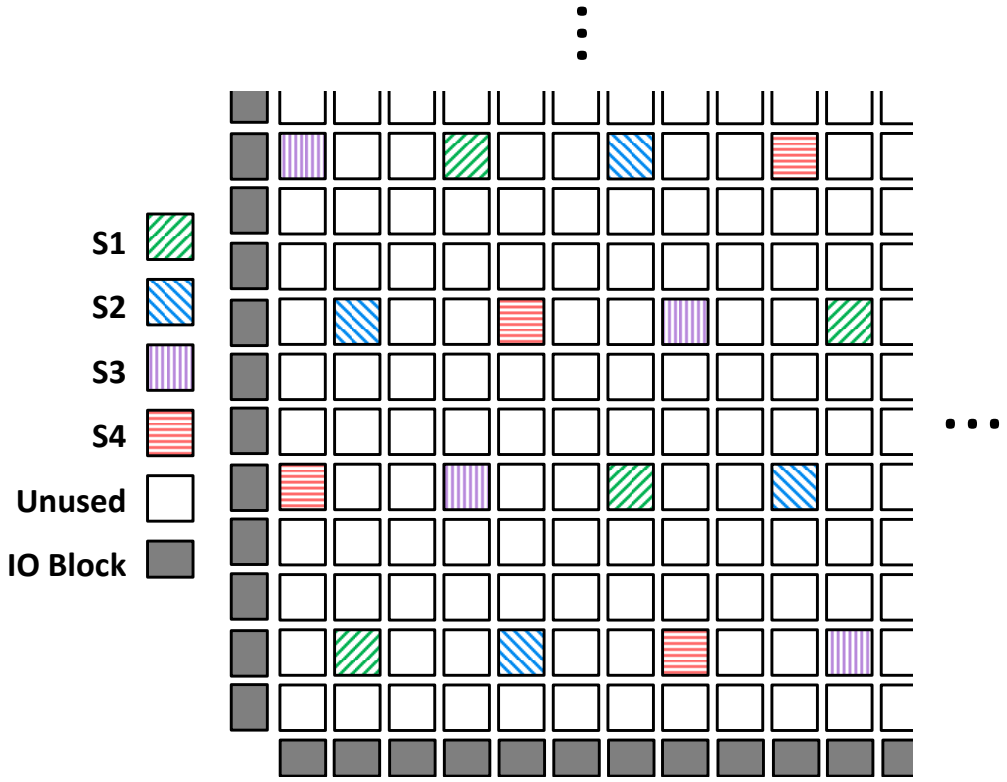


Figure 5.4.: A part of the FPGA floorplan showing the distribution of the sensors (the stress configuration)

The implemented design also allows enabling and disabling the sensor to reach the specified SA and SP values (see Section 5.3.1). The enable signal has a frequency of 10 kHz and a duty cycle of 10%. This signal is generated using a 100 MHz oscillator clock available on Nexys 3 board. Its frequency is decreased to 10 MHz using a *Digital Clock Manager* (DCM) and then a counter generates the final frequency and duty cycle of the enable signal.

5.4. Experimental setup and schedule

In this section, the experimental setup, depicted in Figure 5.5, is described for both applying the accelerated lifetime conditions and monitoring the performance of the sensors before and after the stress. This is followed by an illustration for the schedule in which the experiment is applied.

5.4.1. Setup

In this setup[3], to eliminate the on-board sources of variation, in addition to applying the accelerated lifetime conditions, the Nexys board was modified to have a direct access to the FPGA core voltage. The FPGA board is then supplied using an external high-precision dynamic voltage controller and the temperature is regulated using a dynamically controllable thermal chamber (See Figure 5.5). For monitoring the performance of the sensors before and after the stress, an *Electro-Magnetic* (EM) method [64] is used to guarantee that only variations due to aging are captured. This is unlike related approaches that have on-chip communication modules, which are susceptible to intra-die variations that can influence the measurements. The monitoring is done by configuring the FPGA each time with a single sensor at a certain location, and then capturing its frequency using EM analysis. A near-field EM probe is placed over the package to capture the EM emanations. Afterwards, an FFT for the digitalized EM signal is performed to obtain the frequency of the associated sensor. This process is repeated for each possible location of each sensor type on the FPGA. In this way, a frequency cartography for the FPGA is built. As estimated in [67], the measurement error is lower than 100 kHz with this equipment. The same error estimation procedure was followed again here, and similar results have been acquired as well.

5.4.2. Schedule

The experimental schedule is as follows (it should be noted that the whole experiment is automated using a controlling PC (see Figure 5.5)):

1. At the beginning (at Day 0), before stressing the FPGA circuit, a fresh characterization for the whole FPGA is performed under nominal operational conditions (i.e., with 1.2V and 25°C). This is done by placing S1, S2, S3 and S4 successively (only one sensor from each type at a time) at their pre-defined locations (40 positions for each sensor type) for capturing their fresh oscillating frequency. A whole cartography of the FPGA is also performed as a reference with S0 (the same RO used in [64]).

2. Afterwards, the accelerated lifetime conditions are applied by exposing the FPGA to an elevated temperature and core voltage. The core was supplied by a 1.8V voltage (50% above its nominal value of 1.2 V) using the aforementioned external power supply, and the FPGA was heated to 80°C using the thermal chamber, while the stress configuration (Figure 5.4) was in operation (please note that S0 is not loaded during the stress). These conditions are applied for 7 days continuously without any interrupt. This is to avoid any possible intermediate recovery that may happen to the sensors.

3. At Day 7, directly after the stress, the circuit is set back under the nominal conditions, and a full characterization is performed, exactly as in step 1, for all the sensors (S0 - S4). The FPGA is then powered-off for the rest of the day.

[3]The measurement method and the experimental setup is designed, developed and performed by our collaborators at LIRMM, University of Montpellier 2, France [64].

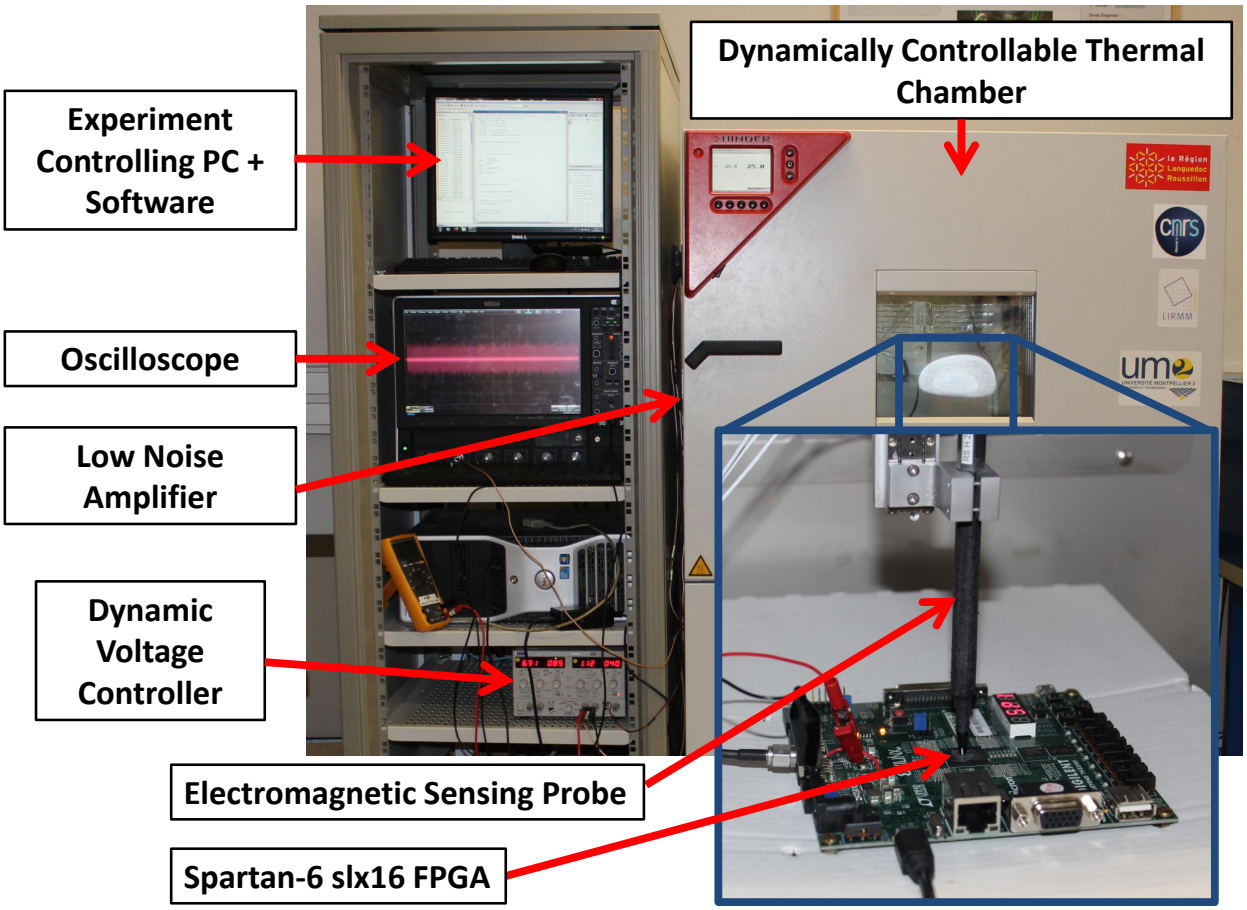


Figure 5.5.: Experimental setup [64]

4. Step 3 is then repeated on a daily basis until Day 14.

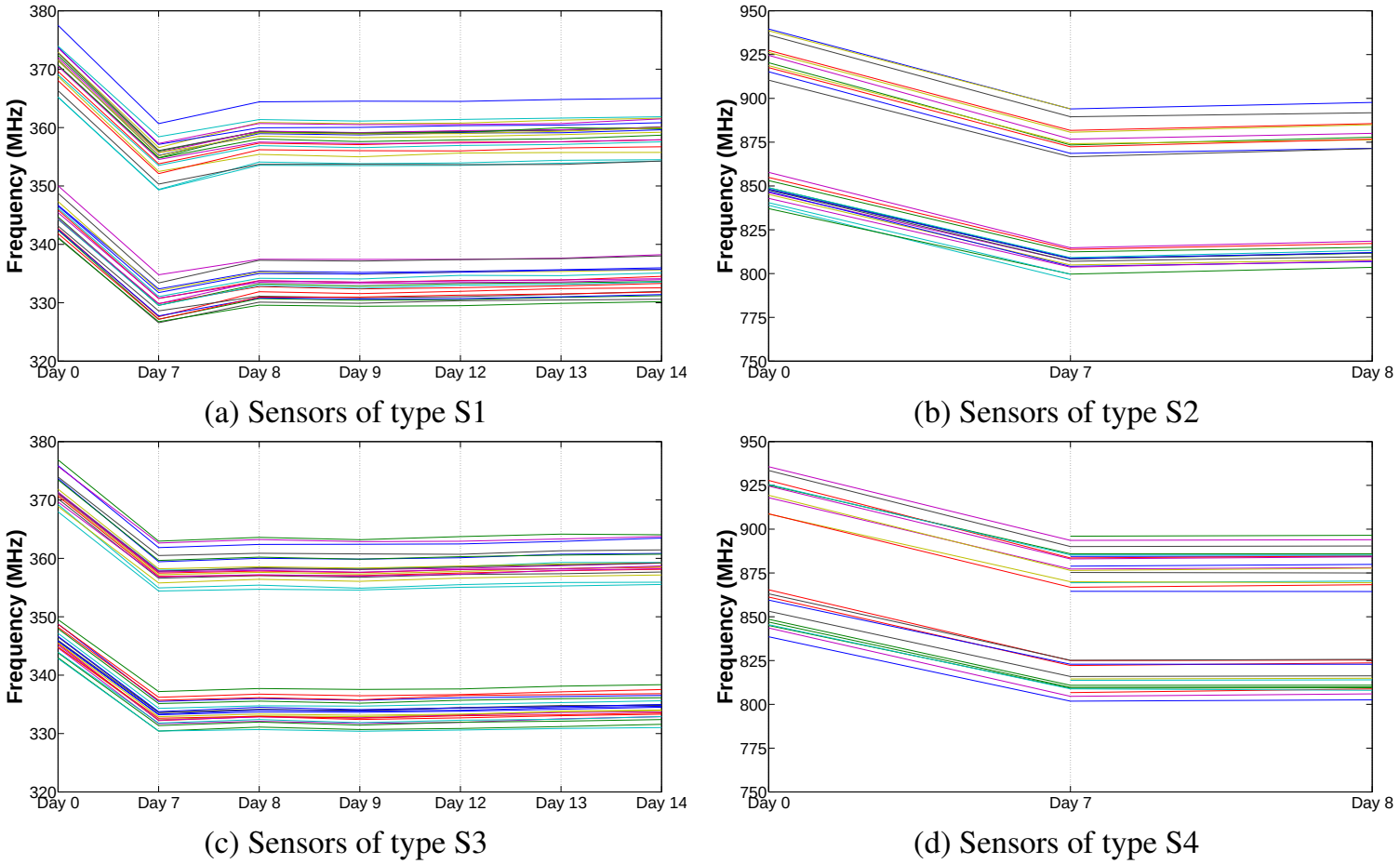
5.5. Experimental results and analysis

As discussed in Section 5.4, the FPGA is characterized before the stress (at Day 0) and after the stress (at Day 7), then on a daily basis during the recovery phase (at Day 8, Day 9, Day 12, Day 13 and Day 14). The main results and observations are discussed in the following section followed by a thorough analysis in Section 5.5.2.

5.5.1. Results

The frequency changes of each type of sensors (40 sensors from each type) during the experiment period are depicted in Figure 5.6. Unfortunately, for the sensors of type S4, the measurement data got corrupted after Day 8. Actually it is because post-processing is more complex with the equipment with such frequencies (which means it requires much time and visual analysis). However, the trend was the same after 2 and 3 days on a subset of points. Therefore, for both S2 and S4 sensors the results are shown only till Day 8 to allow the comparison between these two types.

As can be observed in Figure 5.6, there are two distinguished groups of frequencies appearing for each sensor type. These are the result of mapping the sensors to different



(a) Sensors of type S1

(b) Sensors of type S2

(c) Sensors of type S3

(d) Sensors of type S4

Figure 5.6.: The frequency change of each sensor (40 sensors from each type, each line color represents a single sensor)

Table 5.1.: Mean frequency and max variation

	Sensors of type S1		Sensors of type S3		Sensors of type S2		Sensors of type S4	
	SliceM CLB	**SliceL CLB**	**SliceM CLB**	**SliceL CLB**	**SliceM CLB**	**SliceL CLB**	**SliceM CLB**	**SliceL CLB**
Day 0								
F_{mean} **(MHz)**	344.8	371.1	345.9	371.8	846.2	921.0	849.9	922.8
Var_{max} **(MHz)**	9.0 (2.6%)	12.3 (3.3%)	6.7 (1.9%)	8.9 (2.4%)	23.3 (2.8%)	40.2 (4.4%)	29.9 (3.5%)	27.0 (2.9%)
Day 7 (directly after the stress)								
F_{mean} **(MHz)**	330.0	355.0	333.2	358.4	807.2	881.7	813.1	880.3
Var_{max} **(MHz)**	8.2 (2.5%)	11.4 (3.2%)	6.8 (2.0%)	8.5 (2.4%)	18.2 (2.3%)	33.7 (3.8%)	23.3 (2.9%)	31.5 (3.6%)
Day 8 (after 1 day of recovery)								
F_{mean} **(MHz)**	333.3	358.0	333.7	358.8	810.6	881.1	813.5	880.5
Var_{max} **(MHz)**	7.9 (2.4%)	10.9 (3.0%)	7.0 (2.1%)	8.9 (2.5%)	20.8 (2.6%)	26.4 (3.0%)	23.1 (2.9%)	32.1 (3.7%)
Day 9 (after 2 days of recovery)								
F_{mean} **(MHz)**	333.0	358.3	333.4	358.6	810.5	883.2	NA	NA
Var_{max} **(MHz)**	8.0 (2.4%)	11.0 (3.1%)	7.2 (2.2%)	8.6 (2.4%)	18.6 (2.3%)	30.5 (3.5%)	NA	NA

CLBs. Actually, as discussed in Section 5.3.2, each CLB in Spartan-6 FPGA contains two type of slices: either SliceX with SliceL or SliceX with SliceM. Although only SliceXs are chosen to map the sensors, the results show that the sensors mapped to the CLBs that contain SliceM beside SliceX are slower than those mapped to the CLBs that contain SliceL beside SliceX by about 7 to 9%. This is in line with the previous results of [64]. The other observation is that there is a clear performance variation between the sensors of each type. Table 5.1 separates the sensors mapped to the CLBs with SliceL from those with SliceM and shows the mean frequency (F_{mean}) and the maximum variation (Var_{max}) for each type.

Although the previous observations are interesting, the main observation however in both Figure 5.6 and Table 5.1 is the relatively large performance degradation after the stress for all the sensors. Figure 5.7 shows this degradation as a normalized ΔDelay for the sensors of each type. This degradation reaches to 5.17% for some sensors of type S2. Table 5.2 records the minimum, mean and maximum values for each type during the first 3 days after the stress.

In addition to the previous results, it is mentioned in Section 5.4 that a characterization is made for the unused FPGA resources using a fifth type of sensors (S0) before and after the stress. It should be noted again here that unlike S1-S4, Sensors of type S0 are not mapped during the stress phase and the targeted resources were completely unused (i.e., no user configurations are mapped to them). Similarly, the result of this characterization is shown in Figure 5.8 both as a frequency change (Figure 5.8(a)) and as a normalized ΔDelay change (Figure 5.8(b)). Also, Table 5.3 shows the minimum, mean and maximum values of this degradation.

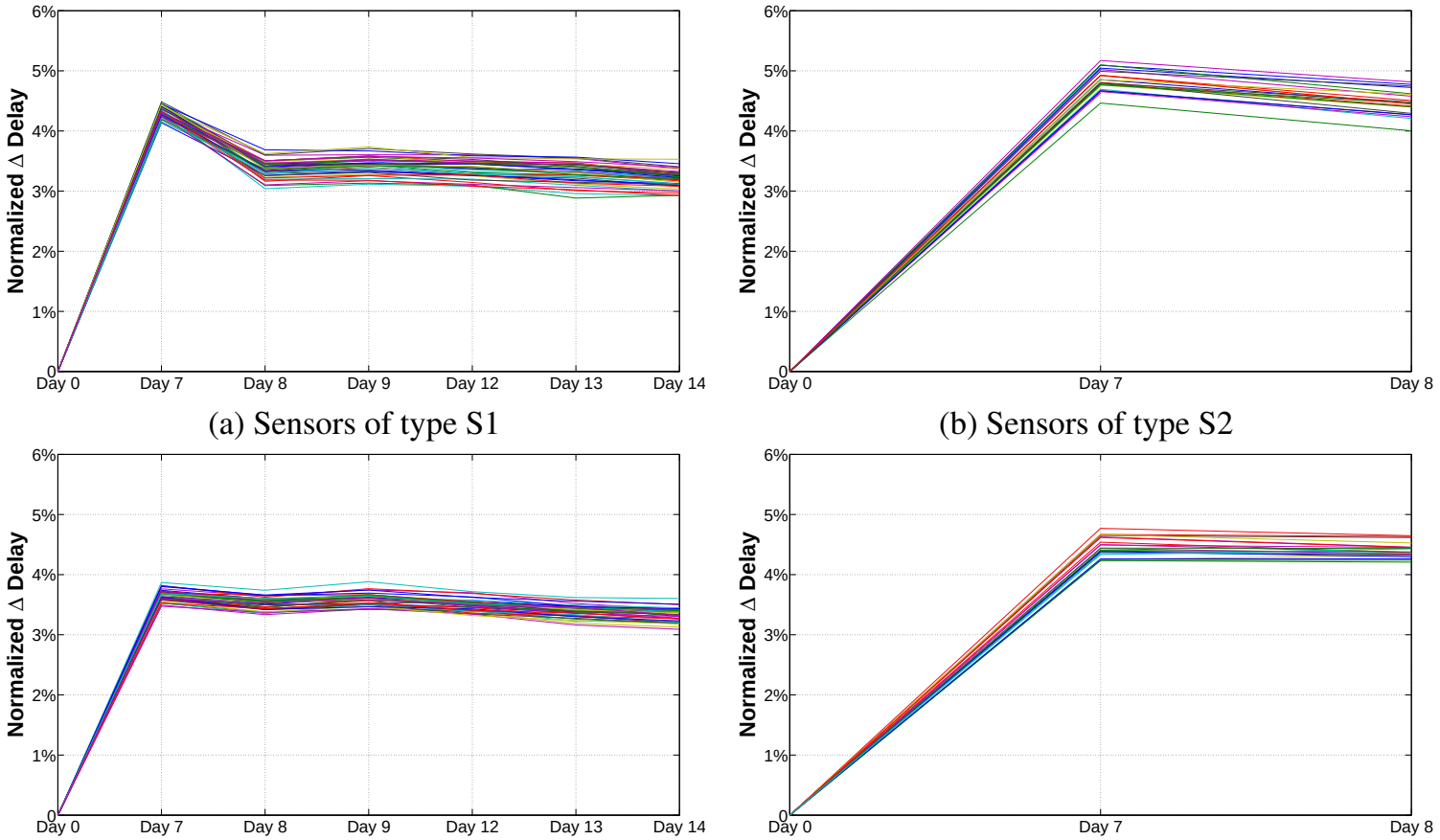
5.5.2. Analysis

The results of the aging experiment, which are given in the previous section, show an aging extent of up to 5.17% after just one week of continuous stress. The effects of different parameters on this extent are analyzed in the following:

5.5.2.1. Effect of Input Signal Probability (SP)

As mentioned in Section 5.3.1, in terms of SPs, S1 is the counterpart of S3 and S2 is the counterpart of S4. If we take the measurements of Day 7 (directly after the stress), we will find that S1 has higher aging than S3 by about 18% on average, also S2 has higher aging than S4 by about 10% on average. The measurements of the next day (Day 8), show that some recovery happens. This recovery was higher in S1 and S2 than in S3 and S4. The results of the recovery make the aging of S1 and S3 comparable. The same also can be observed for S2 and S4. This trend continues until the end of the experiment at Day 14.

Based on the fact that the BTI mechanism is sensitive to SP changes and it is the only mechanism that has a recovery effect (see Chapter 2), these results show that the input SPs play a role in aging. However, more experiments are needed to verify this and to determine what is better in terms of aging, low input SPs or higher ones.



(a) Sensors of type S1

(b) Sensors of type S2

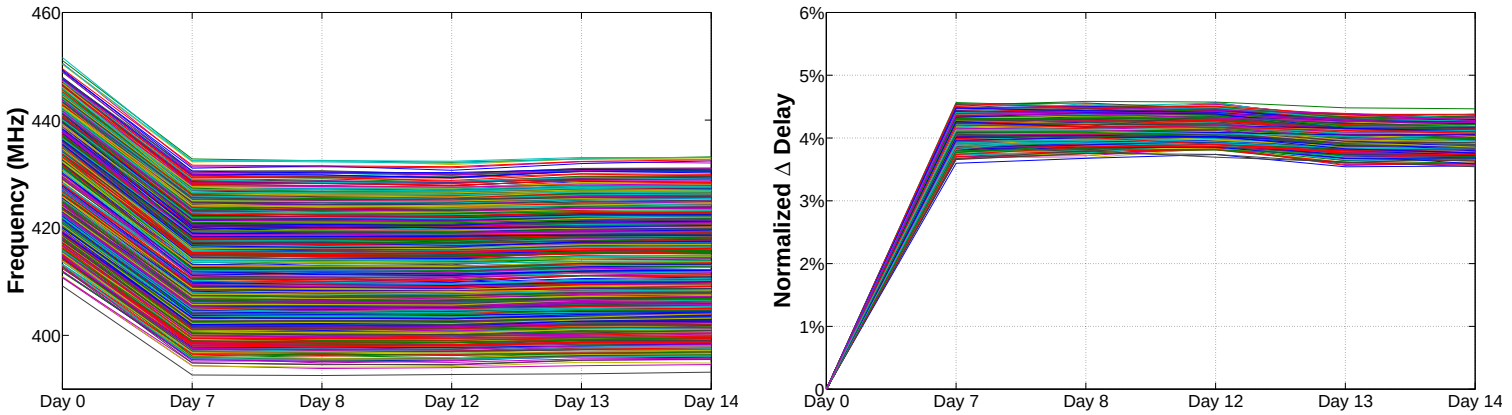
(c) Sensors of type S3

(d) Sensors of type S4

Figure 5.7.: Normalized ΔDelay change of each sensor (40 sensors from each type, each line color represents a single sensor)

Table 5.2.: Normalized ΔDelay increase after the stress (aging rates)

Aging	**Sensors of type S1**			**Sensors of type S3**			**Sensors of type S2**			**Sensors of type S4**		
	Min	**Mean**	**Max**	**Min**	**Mean**	**Max**	**Min**	**Mean**	**Max**	**Min**	**Mean**	**Max**
At Day 7	4.14%	4.33%	4.48%	3.48%	3.66%	3.87%	4.46%	4.86%	5.17%	4.23%	4.43%	4.77%
At Day 8	3.04%	3.37%	3.69%	3.33%	3.53%	3.74%	4.00%	4.47%	4.81%	4.21%	4.39%	4.65%
At Day 9	3.12%	3.43%	3.73%	3.42%	3.60%	3.88%	4.14%	4.51%	4.79%	NA	NA	NA



(a) Frequency change

(b) Normalized ΔDelay change

Figure 5.8.: Unused CLBs (each line color represents a single sensor)

Table 5.3.: Normalized ΔDelay increase after the stress (aging rates) for unused CLBs

Aging	**Min**	**Mean**	**Max**
At Day 7	3.60%	4.10%	4.57%
At Day 8	3.68%	4.13%	4.58%
At Day 12	3.69%	4.16%	4.57%
At Day 13	3.54%	4.01%	4.48%
At Day 14	3.54%	3.98%	4.47%

5.5.2.2. Effect of Switching Activity (SA)

In terms of SA, S1 is the counterpart of S2 and S3 is the counterpart of S4 (see Section 5.3.1). Both S2 and S4 are about 250% faster (i.e. have higher SAs) than their counterparts. However, the measurements at Day 7 (directly after the stress) show that S2 has only about 12% aging on average more than S1, and S4 has about 21% aging on average more than S3. After the recovery at Day 8, this difference becomes about 33% on average S2 more than S1 and about 24% on average S4 more than S3. This trend continues till the end of the experiment at Day 14.

These results show that the frequency change has a limited effect on aging for this FPGA technology. Another support for this conclusion is the results of aging for both CLBs that contain SliceL or SliceM. The sensors mapped to the CLBs that contain SliceL are faster than those mapped to the CLBs that contain SliceM by about 7 - 9% (see Figure 5.6 and Table 5.1). However, there is no noticeable difference between the aging of both (see Figure 5.7 and Table 5.2). Again, this is due to the limited effect of frequency change on aging for this technology. Since only HCI, and not BTI, is affected by frequency changes, it can be concluded that the HCI effect in this technology node is less than the BTI effect.

5.5.2.3. Aging of Unused Resources

As mentioned before, the unused LUTs were completely unmapped during the stress. However, characterizing them using S0 before and after the stress shows that they also age (see Figure 5.8 and Table 5.3). The aging of these unused resources is even more than some of the used resources (it is more than the aging of S1 and S3 sensors). The explanation of this behavior is the fact that there is no power gating in Spartan-6 FPGA, which means that even if the resources are unused, they are powered-on all the time. A default configuration of all zeros or all ones is normally loaded to the unused resources [68] to isolate them from the mapped circuit. This puts the unused resources under a static type of BTI stress, and this is the reason of their degradation.

These results are in line with the simulation results of Chapter 3. As a future work, investigating the effect of applying different configurations for the unused LUTs on the aging rate is planned.

5.5.2.4. Recovery Results

Among all sensors, only S1 and S2 sensors show clear recovery. A possible reason could be that there was not enough recovery time during the stress (the first 7 days) for these two types of sensors, while the other sensors had such time to recover. In fact, since all sensors are ROs, and the transistors under stress were switching, there was an AC type of stress, not a DC. This means that some sort of BTI relaxation was happening during the 7 days of stress for some of the sensors. This may explain why the aging of S1 and S3 (S2 and S4) becomes comparable after Day 8. However, still more experiment is needed to verify this.

5.5.2.5. Effect of Process Variation (PV)

Aging influence on PV Although the performance of the sensors degraded after the stress, the maximum variation (Var_{max}) did slightly change during the entire experiment period for all the sensors (see Figure 5.6 and Table 5.1). This applies for the unused resources as well (see Figure 5.8). In other words, the aging affects the mean value (μ) of the frequency for each sensor type, but has almost no influence on the standard deviation (σ) of each type. Figure 5.9 shows the probability density function for the sensors of type S1 before and after the stress.

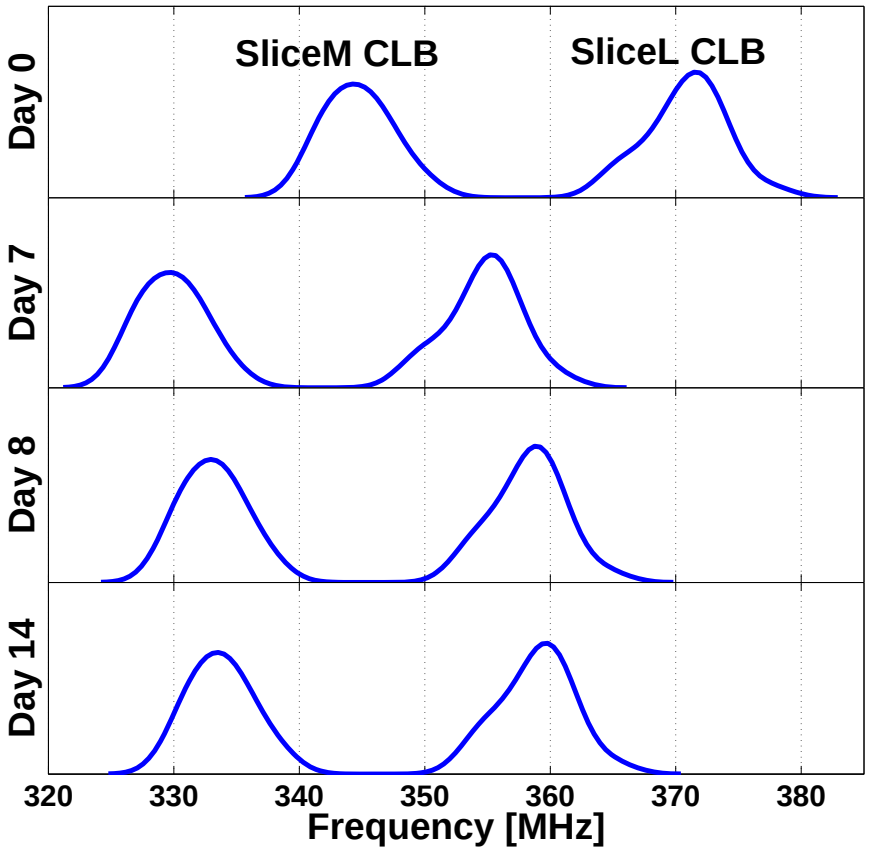


Figure 5.9.: The probability density function for the sensors of type S1 before and after the stress. The aging affects the mean value (μ) of the frequency, but has almost no influence on the standard deviation (σ)

PV influence on aging It was also desired to see how the PV affects both the initial delay and the aging of the sensors. Figure 5.10 shows a comparison for all the sensors of type S1 for their initial frequency at Day 0 and their aging directly after the stress (at Day 7) and at the end of the experiment after a week of recovery (at Day 14). The figure shows clearly that there is no correlation between how the PV affects the initial delay and how it

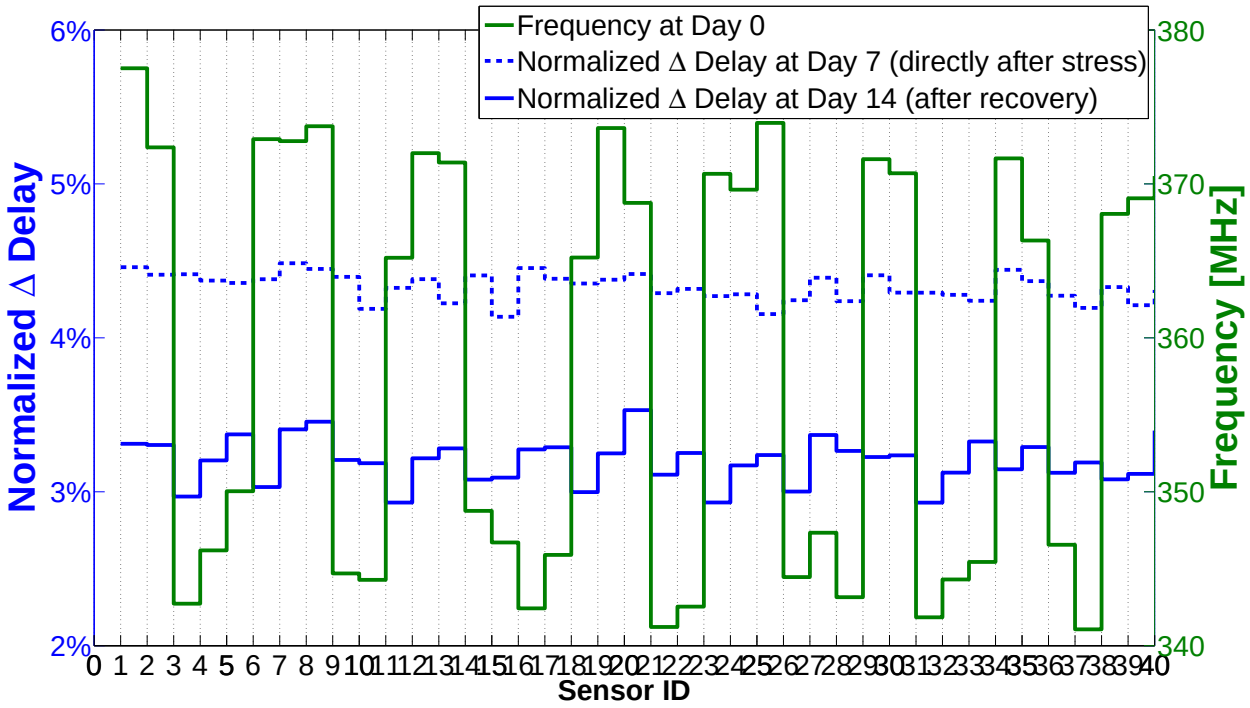


Figure 5.10.: A comparison between the effect of PV on the initial delay of the sensors of type S1 and their aging directly after the stress and after the recovery (40 sensors of type S1)

affects the aging and recovery of different sensors. The same results apply for all sensor types (S1 - S4).

5.6. Summary

In this chapter, an analysis has been presented for some of the main parameters influencing the performance degradation resulted from transistor aging in FPGAs. The analysis is based on the result of stressing a Spartan-6 FPGA, where a set of controlled ring-oscillator-based sensors with different lengths and tunable activity control is implemented. Furthermore, a novel monitoring method based on measuring the electromagnetic emissions of the FPGA is used to accurately monitor the performance of the sensors before and after the stress. The results showed a degradation of up to 5.17% in the performance of the sensors after one week of stress. The following conclusions are also observed:

- Input SPs play a role in degradation.
- The input frequency (SAs) plays also a role, but the impact of operational frequency on the aging was less compared to SP. This suggests that BTI aging is the dominant factor in this technology node compared to HCI.
- There is no correlation between how the PV affects the initial delay and how it affects the aged one.

- The unused FPGA resources age significantly as well, in some cases more than some used/active blocks.

More experiments are planned in the future for further analysis of possible aging mitigation strategies based on the observations of this chapter.

Chapter 6.

Thermal-Profile Estimation of FPGA-Mapped Designs

6.1. Introduction

As discussed in Chapter 2, the temperature has an exponential relation with both BTI and HCI effects. Therefore, to correctly estimate the aging effect, an accurate thermal estimation for the FPGA is required. In this chapter, a method for accurate thermal-profile estimation for the FPGA-mapped design is presented. This thermal estimation is a main part of the high-level aging estimation, which will be discussed later in Chapter 7.

The modern FPGAs are very susceptible against a wide range of reliability problems induced by elevated temperatures and unbalanced thermal distribution across the FPGA die [69]. Besides, the problem of power in SRAM-based FPGAs, particularly leakage power, is even worse than in ASICs, because most of the transistors are powered on all the time even in the unused blocks of the FPGA, and hence the leakage can reach more than 50% of the total power consumption [70]. These reasons made predicting the thermal characteristics of modern FPGA platforms *at design time* something inevitable. This helps to ensure the lifetime of chips and to avoid the unforeseen failures produced by elevated temperatures during system operation. Furthermore, knowing the thermal characteristics, at design time, helps the designers to avoid possible hot-spots in an early phase of the design process by applying alternative routing and placement options, reducing the operational clock frequency or any other possible technique. This eliminates the overhead of the in-field thermal behavior investigation of FPGA chips, and thus reduces the overall design time and cost. Moreover, this helps to check whether the design meets its thermal constraints during the iterative design process (i.e. while optimizing for other design constraints).

Traditionally, studying the thermal characteristics at design time can be done by generating the power trace for different blocks[1] in the chip and then using a thermal simulator (e.g., HotSpot [71, 72], ISAC [73], etc.) to estimate the generated temperatures. These simulators require accurate information about the power distribution (both leakage and dynamic) across the chip to build the corresponding thermal profile. The key challenge in FPGA is that, unlike the dynamic power, which is provided in detail through the vendor's power tool (such as the XPower tool from Xilinx and the PowerPlay tool from Altera), the leakage power for the whole FPGA chip is reported as only one value without any details about its distribution across the FPGA die. The lack of knowledge of how exactly the leakage power is distributed across the FPGA chip leads to highly inaccurate power traces for

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

different FPGA blocks and consequently unfaithful thermal estimation. This is shown later in this chapter.

In this chapter, the challenge of unknown leakage distribution in FPGA chips is addressed. Also, a method is presented, based on temperature-leakage loop estimation model, to accurately distribute the reported leakage power across the FPGA chip for more accurate thermal profile estimation. Based on this method, a generic flow is presented for steady-state thermal profile estimation in FPGA at design time using the information provided from the vendor's tools. This flow can be used with any SRAM-based FPGA. To precisely calibrate the proposed method and its model, an infrared-based thermal camera is employed[2], which measures the emissions from the backside of Xilinx Virtex-5 chip. The Camera is used also to validate the proposed approach against various benchmark circuits. The obtained results show that the proposed flow for steady-state thermal profile estimation comes with an average absolute error of 1°C and 3°C in terms of the thermal variation and peak temperature, respectively.

The rest of this chapter is organized as follows: Section 6.2 describes the related work. Afterward, Section 6.3 highlights the relation between temperature and leakage power which motivates the work in this chapter. Afterward, Section 6.4 presents the proposed approach for leakage power distribution and adaptation. Section 6.5 then explains the proposed flow to estimate the thermal behavior of the FPGA. Section 6.6 illustrates the utilized experimental platform that employs an infrared camera to accurately calibrate and validate the estimated thermal characteristics. Later, the evaluation and the comparison results are given in Section 6.7. Finally, Section 6.8 concludes this chapter.

6.2. Related work

Thermal simulators for estimating the thermal profiles of the FPGA has been proposed in [71] and [74]. In [71], the authors measure the power consumption of the FPGA before and after loading their design to determine its power consumption. Although this approach can measure the power consumption accurately, it does that for the whole design or the loaded module, which means that the generation of a fine-grained power trace is not possible and hence, the resulted thermal profile will not have enough resolution to determine the hot-spots within the loaded design. In [74], the power trace is generated considering the worst-case power consumption of the FPGA components. This may not provide accurate thermal results. Furthermore, no details are given on how the leakage power, which is reported from the FPGA's power tool as a single value, is distributed across the FPGA chip. In summary, although the leakage power can be estimated for the whole FPGA accurately, no enough details are given in literature or the FPGA toolset about its distribution across the FPGA chip for the thermal simulation.

[2]I acknowledge here Prof. Jörg Henkel and his group from the chair of embedded systems at Karlsruhe Institute of Technology who have done the thermal camera setup

6.3. Motivation

In order to motivate the proposed thermal profile estimation flow, the relationship between temperature and leakage power is first highlighted through theoretical models and FPGA experiments. Afterward, the leakage power issue in FPGAs is explained.

6.3.1. Leakage-Temperature Relation

Leakage power and temperature have a closed-loop relation. This relation is modeled in literature in two different equations. The first one (Equation 6.1) describes the relation between leakage power and temperature of *Metal-Oxide-Semiconductor Field-Effect Transistor* (MOSFET) transistors [75].

$$P_{leak} = P_0 \times e^{-k/T} \tag{6.1}$$

where P_{leak} is leakage power, T is temperature, P_0 and k are process dependent constants. The second model (Equation 6.2) describes the relation between the junction temperature of the package and the total power consumption of it (leakage + dynamic) [75].

$$T = T_a + \theta \times (P_{leak} + P_d) \tag{6.2}$$

where T_a is the ambient temperature, θ is the thermal resistance of the package, P_{leak} and P_d are the leakage and dynamic power consumption of the chip, respectively. It is clear from the two equations that rising the temperature will increase the leakage, and in turn will rise the temperature again and so on until the heat generated from the package stabilizes with the heat dissipated from it.

In order to better understand this closed-loop relation between the temperature and the leakage power in FPGAs, several experiments using different FPGA platforms are carried out. In one of the experiments, a Spartan6-based board (Digilent Atlys) with built-in power monitoring capabilities for each power supply voltage on the board is employed. As a test-bench circuit, a 128-bit pipelined AES-encoder circuit at 300 MHz is used. Additionally, to override the contribution of the dynamic power on the total power, constant signal activities are assured for the internal logic of the testbench circuit. In that way, any change to the total power value that may happen would be only due to the leakage contribution.

The board is switched-on, and left until stable power values are reached. Then, the testbench design is loaded to the FPGA. After few minutes, a fan is used to circulate air at the heatsink of the FPGA, in order to cool the FPGA down. The fan is left on for about one minute and then switched off for the rest of the experiment. The power behavior of FPGA's internal logic supply voltage (V_{int}) during this experiment is depicted in Figure 6.1.

As it can be seen in this figure, the power value has a sudden increase directly after the testbench is loaded to the FPGA. Then, a logarithmic increase to the power value happens after the design is loaded to the FPGA, which can be explained as follow: as the power value increased due to the loading of the testbench on the FPGA, the junction temperature of the package increased as well according to Equation 6.2. This increase in temperature caused an increase to the leakage power of the internal transistors of the FPGA according to

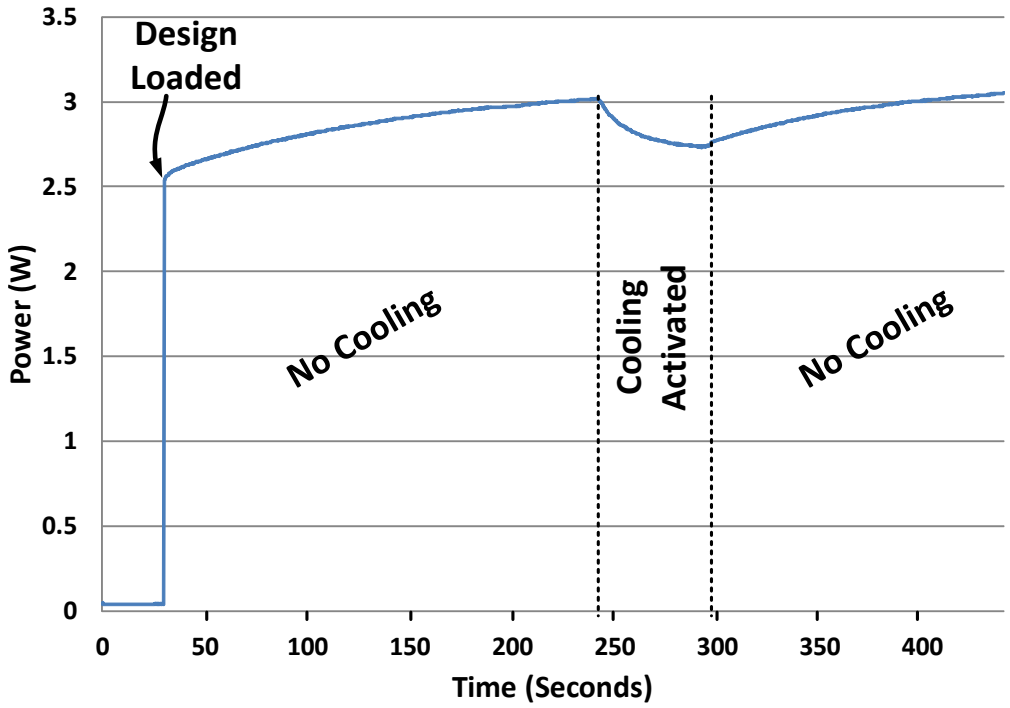


Figure 6.1.: Temperature - Power relation through the time for V_{int} power source in a SPARTAN-6 FPGA

Equation 6.1, which in turn caused an increase to temperature and so on this loop continued until the heat dissipation of the heatsink stabilized with the heat generation of the FPGA package. The support of this explanation is the second phase of the experiment, where the fan is activated so the temperature decreased and caused the leakage to decrease as well. The third phase supports the same trend as well, where switching off the fan increases the temperature and the leakage again. This experiment clearly shows the relation between temperature and leakage power.

6.3.2. FPGA Leakage Power

In SRAM based FPGAs, most of the internal transistors are powered on all the time during operation. This causes a large amount of leakage power consumption even if the design loaded onto the FPGA is very small, because in addition to the used FPGA blocks, the unused blocks of the FPGA are always powered on and consume leakage power. Fortunately, the power analysis tool, that comes with the vendor toolset, can estimates this amount of leakage based on different operational conditions. However, the estimated amount of leakage is reported as a single value for the whole FPGA chip. This is useful when a power budget analysis for the FPGA is needed. However, for the thermal profile analysis of designs mapped to FPGA, one needs to have 1) an accurate model of leakage *distribution* across the FPGA device and 2) close the loop for leakage and temperature. With the lack of such information no accurate thermal profile estimation can be made.

In the following section, a method for distributing and increasing the reported leakage value across the FPGA is presented based on the relation between leakage power and tem-

perature, in such a way that hotter areas get more percentage of the leakage than the cooler ones. Afterward, a flow for steady-state thermal estimation using the proposed method is presented.

6.4. The Proposed Approach

6.4.1. Leakage Power Redistribution

Starting from the fact that MOSFET transistors leak more power under higher temperatures, a leakage-temperature-loop-based model is proposed to redistribute the leakage power to the FPGA blocks, in such a way that blocks which have higher temperature values get more percentage of the reported total leakage power than the other blocks. The traditional leakage-temperature relation model [75] is given in Equation (6.1). The proposed model, given in Equation (6.3), is inspired by the model in Equation (6.1) to determine the percentage of leakage power contributed by each block in the FPGA floorplan.

$$Factor_i = e^{-B/(T_i - C)} \tag{6.3}$$

where i is the block number, T is the block temperature, $Factor$ is a value, which represents a part of leakage power that the specified block will get, B and C are fitting parameters with positive values, which are obtained and calibrated through some generic test cases using the camera measurements, as will be explained later in Section 6.6.2. The leakage redistribution algorithm can be summarized in the following steps:

1. At the beginning, the leakage power ($P^{total}_{leakage}$) is distributed equally to the FPGA blocks, then HotSpot (as a thermal simulator) is called to get the first per-block steady-state temperature estimation (T_i).

2. Afterward, the model in Equation (6.3) is used to calculate the percentage of leakage that each block will get.

3. The original amount of leakage reported in the power report is then divided to the summation of all $Factor$ values to determine a normalized per-factor leakage.

4. Finally, the per-block leakage for each block ($P^{block_i}_{leakage}$) is calculated by multiplying the calculated per-factor leakage by the $Factor$ value of each block. In that way, the amount of the reported leakage power does not increase.

This loop continues until the steady-state temperature values of each block T_i converge over subsequent loop iterations ($\Delta T_i > \delta$, e.g. $\delta = 0.1°C$). A pseudo-code of this process is shown in Figure 6.2.

6.4.2. Temperature-Leakage Loop Estimation

In the previous section, the leakage power, which is reported in the power tool as a single value, is redistributed according to the temperature values across the FPGA chip. In this

```
 1. Read  P_leakage^total, n  ⇐  No_of_blocks
 2. P_leakage^block  ⇐  P_leakage^total  /  n
 3. For each block_i
 4.     P_leakage^block_i  ⇐  P_leakage^block
 5.     P^block_i  ⇐  P_leakage^block_i  +  P_dynamic^block_i
 6. End For
 7. Count  ⇐  0
 8. Do
 9.     Call HotSpot (P^block_i)
10.     Read Per_block Temperatures (T_i)
11.     For each block_i
12.         Factor_i  ⇐  e^(−B/(T_i−C))
13.     End For
14.     Factors  ⇐  Σ_{i=0}^{n}  Factor_i
15.     P_leakage^factor  ⇐  P_leakage^total  /  Factors
16.     For each block_i
17.         P_leakage^block_i  ⇐  Factor_i  ×  P_leakage^factor
18.         P^block_i  ⇐  P_leakage^block_i  +  P_dynamic^block_i
19.     End For
20.     Count  ⇐  Count  +  1
21. While (Δ T_i  >  δ)  AND  (Count  <  Count_max)
22. Return (P_leakage^block_i)_fixed  ⇐  P_leakage^block_i
```

Figure 6.2.: Redistribution of the leakage power across the FPGA chip according to temperature values

section, the resulted per-block leakage values $(P_{leakage}^{block_i})_{fixed}$ are allowed to increase, to reflect the real circuit operation (i.e. closing the temperature-leakage loop). The direct use of Equation (6.1), for this purpose, requires knowledge about the range of possible leakage power inside an FPGA block for a certain technology node. Without this knowledge, the parameters P_0 and k cannot be adjusted. In our case, a primary estimation for the per-block leakage resulted from the previous section is already there and hence, these low level details are not required. The new per-block leakage $(P_{leakage}^{block_i})_{new}$ is calculated according to Equation (6.4).

$$\left(P_{leakage}^{block_i}\right)_{new} = (\bar{A} \times e^{-\bar{B}/(T_i-\bar{C})} + 1) \times \left(P_{leakage}^{block_i}\right)_{fixed} \tag{6.4}$$

where $\bar{A}$, $\bar{B}$ and $\bar{C}$ are fitting parameters with positive values. Similar to Equation (6.3) the calibration of these parameters will be explained later in Section 6.6.2. The calculation of the steady state temperature happens in a loop similar to Figure 6.2 by replacing lines 12 - 17 with Equation (6.4). It should be noted, that the values of $(P_{leakage}^{block_i})_{fixed}$, which are obtained in Section 6.4.1 are fixed through the whole loop and do not change at each iteration. A pseudo-code of this process is shown in Figure 6.3.

```
1.  For each block_i
2.      Get (P_leakage^block_i)_fixed
3.      P^block_i  ⇐  (P_leakage^block_i)_fixed  +  P_dynamic^block_i
4.  End For
5.  Count  ⇐  0
6.  Do
7.      Call HotSpot (P^block_i)
8.      Read Per_block Temperatures (T_i)
9.      For each block_i
10.         (P_leakage^block_i)_new
            ⇐  (Ā  ×  e^(−B̄/(T_i − C̄))  +  1 )  ×  (P_leakage^block_i)_fixed
11.         P^block_i  ⇐  (P_leakage^block_i)_new  +  P_dynamic^block_i
12.     End For
13.     Count  ⇐  Count  +  1
14. While (Δ T_block  >  δ) AND (Count  <  Count_max)
```

Figure 6.3.: Leakage-temperature loop calculation

Actually, HotSpot simulator contains a built-in function intended to account for the temperature-leakage loop. However, the results of using this function do not make any noticeable change to the obtained thermal profile. We believe that this is because HotSpot has no information about the percentage of leakage power in the total power of each block.

6.5. FPGA Thermal Estimation Flow

In order to estimate the temperature profile across the FPGA chip, thermal simulators (e.g. HotSpot, ISAC) require two types of information: i) a floorplan of the chip and ii) a power trace file describing the power consumption (including both dynamic and leakage power) of each part of the chip. In the following, Xilinx's design tools are targeted, because the proposed experimental setup utilizes a Virtex-5 FPGA (see Section 6.6). Nevertheless, the same flow can be easily adjusted for other design tools from other FPGA vendors.

A method similar to the one presented in [74] is followed for preparing the needed FPGA information to be used in the targeted thermal simulator. Additionally, the HotSpot 5.02 [72] simulator is used for thermal modeling.

6.5.1. Floorplan Creation

The FPGA's floorplan is generated using 1) the information from the chip resource file (*.xdlrc) and 2) the die dimensions (see floorplan creation in Figure 6.5). The information provided from the xdlrc file divides the FPGA chip into two-dimensional array of physical *tiles*. Each tile contains a set of the FPGA internal components. For the generation of the floorplan, it would be possible to divide the floorplan exactly as the FPGA is divided in the xdlrc file (i.e. each tile from the xdlrc file is mapped to one block of the floorplan). However, the number of FPGA tiles is usually large (e.g. 164×177 for Virtex-5 110t), which

can result in excessively long thermal simulation runtime. Fortunately, the temperature has a negligible variation at such granularity. For that reason, the FPGA die is divided into a two-dimensional set of identical blocks, where each block has a dimension of $n \times m$ of physical tiles (it is found that a block size of 5×5 of physical tiles gives the best trade-off between temperature accuracy and simulation runtime). The FPGA die dimensions, which are obtained either from the FPGA datasheet or from manual measurements, are used then to calculate the dimensions of each physical tile (assuming all tiles are identical in size), and in turn to calculate the dimensions of each block in the floorplan.

6.5.2. Power Trace Generation

The generation of the power trace file, on the other side, requires detailed information about the power distribution across the FPGA for the targeted mapped design. FPGA designers can get detailed power estimation for a certain design using the vendor's power analyzer tool. In our case, Xilinx's *XPower* is used to generate the detailed power report. The generation of accurate power values, for different parts of the circuit, requires detailed information about the internal signal activities. These can be obtained using a logic simulator (e.g. ModelSim) for the post-place-and-route circuit model (see Figure 6.4). The internal signal activities can be saved in a *value change dump* (VCD) file and then can be given to the power tool. The power report, generated in that way, contains a detailed dynamic power description for each *node* in the design (logic (LUT), routing signals, clocks, etc.).

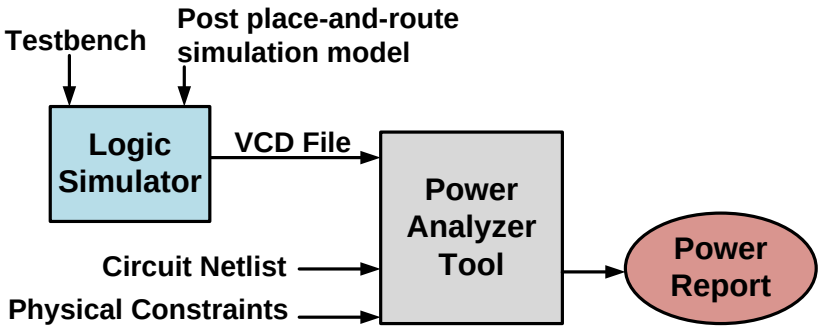


Figure 6.4.: Power report generation flow

The power report is parsed to extract the dynamic-power value for each node in the design. Then the physical locations for each node is obtained from the physical mapping information (the design's XDL file), and mapped to the chip resource file (XDLRC file) to find the exact physical tile place on the FPGA die, and in turn to find the exact block place in the floorplan. If a node occupies more than one tile (which is the case in the routing signals), its power is distributed to the tiles, in such a way that the tiles that drive a larger number of other signals (i.e. have a high fan-out) get more percentage of the power. Such

information can be obtained from the design' XDL file. Afterward, for each block in the floorplan, the values of the dynamic power of the tiles inside that block are summed up to get the block's dynamic power value. Figure 6.5 shows an overview of this flow.

For the leakage power, the method presented in Section 6.4 is followed for distributing and increasing it across the FPGA chip. After that loop converges, the thermal profile is then obtained. For the leakage power, the amount reported in the power report is given as a single value for the whole chip and not per node, unlike the dynamic power. For that reason, taking into account that the FPGA chips, in general, has regular structure, the first intuitive idea that may come to mind to get a rough estimation for the per-block leakage, is to divide the leakage power equally to all the blocks. However, when the temperature results from the simulation are compared with real measurements from infra-red camera as we will see in Section 6.7, the results did not match. Therefore, it is necessary to accurately redistribute the leakage power in a more proper way.

6.5.3. Thermal Profile Estimation For Dynamically Changing Circuits

The output of the thermal profile estimation flow, described in the previous subsections, is the steady-state thermal behavior of the FPGA-mapped circuit. However, in reconfigurable applications in which some portions of the configuration change over time, the thermal behavior of the circuit may not have a single steady-state case. In other words, for each partial or full reconfiguration the steady-state thermal profile of the FPGA can be different. The proposed flow can be easily extended to handle reconfigurable systems. In fact, it is only required to perform a thermal profile analysis for each of the reconfiguration options of the circuit, which are typically known at the design time (e.g. the bitsream, floorplan, and all other required details are available at design time).

On the other hand, the FPGA-mapped circuit itself can have different signal activities through the lifetime operation. This means that its thermal profile is dynamically changing through the lifetime operation, which necessitates a dynamically changing thermal profile estimation to reflect the circuit operation. For this purpose, the following approach is proposed:
The simulation time of the post-place-and-route model is divided to a series of time windows. Then, for each time window a separate VCD file can be generated. Afterward, using the FPGA's power tool, for each of the generated VCD file a separate power report can be generated. Finally, using the proposed thermal profile estimation flow, for each of the generated power report a separate thermal profile is generated, in such a way that the primary input temperatures used in the estimation of the leakage power is simply the steady-state temperatures of the previous thermal profile estimation. The details are as follow:

- Determine a time window (t^w) in which a separate VCD file for the signal activities should be captured. In such a way the total simulation time is divided into several time windows from t_0^w to t_n^w

- Using the logic simulator, for each time window (t_i^w), generate a separate VCD file (VCD_i).

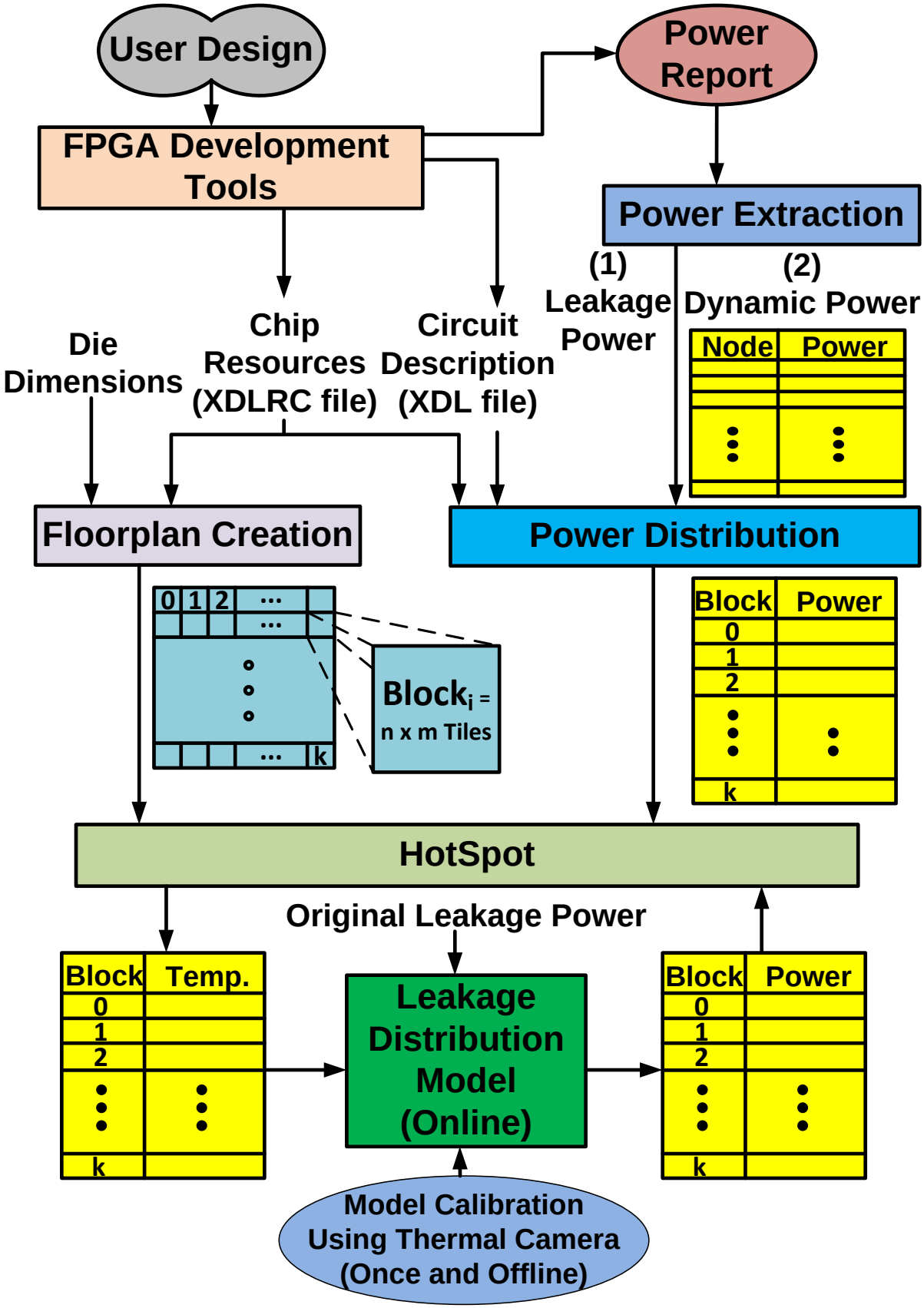


Figure 6.5.: An overview of the proposed temperature estimation flow

- Then, using the power tool, for each VCD file (VCD_i), a separate power report (PR_i) is generated for the mapped design.
- Afterward, following the proposed thermal-profile estimation flow, for PR_0 generate the thermal profile (TP_0)
- Then, for each of the following PR_i, use the previous thermal profile (TP_{i-1}) as initial temperature values to generate the new thermal profile (TP_i).

6.6. Experimental Setup

6.6.1. Thermal Camera Usage

To accurately adapt, analyze and validate the presented approach, a thermal camera is employed[3] that monitors the infrared emissions from the back side of an FPGA's die. In contrast to the widely used method of distributing a large number of ring oscillators across the chip [76] to build the thermal map of the FPGA over runtime, this setup enables us to obtain accurate measurements of the chip's temperature at a much higher spatial resolution, especially considering that key factors, such as the peak temperature, can be missed by on-chip sensors if there are high spatial thermal gradients. Another advantage is that this method is non-intrusive meaning that there is no need to modify the user application (bitstream) to include soft sensor arrays. For more details about this setup, the reader is referred to [77].

6.6.2. Model Calibration

To obtain the values of the fitting parameters B and C in Equation (6.3) and $\bar{A}$, $\bar{B}$ and $\bar{C}$ in Equation (6.4), a generic circuit that utilizes a variety of the FPGA components (LUTs, FFs, DSPs, BRAMs, and DCMs) is used. This circuit is designed in such a way that it can operate under different frequencies (33 - 600 MHz). Thus, the maximum possible range of temperatures can be generated. For each operating frequency, a separate thermal profile of the circuit is captured with the thermal camera. Then, the simulation results, using the proposed approach, are fitted with their corresponding camera measurements for each frequency to obtain the fitting parameters (while the model is trained and tuned). Afterward, the same values for these fitting parameters, which are obtained in this step, are used with all other benchmark circuits in this chapter. It should be noted that this calibration process is needed only one time for each targeted FPGA chip.

6.7. Experimental Results

In order to validate the proposed approach, several designs with various sizes and logic densities are tested with the thermal camera, and compared the results of the simulation

[3] I acknowledge here Prof. Jörg Henkel and his group from the chair of embedded systems at Karlsruhe Institute of Technology who have done the thermal camera setup

with the measurements of the camera. Table 6.1 lists these circuits and their properties. The first circuit (8-bit *Finite Impulse Response* (FIR) filter) represents a lightweight FPGA load case. The second circuit is 128-bit pipelined AES encoder, which represents a medium FPGA load case. The remaining four circuits, are used to show extreme load cases, where a set of LUTs, FFs, DSPs, and BlockRAMs are designed in such a way that they keep switching at their inputs and/or outputs in a continuous manner, so that maximum dynamic power consumption for each case can be obtained. For each circuit a test-bench is integrated in the design to independently generate random vectors for the inputs of the circuit. The circuits are synthesized and built for Virtex-5 FPGA (XC5VLX110t-1FF1136) using Xilinx ISE 12.2.

The thermal camera is used to record the thermal behavior of each circuit loaded into the FPGA. At the beginning, the FPGA was left switched on for long period before loading the circuits, to allow it to heat up and reaches a stable temperature. Afterward, each configuration file is loaded to the FPGA and the thermal behavior is recorded using the camera until the temperature values stabilized. At the end, the FPGA is reset but left switched on until its temperature become stable, then the next design is loaded, and so on until all the circuits are tested. For the thermal simulation, the method described in Section 6.5 is followed for estimating the thermal profile of each design. HotSpot simulator is configured to represent the same experimental conditions of the camera, where the packaging of the FPGA is removed. The leakage distribution loops in Section 6.4.1 and 6.4.2 converges within 6-15 iterations for all designs. The obtained thermal profiles from the thermal camera and the proposed simulation flow for two of the benchmark circuits are shown in Figure 6.6.

Figure 6.7 shows a comparison between the measurements of the thermal camera, for "Toggling LUTs-FFs" circuit, and the resulted thermal profile for each of the three steps of the approach presented in Section 6.4. Figure 6.7-(e) shows the stable leakage values across the chip after applying the proposed approach.

To compare the thermal profiles obtained from the simulation results using the proposed approach and the camera measurements, the difference range, between any two points on the FPGA chip, is considered to be the measure for the accuracy of the simulation. Five points for the comparison have been set as seen in Figure 6.7-(a). The temperature differences between the peak temperature and the five comparison points (P1 - P5) are reported in Table 6.2 for each of the benchmark circuits in Table 6.1. The average absolute estimation error is about 1.0°C for temperature variation across the FPGA, which is represented by the five comparison points (P1 - P5), while the absolute estimation error for the peak (max) temperature for each circuit is 3°C. These results show that the proposed flow can provide accurate thermal estimation results at the design time, which helps to determine and avoid critical hot spots early in the design phase (i.e. before loading the design to the FPGA for in-field operation).

6.8. Summary

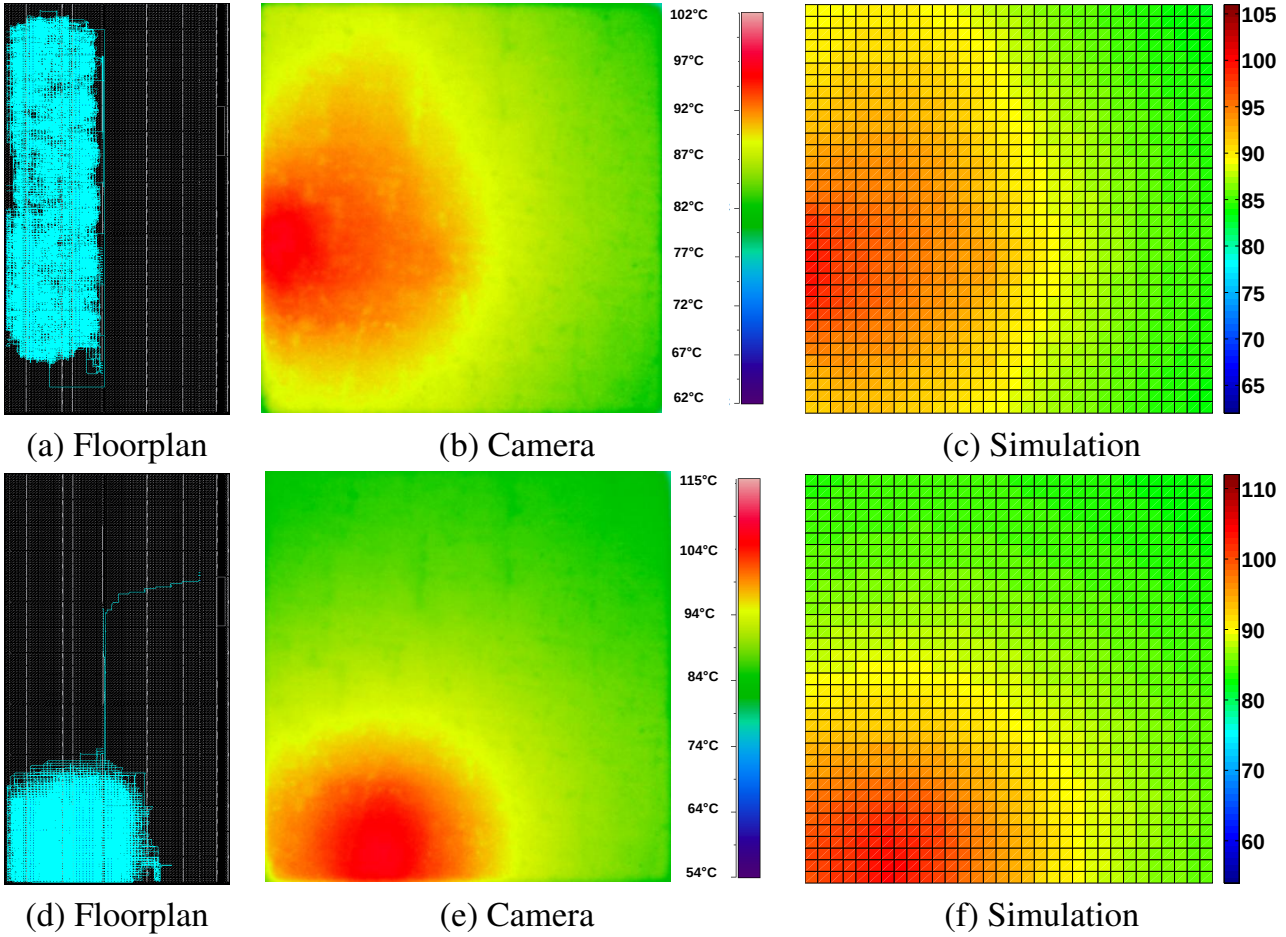
Due to the lack of detailed information about how the leakage power is distributed across the modern FPGAs, the traditional method to estimate the thermal profiles suffers from inaccuracy. In this chapter, a method for properly distributing the leakage power across the

Table 6.1.: Properties of the circuits used in the thermal comparison

Circuit	Clock Frequency	Size	Other FPGA Components
FIR filter	152 MHz	1699 LUTs / 528 FFs / 708 Slices	1 DCM / 17 IOBs
AES encoder	294 MHz	10786 LUTs / 7769 FFs / 3185 Slices	1 DCM / 129 IOBs
Toggling LUTs	no clock	3840 LUTs / 0 FFs / 960 Slices	1 IOBs
Toggling LUTs-FFs	100 MHz	3293 LUTs / 1035 FFs / 874 Slices	3 IOBs
Toggling LUTs-FFs-DSPs	100 MHz	7913 LUTs / 3229 FFs / 2298 Slices	12 DSPs / 18 IOBs

Table 6.2.: The comparison results between the thermal camera measurements and the thermal simulation using the proposed approach

Circuit	Camera Measurements [$^{\circ}C$]						Simulation Results [$^{\circ}C$]						Average abs. Estimation Error [$^{\circ}C$]
	Max	Deviation from the Max					Max	Deviation from the Max					
		P1	P2	P3	P4	Center		P1	P2	P3	P4	Center	
FIR filter	49.9	2.7	2.7	2.9	2.6	1.4	49.2	1.6	1.9	1.7	1.2	0.3	1.1
AES encoder	96.6	8.0	13.9	14.2	9.2	7.3	99.4	7.9	15.6	16.1	10.3	8.5	1.2
Toggling LUTs	60.5	7.6	1.6	6.7	8.1	5.4	57.1	8.9	0.5	8.8	9.6	7.4	1.6
Toggling LUTs-FFs	57.1	2.6	6.0	7.5	7.1	4.5	55.0	1.0	6.5	7.5	6.9	5.2	0.6
Toggling LUTs-FFs-DSPs	107.1	7.4	19.1	23.1	21.6	15.2	104.1	3.9	18.7	21.7	20.3	15.8	1.4



(a) Floorplan (b) Camera (c) Simulation

(d) Floorplan (e) Camera (f) Simulation

Figure 6.6.: Comparing the obtained thermal profiles between the measurements of the infrared camera and the proposed simulation flow: (a-c) AES encoder, (d-f) Toggling LUTs-FFs-DSPs. *Please note that there is a little difference between the color coding used in the software of the camera and the color coding of Matlab, in which the simulation profiles are drawn.*

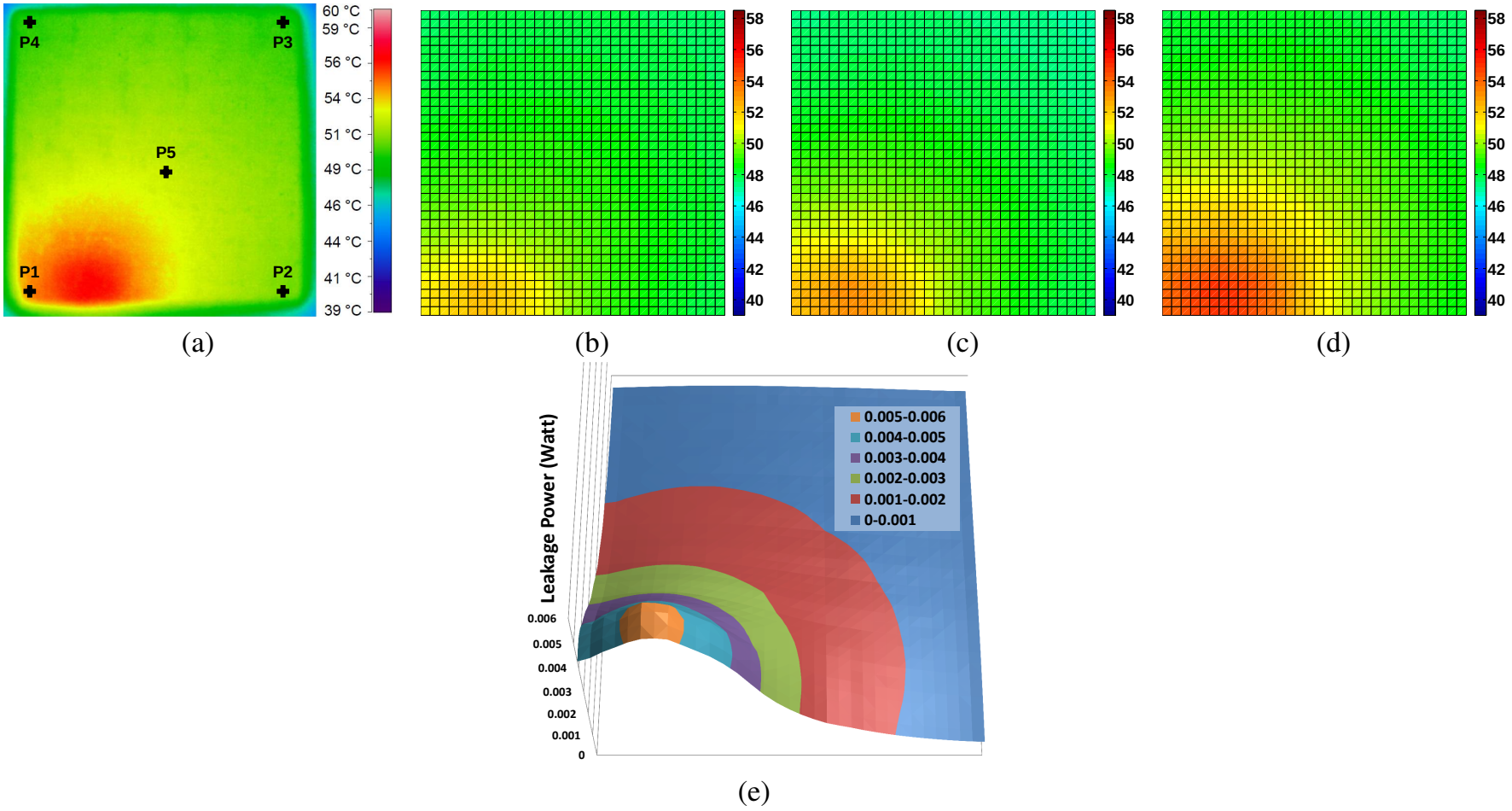


Figure 6.7.: Comparing different leakage distribution models. (a) Infrared camera measurement, (b) The total leakage is equally-distributed across the FPGA, (c) The total leakage is redistributed according to temperature values (as in Section 6.4.1) and (d) (e) The leakage-temperature loop is applied (as in Section 6.4.2)

FPGA chip has been presented. The method uses a temperature-leakage loop estimation model for distributing the leakage power. The model is calibrated and validated using real measurements from an infrared camera, which measures the emissions from the backside of Virtex-5 FPGA chip. The comparison results between the camera measurements and the simulation of several designs, with different sizes and frequencies, showed that the proposed approach can achieve accurate thermal-profile estimation, with average absolute estimation error of around 1°C across the FPGA chip.

Chapter 7.

High-level Aging Estimation

7.1. Introduction

To make the FPGA-based circuit designers capable of analyzing the impact of transistor aging on their designs, they have to consider BTI and HCI-based aging models. In this case, the aging of the mapped design depends both on the FPGA device characteristics and the mapped design netlist. However, the currently available models are at device level [16, 15, 45], which makes the estimation of the aging for a complete FPGA-mapped circuit using such models an infeasible job. This is mainly because of the simulation overhead at device-level and its complexity. Additionally, some of the main factors that influence the aging are originated from higher-levels (e.g., it is extremely usage, workload and temperature dependent), which requires access to higher-level details for correct estimation at device-level. Aside from that, in FPGAs most of the device-level details are proprietary to the manufacturer and the users have access only to the final fabricated chip. This makes the aging estimation, for the designers, an unknown and inaccurate process, and hence, the designers may overestimate it, which results in performance loss, or underestimate it, which may end with an unreliable design. Based on that, the need for fast and accurate aging estimation at higher level emerges, even if it is inherently less accurate than the device-level correspondent.

In this chapter, an abstraction for both the BTI and HCI device level models to RTL models is proposed, which allows faster aging estimation for the FPGA-mapped designs than the device-level models. Based on these abstracted models, an aging estimation/prediction tool for designs mapped into state-of-the-art FPGAs is presented. This tool considers both the mapped design (logic-level netlist as well as placement and routing) and FPGA device characteristics. The key point of this tool is that it infers the needed FPGA device information from the available FPGA timing and power reports provided by the FPGA design toolset, and also from the post-place-and-route simulation model, without the need for detailed device-level/circuit-level manufacturing info/parameters, that is usually needed for the BTI and HCI models. Moreover, the tool uses both thermal modeling and logic simulation of the FPGA mapped design, to obtain accurate stress/relaxation information, which is needed for aging analysis. The tool is implemented based on the information provided by Xilinx's FPGA design tools. Nevertheless, it can be easily adjusted for other FPGA design tools from other vendors.

The tool can be used to evaluate, predict or estimate the amount of aging-induced degradation, and hence the lifetime of the FPGA-mapped designs. It also identifies aging-vulnerable nodes in the design. Designers can obtain the impact of different designs, mapping styles, usage, etc. on the wear-out (lifetime) of their circuits mapped into the FPGA.

So, along with existing toolset for power, performance and area analysis, the designers would be able to co-optimize the wear-out along with other design constraints.

As a case study of the tool usage, the tool have been examined with several ITC'99 test-bench circuits using a Virtex-6 FPGA, to explore the effect of the design parameters on the amount of expected circuit aging. The results show that the amount of aging-induced degradation is strongly dependent on the circuit mapped to the FPGA device, so that different circuits loaded to a single FPGA device have different lifetimes. Also, different mapping and optimization algorithms for the same circuit show tendencies of different aging rates.

The rest of this chapter is organized as follows: In Section 7.2 the related works are reviewed. Afterwards, in Section 7.3, the proposed aging modeling tool is described. The validation of the abstracted device model followed by the experimental results are both discussed in Section 7.4. At the end, Section 7.5 concludes this chapter.

7.2. Related Work

There are many papers which discuss the effect of BTI and HCI-induced degradations and their estimations. However, only few works presented a complete framework/tool for degradation assessment. For NBTI degradation assessment in microprocessors, the authors in [78] have presented a framework called *New-age* that takes as input parameters the netlist, the technology node used, the operating conditions of the circuit (voltage, temperature, etc.) and the probabilities of the inputs, to calculate the aging estimation due to NBTI. The estimation is done at gate level. A similar framework is presented in [17]. The authors in [69] have also presented a framework for estimating the Mean-Time-To-Failure (MTTF) due to the effects of NBTI, TDDB and EM in FPGAs. The framework depends on a 65 nm PTM model to get the needed device level information for the estimation, because the FPGA device level information is proprietary. It was not clear how the circuits that represent the different elements of the FPGA are obtained to perform the estimation at their transistors.

The main difference of the tool proposed in this chapter to the aforementioned frameworks, is that the proposed tool does the aging estimation for both BTI and HCI phenomena at system-level yet device/technology-level relevant, not at transistor or gate-levels. This eliminates any need for detailed device-level information of the FPGA, as this information is usually not available (proprietary). Furthermore, the estimation at system-level is easier, faster and meets the accuracy needs of the FPGA designers to decide whether a certain mapping / routing / placement technique is better than another in terms of aging.

7.3. Methodology

The methodology for implementing the aging estimation tool for FPGAs can be divided into three parts: i) the abstraction of the device level models of both BTI and HCI, ii) the inference of the required information from the power and timing reports of a certain design as well as the thermal model, and iii) the aging estimation process.

7.3.1. Aging model abstraction

In order to achieve an accurate estimation for the amount of aging-induced degradation, without the need for detailed device level information, a high level abstraction for the device level models for both BTI and HCI is needed. However, this abstraction must use the most detailed information that can be gained from various sources, such as logic-level simulations, detailed physical design (placement and routing) information, thermal profile and the FPGA's power and timing reports, to achieve the highest possible accuracy. By analyzing the FPGA reports, it can be seen that the finest details are given at signal and logic level. Therefore, the abstraction is done at this level, which is called *node level*. A single node can represent an LUT, a flip-flop, a routing signal, a half adder, etc. (see Fig. 7.1). This node separation is taken at the granularity of what the timing report can provide. A path in the circuit from an input flip-flop to an output flip-flop consists of a set of nodes. It should be noted that several paths can pass through a single node at the same time (e.g. a 6-inputs LUT can have up to six different paths crossing it).

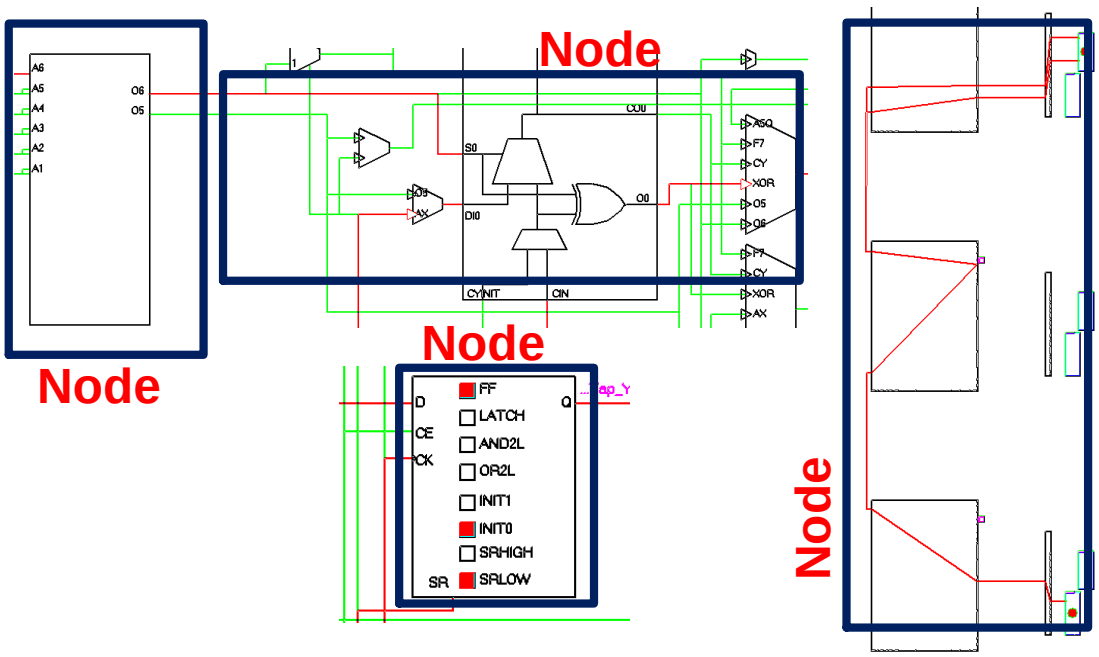


Figure 7.1.: Different nodes in the FPGA

For a certain node, all the needed information for aging estimation (e.g. signal probabilities, signal activities, initial delays, etc.) can be gained from the FPGA reports for each and every input and output of that node separately. In other words, the device-level details, which are needed for the aging estimation, are already given for the inputs and the outputs of any certain node in the FPGA. However, the device-level details of the node's internals are not known for the FPGA designer. In the proposed model, it is get used of this information at the inputs and the outputs of each node to estimate the internal details

of that node. The idea is to represent each internal path from an input of a certain node to an output of that node by a series of blocks. Each block is represented by an inverter in the corresponding technology node used in the targeted FPGA, so that it contains the impact of the technology node as well as the aging effect on both NMOS and PMOS transistors. The number of blocks (m) in the series is determined according to the delay of a representative inverter in the technology node, such that m = delay of the internal path / delay of a single inverter. In that way, the logic depth of the internal path is represented (see Fig. 7.2). The validation of this model is discussed in Section 7.4.1.

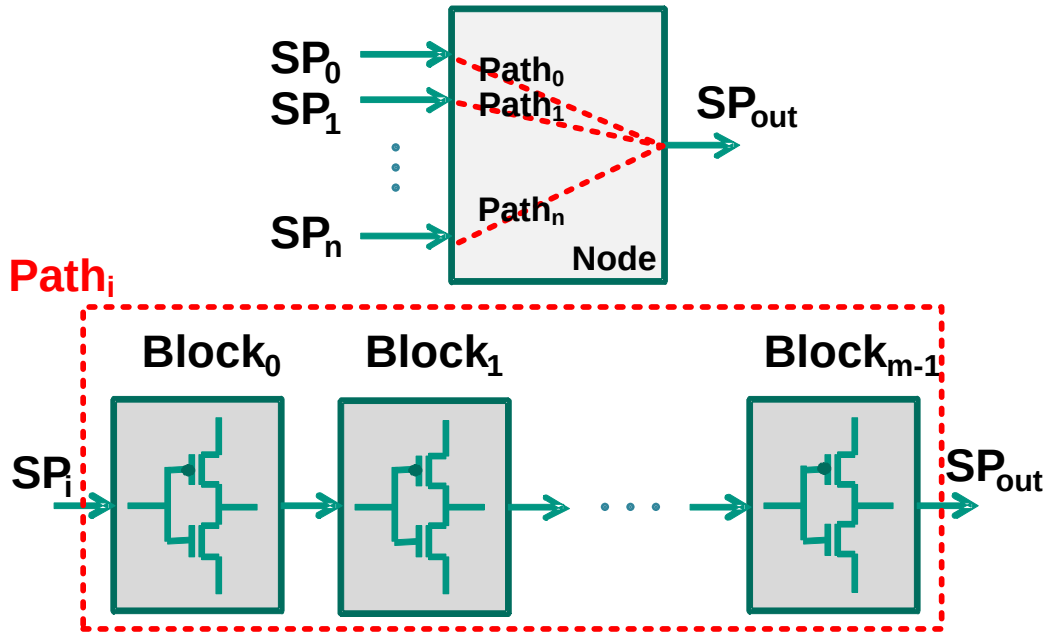


Figure 7.2.: Different paths through a node and a representation of a single path by a series of blocks

The internal details, at the input of each block, are estimated as a function of the series' input and output (which are given from the FPGA reports for the corresponding node) in addition to the block position in the series. For example, for estimating the NBTI effect, the signal probability (SP), at the gate of a certain transistor, is required to calculate the duty cycle (Y) of that transistor as given in Equation (2.4). By logic simulation, all signal probabilities, at the inputs/outputs of each node in the circuit, can be calculated. The internal SP for each representative block in Fig. 7.2, according to the proposed model, can be written as:

$$SP_j = \left\{ \begin{array}{ll} SP_i + j * \frac{|SP_{out} - SP_i|}{m-1}, & SP_{out} \geq SP_i \\ SP_i - j * \frac{|SP_{out} - SP_i|}{m-1}, & else \end{array} \right\} \tag{7.1}$$

where $j \in \{0, 1, 2, ..., m-1\}$ is the position of the block in the series. Similarly, for HCI (see Equation (2.5)), the activity rate (*AR*) can be read from the power report for each input/output of every node in the design. The internal *AR* can be then estimated in a similar way like (7.1). By representing the logic depth, the transistor types, and the delays in the same technology node used in the targeted FPGA, in addition to estimating the internal details (*SPs* and *ARs*) from the known information at inputs and outputs of the node, this cascaded block model can better reflect the aging of internal elements of the node.

7.3.2. Information gathering

The aging estimation tool is based on the abstracted models of BTI and HCI presented in the previous section, and uses these models to calculate the amount of aging-induced degradation. The input parameters for these models are taken from the information provided by:

A) The timing report to obtain the pre-aging critical paths with their breakdowns of their nodes.

B) The detailed physical design information, for obtaining the nodes' information and the breakdown of the logic-level netlist to nodes with FPGA coordinates.

C) Logic simulations for the placed and routed design (with nodes' information) to obtain the duty cycle and activity rates of each input/output of every node.

D) The thermal model, which uses the power report, physical design information (B) and signal activities (C) to obtain nodes' temperatures.

The tool overview is depicted in Fig. 7.3. Each part of the tool is discussed in detail in the following subsections.

7.3.2.1. Determination of nodes and their delays

The first step of the tool, is to determine the nodes and the pre-aging delay of each path from an input to an output of every node in the finally placed and routed design. This information is taken from the timing report (see Fig. 7.4). The timing report is generated, in our case, using Xilinx's timing analyzer tool, by specifying a certain number of paths per constraint ($P_{Constr.}$). The value of $P_{Constr.}$ depends on the size of the design, and has to be given such that all the paths that have a slack of $x\%$ are reported (x is the maximum expected degradation during the lifetime of the FPGA, e.g. 15%). A parser is used afterward to extract these paths with their nodes breakdown and also their pre-aging delays.

7.3.2.2. Determination of activity rates

The activity rate for each input/output of every node (*AR*) is determined from the power report of the finally placed and routed design. For accurate activity rate calculation, Xilinx's Xpower analyzer needs the *Value Change Dump* file (*.vcd) of the design's simulation. With a logic simulator (e.g. Modelsim), the vcd file can be generated using the post-place-and-route simulation model of the design. However, a testbench that reflects a realistic

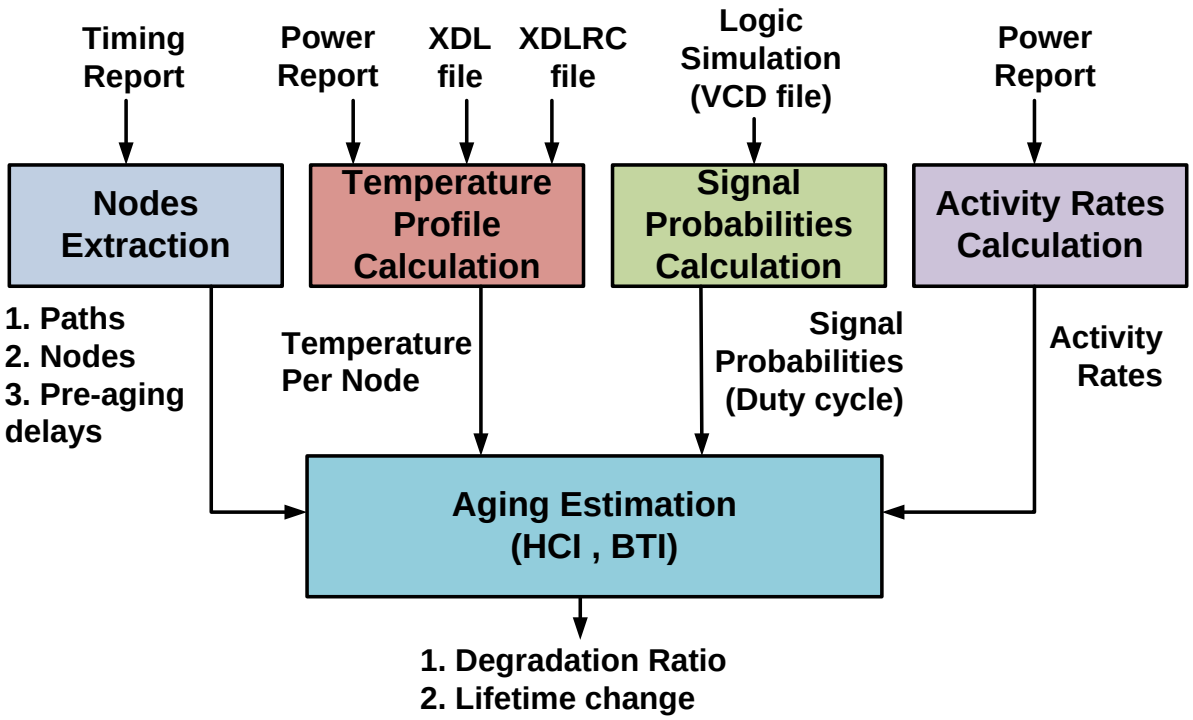


Figure 7.3.: Overview of the proposed aging estimation tool

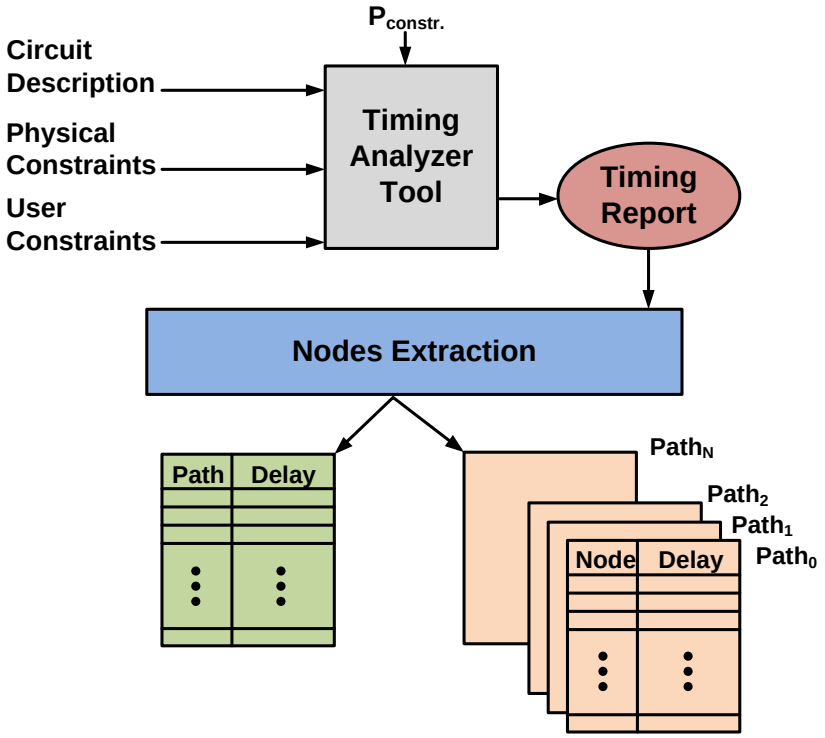


Figure 7.4.: Determining the nodes in the design and their pre-aging delays

operation of the desired circuit must be used for the simulation, thus real values for the activity rates can be reported, which results in a more accurate power estimation of the mapped circuit. A parser is used afterward to read the generated power report, and to extract the nodes and the activity rates at their inputs and outputs. An overview of this process is shown in Fig. 7.5.

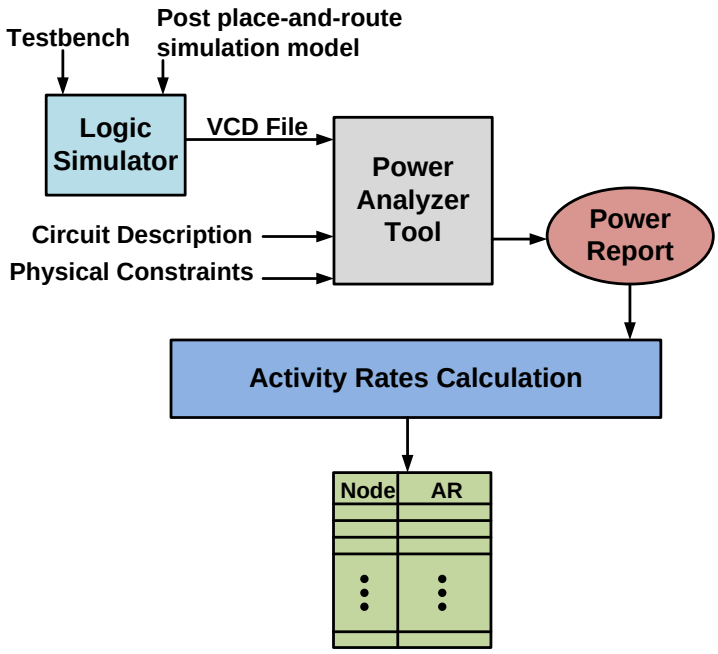


Figure 7.5.: Calculating the activity rate of the nodes

7.3.2.3. Determination of signal probabilities

A separate routine is used for this purpose, it parses the vcd file generated in the previous section, then it creates a table with all nodes, and assigns for each node 3 counters to count the times in which the node's input was one, zero or unknown respectively during the simulation. Afterward, the probability of having one at the input is calculated for all nodes, and saved in a separate table. The calculated signal probabilities (SPs) are used afterward to calculate the duty cycle (Y) of each input/output of every node, such that for NBTI: $Y = 1 - SP$, and for PBTI: $Y = SP$.

7.3.2.4. Temperature profile calculation

The method presented in Chapter 6 is used here to estimate the thermal profile of the FPGA-mapped design.

7.3.3. Aging estimation

At this stage, all the necessary pieces of information for applying the abstracted BTI and HCI models to the nodes are available, and the aging estimation can be made. The estimation process can be simply summarized by three main steps: i) finding the amount of change in node's delay due to aging ($\Delta d_{(node)}$), then ii) calculating, for each node, the post-aging delay $\left(d_{0(node)} + \Delta d_{(node)}\right)$. Afterward, iii) finding the post-aging critical path delay ($\tilde{d}_{critical}$), and compare it to the pre-aging critical path delay ($d_{critical}$), to determine the percentage of performance loss due to aging. The details are as follows:

A) Each path in the design P_i contains nodes g^i_1 to g^i_N, and each node g^i_j has a pre-aging delay $d_{0(node^i_j)}$. This information is already stored in the table presented in Section 7.3.2.1.

B) The pre-aging delay of each path P_i is calculated from $d^i_0 = \sum_{j=1}^{N} d_{0(node^i_j)}$

C) The pre-aging critical delay $d_{critical}$ is the maximum d^i_0 for all paths.

D) Now, for each node g^i_j: i) Find its activity rate $AR_{(node^i_j)}$ from the table in Section 7.3.2.2. ii) Find its duty cycle $Y_{(node^i_j)}$ from the table in Section 7.3.2.3. iii) Find its temperature $T_{(node^i_j)}$ from the table in Section 7.3.2.4.

E) The amount of change in node's delay due to aging ($\Delta d_{(node)}$) can be then calculated for each node. For HCI: i) For the maximum frequency of the targeted FPGA ($AR = max$), assume $x\%$ maximum (worst-case) HCI delay increase in z years[1], under ambient temperature T, and for a representative inverter delay d_0. Solve in (2.5) to obtain A_{HCI}. ii) For each node g^i_j, generate the representative series of blocks as discussed in Section 7.3.1. iii) Having the activity rates at the input and output of the node, calculate for each block the internal activity rate AR similar to (7.1), then calculate the change in the block delay using (2.5). iv) The summation of the block delays gives the new delay of the node due to HCI.

F) Similarly, for BTI: i) For $SP = 1$ (NBTI) or $SP = 0$ (PBTI) Assume $y\%$ maximum (worst-case) BTI delay increase in z years[1], under ambient temperature T, and for a representative inverter delay d_0. Solve in (2.4) to obtain A_{BTI}. ii) For each node g^i_j, generate the representative series of blocks as discussed in Section 7.3.1. iii) Having the signal probabilities at the input and output of the node, calculate for each block the internal signal probability SP as in (7.1), then calculate the change in the block delay using (2.4). iv) The summation of the block delays gives the new delay of the node due to BTI.

G) Afterward, for each node g^i_j, the post aging delay is simply calculated $d_{0(node^i_j)} + \Delta d_{(node^i_j)}$.

[1]The values for x, y and z can be taken from user, or alternatively, from real measurements (e.g. as done in Chapter 5 or in [53])

H) At the end, for each path P_i, the post aging delay is calculated $d^i = \sum_{j=1}^{N} \left(d_{0(node^i_j)} + \Delta d_{(node^i_j)} \right)$

I) The post-aging critical delay $\tilde{d}_{critical}$ is the maximum d^i for all paths.

Having both the pre-aging critical delay $(d_{critical})$, and the post-aging critical delay $\left(\tilde{d}_{critical}\right)$ of the FPGA-mapped design, the percentage of delay degradation due to aging can be simply calculated $aging\ rate = \frac{\tilde{d}_{critical}}{d_{critical}} - 1$.

7.4. Experimental results

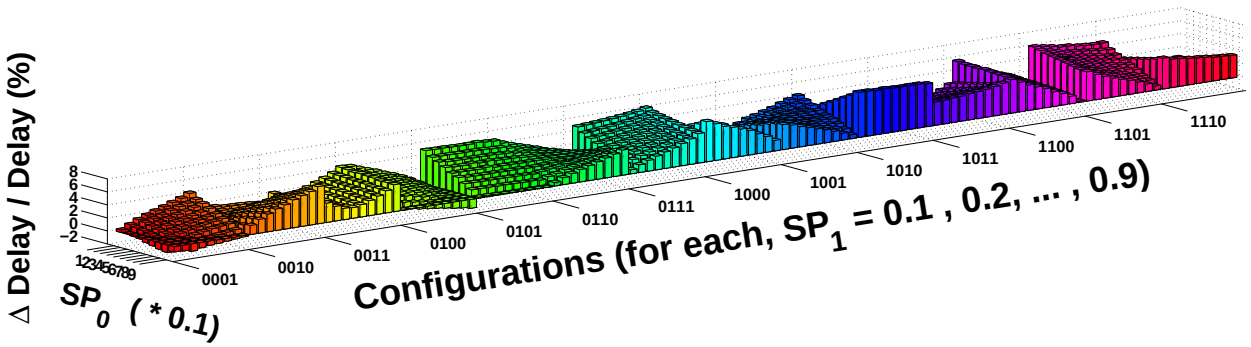
7.4.1. Validation of the abstracted aging model

To validate the model proposed in Section 7.3.1, an HSPICE device level simulation, for a 2-input LUT and for a half adder circuit, is carried out using a 22 nm predictive technology model. The same circuits and method presented in Chapter 3 are used to emulate the effect of aging on the transistors. For each input of the LUT a set of 9 different SPs (0.1, 0.2, ..., 0.9) and their permutations (9*9) are tested for each possible LUT configuration, then the relative delay increase is recorded for each case (see Fig. 7.6(a)). On the other side, using Matlab, all possible SPs of the LUT's output (SP_{output}) are calculated, then for the path that represent the first input of the LUT to the output, the proposed model is applied. Afterward, for each set of both first-input's SP (SP_0) and SP_{output}, the relative delay increase due to NBTI is recorded and compared to the HSPICE results (see Fig. 7.6(b)). The comparison results show that, for 14 configurations out of the possible 16, and for all possible permutations of input and output *SPs*, the trends of NBTI degradation between the model and the HSPICE simulation match. For the remaining 2 configurations, only parts of the permutations of input and output *SPs* yield similar NBTI degradation trends. In total, around 90% of the results match the trend (see Fig. 7.6).

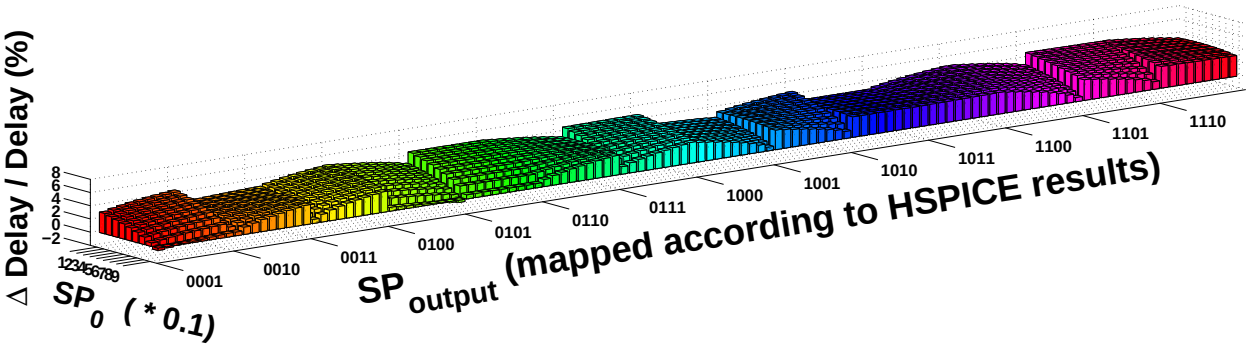
It should be noted that the proposed model is to abstract relatively small logic circuits, not large ones. This is thought for the basic elements of the FPGA (nodes). Furthermore, these results show that using the abstracted model alone may not be enough to have an accurate design margin determination. However, for judging whether a certain implementation of the design is better than another, the proposed model gives the needed accuracy. In fact, with the lack of the device level information of the FPGA, this abstraction provides an easy and fast way for aging estimation, which may also be the only available way for FPGA users.

7.4.2. Case study: influence of mapping and optimization algorithms

The idea is to explore the effect of different design parameters on the amount of estimated aging-induced degradation. Therefore, several ITC'99 testbenches have been tested with different mapping and optimization algorithms. The aging estimation tool is built based on the information provided by Xilinx's FPGA development tools. Virtex6-vlx75t is used as a targeted FPGA for estimating the aging of different designs onto it, because it can fit all the



(a) HSPICE simulation



(b) Abstracted model

Figure 7.6.: Validating the abstracted model with HSPICE simulation for PT-based 2-input LUT, for each input 9 SPs from 0.1 to 0.9 are tested (as Chapter 3

selected testbench circuits. All circuits are forced to be mapped only to LUT and flip-flops, thus a fair comparison can be made. Furthermore, as PBTI effect is similar to NBTI, only NBTI effect is considered in the following. However, the extension of the proposed flow for PBTI is quite straightforward. Three experiments are made for each circuit: the first one is by making the ISE tool choose the best mapping, placing and routing conditions with speed optimization (the default options). The second one is by forcing the placement to a corner of the FPGA with speed optimization. The third experiment is to force area optimization with freely mapping, placing, and routing. In all the experiments, the maximum amount of degradation (worst case), used in Section 7.3.3 to get the values of A_{BTI} and A_{HCI}, is set to be 15% in 5 years for $SP = 1$ (PMOS transistors always ON $\rightarrow$ worst case NBTI) and $AR = 500$ MHz (maximum frequency $\rightarrow$ worst case HCI). 10000 vectors are tested for each experiment to determine signal probabilities and activities. The degradation ratio due to NBTI and HCI are reported in Table 7.1 for each experiment. The explanation for the low HCI-induced degradation ratios compared to the NBTI ones is because, for the HCI case, the tested circuits operates with frequencies less than the maximum possible frequency on the FPGA ($AR = 500$ MHz) which is considered the worst case as mentioned before.

It should be noted that the amount of degradation has a direct relation with the circuit behavior (i.e. signal activities and probabilities) as described in Section 7.3.2.2. Therefore, it is necessary to write a realistic testbench for each circuit to get an accurate circuit behavior. The testbenches for the ITC'99 circuits are written, taking into consideration the original functionality of each circuit that is reported in [79]. Nevertheless, for b14, as it represents a sub-circuit of a processor, it was not practical to write a realistic testbench. Therefore, random vector inputs are considered for it.

It's also important to mention that a small change, in the degradation ratio of BTI or HCI-induced aging, can result in a significant change to the operational lifetime of the design. For example, according to the NBTI device-level model (see Equation (2.4)), if two degradation ratios (r_0 and r_1) are considered for a certain PMOS transistor, where $r_0 < r_1$ (see Fig. 7.7). The required time (transistor age) until a transistor with a degradation ratio r_0 reaches a critical delay will be significantly different from the same transistor with degradation ratio r_1. In a similar way, a rough estimation can be made to calculate the percentage of lifetime extension for each case in Table 7.1. This can be done by considering average properties (temperature, signal probabilities, activity ratios, etc.) of all transistors across the critical path of each design. In this way, the percentage of lifetime extension (LT) can be written for BTI as:

$$LT = \left(\left(\frac{r_1}{r_0} \right)^{1/n} \times \frac{e^{\left(\frac{-E_a}{kT_0^{avg}} \right)} \times \left(Y_0^{avg} \right)^n}{e^{\left(\frac{-E_a}{kT_1^{avg}} \right)} \times \left(Y_1^{avg} \right)^n} \right) - 1 \tag{7.2}$$

Similarly for HCI:

Table 7.1.: Results of investigating different mapping options for each design and the effect on both HCI/NBTI-induced aging

Circuit	Experiment	Area (LUT / FF)	Pre-aging critical delay	Post-aging critical delay (NBTI)	Post-aging critical delay (HCI)	Degradation ratio (NBTI)	Lifetime extension ratio	Degradation ratio (HCI)	Lifetime extension ratio
b04	Speed optimized	111 / 67	4.108 ns	4.622 ns	4.233 ns	12.51%	75.60%	3.04%	39.26%
	Forced placement	112 / 67	4.080 ns	4.617 ns	4.226 ns	13.16%	29.60%	3.58%	Reference
	Area optimized	106 / 59	5.341 ns	6.075 ns	5.481 ns	13.74%	Reference	2.62%	143.94%
b05	Speed optimized	133 / 41	2.777 ns	3.056 ns	2.794 ns	10.05%	120.33%	0.61%	54.42%
	Forced placement	133 / 41	2.766 ns	3.083 ns	2.787 ns	11.46%	Reference	0.76%	Reference
	Area optimized	96 / 34	3.169 ns	3.497 ns	3.185 ns	10.35%	84.30%	0.50%	159.06%
b11	Speed optimized	68 / 31	2.142 ns	2.442 ns	2.225 ns	14.01%	Reference	3.87%	98.56%
	Forced placement	68 / 31	2.263 ns	2.569 ns	2.390 ns	13.52%	23.47%	5.61%	Reference
	Area optimized	63 / 19	2.603 ns	2.966 ns	2.711 ns	13.95%	2.61%	4.15%	110.44%
b12	Speed optimized	256 / 123	2.348 ns	2.618 ns	2.365 ns	11.50%	60.15%	0.72%	21.47%
	Forced placement	256 / 123	2.370 ns	2.654 ns	2.389 ns	11.98%	25.05%	0.80%	Reference
	Area optimized	192 / 119	3.023 ns	3.399 ns	3.043 ns	12.43%	Reference	0.66%	87.29%
b14	Speed optimized	856 / 216	6.451 ns	7.256 ns	6.604 ns	12.48%	31.67%	2.37%	Reference
	Forced placement	856 / 216	7.173 ns	8.110 ns	7.285 ns	13.06%	Reference	1.56%	156.38%
	Area optimized	765 / 162	8.256 ns	9.305 ns	8.320 ns	12.70%	18.04%	0.78%	1096.71%
FIR Filter	Speed optimized	1086 / 515	3.513 ns	4.026 ns	3.574 ns	14.60%	2.73%	1.74%	15.24%
	Forced placement	1086 / 515	3.784 ns	4.339 ns	3.843 ns	14.67%	Reference	1.32%	53.85%
	Area optimized	962 / 499	4.311 ns	4.908 ns	4.400 ns	13.85%	41.18%	2.06%	Reference

$$LT = \left(\left(\frac{r_1}{r_0} \right)^2 \times \frac{e^{\left(\frac{-E_b}{kT_0^{avg}} \right)} \times AR_0^{avg}}{e^{\left(\frac{-E_b}{kT_1^{avg}} \right)} \times AR_1^{avg}} \right) - 1 \tag{7.3}$$

where T^{avg}, Y^{avg} and AR^{avg} are the average temperature, duty cycle and activation ratio respectively, for all transistors across the critical path of each design. The percentage of lifetime extension for each mapping compared to the worst mapping of every testbench circuit is reported in columns "Lifetime extension" in Table 7.1 for both NBTI and HCI.

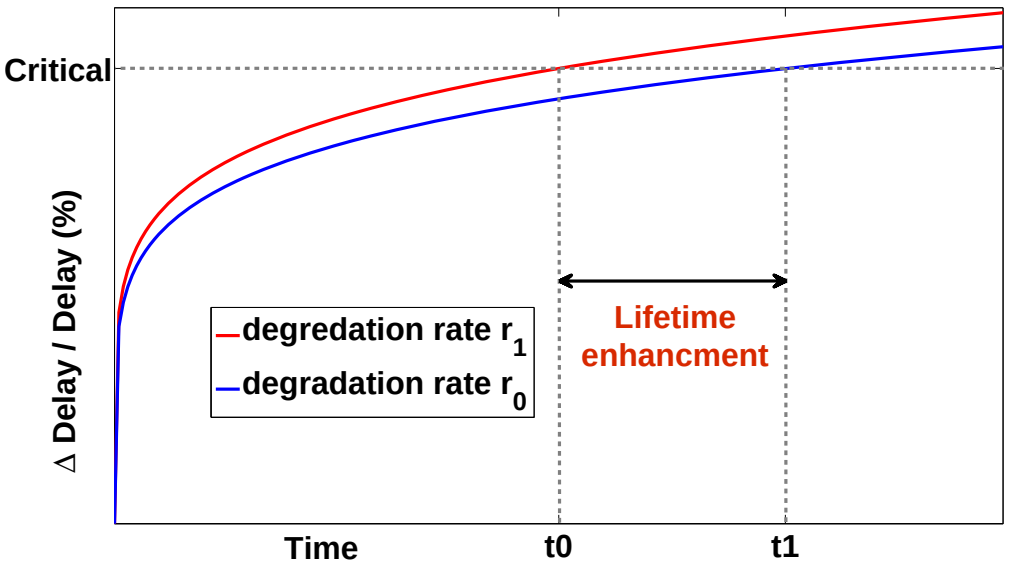


Figure 7.7.: A small change in the degradation ratio has a significant effect on the lifetime

Actually, several conclusions can be drawn from the results in Table 7.1:

- Aging of the FPGA device (both the NBTI-induced and the HCI-induced), and hence the lifetime, is significantly dependent on the design loaded onto it. This can be clearly seen from the degradation ratios column for both NBTI and HCI.
- Different mappings of the same circuit can results in relatively different aging rates.
- Changing the mapping and/or the optimization algorithm can significantly affect the lifetime of the design.
- It cannot be concluded whether the area-optimized mapping is always better in terms of aging compared to delay-optimized or vice versa (for both NBTI and HCI). It depends on and differs for each application.

7.5. Summary

In this chapter, an abstraction for both the BTI and HCI device level models to RTL models is proposed, which allows faster aging estimation for FPGA-mapped designs. These abstracted models, in addition to implicit device-level information existed in the power and timing reports provided from the FPGA's vendor tools, are used to present a tool for high-level aging estimation in FPGA. Using the proposed tool, several experiments have been made to explore the effect of different designs and mapping options. The results showed that aging of the FPGA device is dependent on the design loaded onto it. Furthermore, different mappings of the same circuit can result in different aging rates.

Part II.

Monitoring

Chapter 8.

Aging Monitoring in FPGA-Mapped Designs

8.1. Introduction

In order to increase the reliability of FPGA-mapped circuits, the aging effects must be monitored, thus suitable countermeasures can be made to avoid critical failures. In fact, the aging is a process dependent (i.e., due to process variation, the amount of degradation of each chip could be different). Additionally, the degradation depends not only on the time the chip is exploited in the field (e.g. 2 years) but also on the workload and environment (e.g. ambient and chip temperature). Therefore, to identify whether a particular FPGA chip is worn out, or to warn just before its useful lifetime is over, it needs to be continuously monitored in field. For this purpose, a logic-level circuitry for the detection of late-transitions that happen due to transistor aging in modern FPGAs is presented in this chapter. The advantage of the resources available in FPGAs is taken to design and implement low-cost and highly accurate online aging sensors. A scheme is provided to select which paths are to be monitored (the most aging-vulnerable paths) in the circuit using the high-level estimation tool presented in Chapter 7, thus a highly-efficient monitoring can be achieved.

By using the proposed sensor mapping techniques, the sensitivity of the sensor can selectively be adjusted to the range from a warning sensor to a late transition detector with a desired window. When used as a warning sensor, it can signal aging when the transitions happen in the timing guards, to be able to detect and mitigate aging of critical paths before it causes erroneous captures.

Unlike most of previous work which are based on ring oscillators and counters to measure variation and aging across the FPGA chip [80][81], the proposed sensor is *application dependent*, i.e. it monitors the correct functionality of critical paths in the FPGA-mapped design. To the best of our knowledge, the work in this chapter is the first approach for design and mapping of a logic-level aging sensor for FPGA-based designs.

The rest of the chapter is organized as follows. In section 8.2 the related work is reviewed. Section 8.3 presents the main idea of the proposed aging sensor. The FPGA mapping of the sensor is discussed in section 8.4. Section 8.5 contains the experimental results and analysis of the aging sensors. Finally, section 8.6 concludes this chapter.

8.2. Related Work

For delay fault testing in FPGAs, a BIST-based approach has been presented in [82], which stimulates several paths with a same length, then compares their output to detect faults. However, this method targets mainly manufacturing delay faults. In [52], a method is

developed for measuring and monitoring degradation in an FPGA, based on measuring the difference on transitions at inputs and outputs and their probabilities for a single path, but it is used offline to find at which frequency the circuit starts to fail. The chapter also estimates the effects of the aging-phenomena based on an NBTI model. The authors in [80] present a multi-use sensor implemented in reconfigurable logic in order to help estimating variation in delay, static and dynamic power, and temperature. The sensor is based on a ring oscillator and a residue-number-system ring counter, where an array of such sensors is arranged across the FPGA chip in order to measure the variation. Each sensor occupies 8 LUT of a Virtex-5 FPGA. The relation between the frequency of the ring oscillator and the desired parameter variation is used to estimate that variation. In [83] a delay measurement method based on *transition probabilities* (TPs) is presented. A set of test vectors are used to test the desired paths in the circuit, and to calculate the TPs at the output of each path, then based on the TPs, the delay of the paths are determined.

In the scope of reliability analysis for FPGAs, a dynamic thermal-aware reliability management framework has been presented in [81], which estimates the lifetime reliability of FPGAs, based on estimation of several phenomena and hard errors such as TDDB and EM, in addition to NBTI impacts. The proposed framework uses some tools and simulators to calculate the temperature variation across the chip, the switching activity of the design and the static probability of the signals, to do the estimation of the TDDB, The EM and the NBTI. Performance degradation of FPGAs due to HCI and TDDB has been analyzed in [84] and some load balancing and alternate routing techniques have been proposed to improve the reliability MTTF of the FPGA chip. The first proposed technique is to use controlled input vectors to optimize the active leakage power of logic blocks, and hence reduce the TDDB effect. The second technique is to balance the load on the circuit, and hence mitigating the HCI impact. Another technique of using a selective alternate routing is used to reduce the EM impact. In [53] accelerated-life tests are performed on FPGAs, to study the effects of the degradation, and three degradation-mitigation strategies are also discussed. The first strategy is relocating the logic functions to unused LUTs, the second strategy is to reroute signals to unused interconnects, and the third strategy is to exploit the unused regions of LUTs with spare inputs.

Logic-based sensors for ASIC designs to measure delay degradation in the circuit due to transistor aging are presented in [85] and [86]. In [85], the sensor is based on two ring oscillators, one as a reference and one under stressed conditions. The outputs of both oscillators are passed to a phase comparator to determine the difference. The difference between their frequencies is used to measure the delay. In [86], the sensor composed of a simple inverter and two tri-state buffers to measure the instability in the output of the critical path during a specific period. The sensor is to be placed at the output of the critical path, if the output is unstable within the desired period, due to aging, the output of the two tri-state buffers will be the same, and thus an error can be reported.

Techniques for eliminating the design margins in processor pipelines are presented in [87, 88, 89, 90, 91]. Although these techniques are intended to be used to detect violation in design margins caused by power saving techniques, or process variation, they can be used, in principle, for aging detection as well. However, the applicability for FPGA designs is not certain, because they require either non-logic type of resources or delay el-

ements, which have relatively large area impact when ported to FPGA. RAZOR [87, 90] for example, is a simple and good technique to detect delay faults in ASIC designs. However, mapping it to FPGA resources is difficult, as the placement of its shadow latch and the XOR comparator should be very accurate in order to catch the delayed transitions correctly. Given the constraints in type of available logic resources and routing path delay on current FPGAs; such precise timing cannot be met. Hence such sensors cannot be directly mapped to current FPGAs. In addition to these approaches, there are a large set of sensors and techniques for delay detection in ASIC designs, which use resources not available on FPGAs.

The main difference of the work in this chapter with the existing techniques is that, it exploits the native resources available on FPGA, to design a low-cost aging sensor with adjustable sensitivity. Additionally, a scheme is provided to select the places where the sensor should be placed to have a more reliable monitoring.

8.3. Aging Sensor: Main Idea

8.3.1. Critical path and aging

The maximum operational frequency of a circuit is determined according to the delay of the longest combinational path (*critical path*). A guard period is also added to this delay to define the minimum clock period ($T_{critical}$) to ensure correct functionality. This guard period is used to allow the signal to be stable before the FF setup time (Fig. 8.1), and also to consider process variation. Due to transistor aging, gate delays and in turn path delays are increased, causing transitions to arrive later and ultimately an incorrect value can be latched in the FFs (as illustrated in Fig. 8.1).

It should be noted that, in the presence of aging, the critical paths may change over time, i.e., some non-critical paths at the beginning of FPGA lifetime (t_0) may become critical after aging (t_n). More details are discussed in section 8.4.4, where a selection scheme to determine the highly-vulnerable paths to aging (aging-critical paths) is presented.

Two main concepts should be considered when designing a sensor to detect these late transitions due to aging or delay faults:

- *Warnability*: The sensor/detector should be able to generate a warning signal when the output of the critical path gets very close to the clock edge before exceeding it, which is the case of the aging phenomena that happens gradually. Thus a suitable action can be taken to mitigate aging (e.g., reducing the operating frequency).

- *Detectability*: When the output of the critical path exceeds the clock edge, a delay fault happens. The sensor/detector should be able to detect this fault and generate an error signal.

8.3.2. The proposed sensor

The proposed sensor is to be placed in parallel with the aging-critical path output (D_{out}), meaning the path that has most (initial delay + aging-induced delay increase), to detect

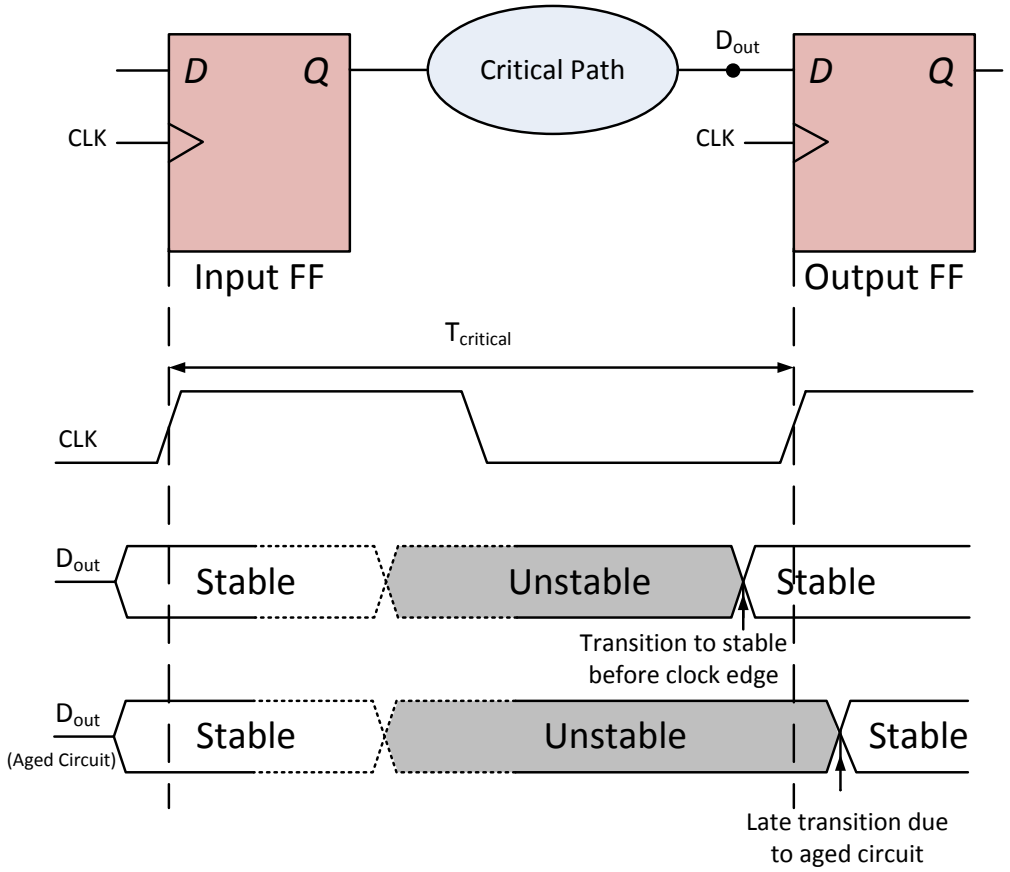


Figure 8.1.: Description of the effect of the aged circuit

whether the circuit generates late transitions after the clock edge, or too close before it, as shown in Figure 8.2. The schematic of the sensor is depicted in Figure 8.3. The idea is to use the signal on the critical path output itself as a clock input to two different edge-triggered D-FFs. Assuming that the original design works with the positive edge transition of the clock, the proposed circuit detects both positive and negative transitions on the critical path which happen during the first half (assuming a 50% duty cycle for the clock) of the next clock cycle (i.e. when the clock level is high). That is why the system clock is fed to the inputs of the D-FFs. The use of two FFs is because one latches rising transitions and the other one detects falling transitions. When the end point of the critical path makes a transition at the positive level (i.e. the first half) of the (next) clock, it means that the path is aged and makes a late transition.

One can argue that the non-aged (fault-free) path can also make the early transitions during the first half of the clock cycle. This may invalidate the functionality of the sensor as outlined above and cause it to wrongly raise the error signal. Modifications to the sensor to deal with such issues are discussed in section 8.4, where a narrower *detection window* is generated and used instead of the clock to reduce the detection period, and avoid the false detection of the early transitions that may happen (Figure 8.6).

To illustrate the sensor functionality, a chain of odd number of inverters is considered as a basic example for a critical path. The timing diagram in Figure 8.4 shows two cases,

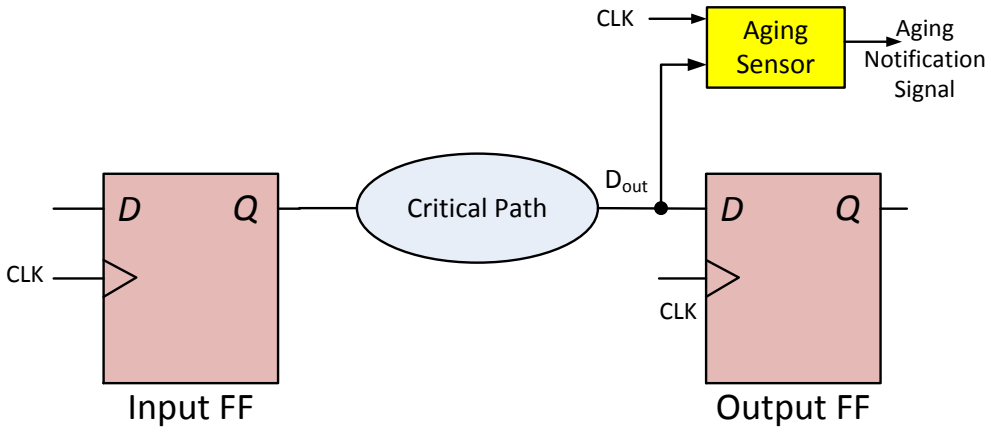


Figure 8.2.: The placement of the aging sensor on the critical path

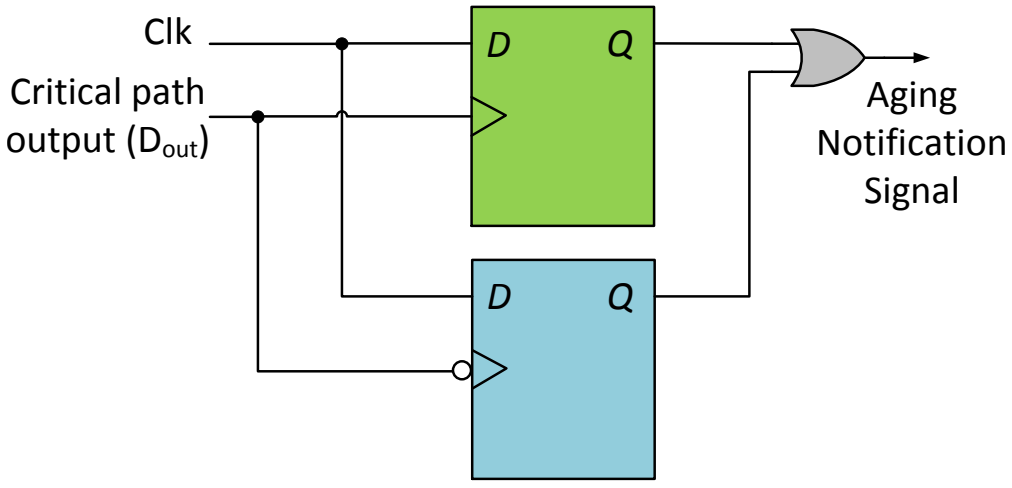


Figure 8.3.: Schematic diagram of the proposed aging sensor

namely the fault-free (non-aged) circuit (its output is D_{out}), and the aged circuit (with delay fault). In the non-aged case, the input to the inverters chain causes a transition at the output lies in the second half of the clock cycle (clk = 0), and will cause the sensor (Figure 8.3) to latch the current state of the clock, that is '0', which means no notification of aging. In the second case, the circuit is aged, and the transition in the critical path happens after the clock edge, in the first half of the next clock cycle (clk = 1). This transition causes the sensor to latch the current state of the clock, that is '1', and the sensor raises the error signal.

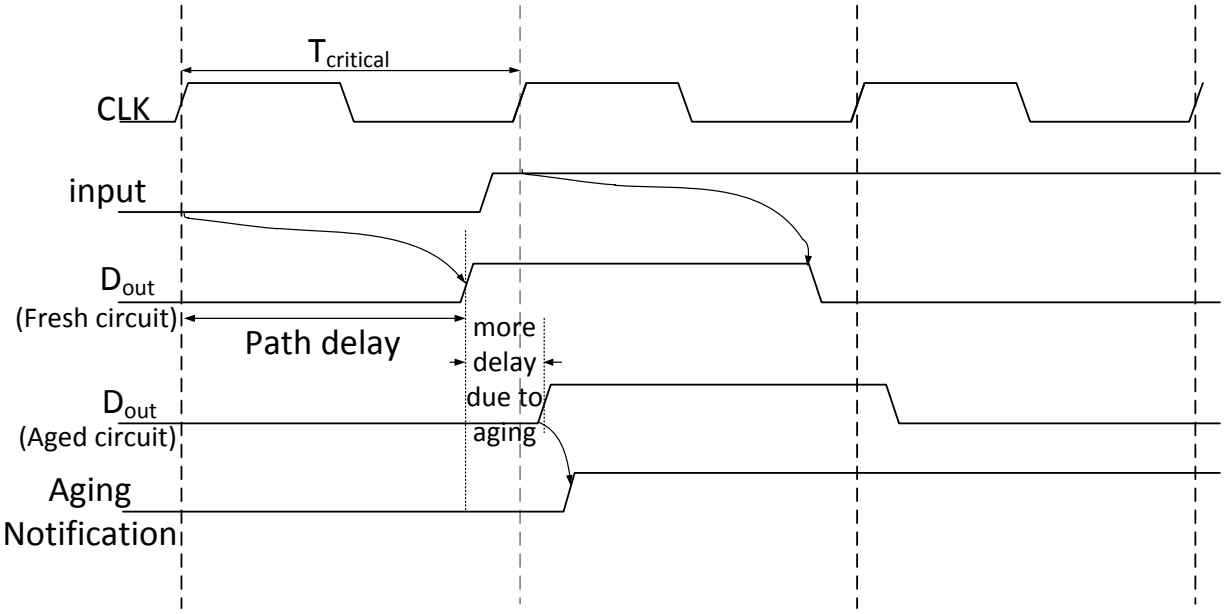


Figure 8.4.: Example to show the sensor functionality when the aging happens in the circuit

8.3.3. Sensor sensitivity analysis

The *sensitivity* of the sensor, T_{ss}, is defined as the time period from the latching edge of the clock to the earliest time that the path makes a transition after the clock edge and the sensor is able to raise an aging notification signal. In other words, if the clock period is T_{clk}, the delay of the critical path must be at least $T_{clk} + T_{ss}$ to be detected by the aging sensor. This means that any aging delay less than T_{ss} would not be detected by the sensor. In case of a warning sensor, T_{ss} is negative, thus the detection can happen before the clock edge (see Figure 8.5).

The amount of T_{ss} can be controlled according to the design requirements. To explain how this can be achieved, let's consider Figure 8.6 that shows the sensor's FFs and the routing delays for its inputs.

The FF requires the transition on its input to be stable before the clock edge, at least in an amount equal to its setup time (T_{set}), to latch the input successfully. Another fact that should be considered is that the timing, at which the sensor detects, must be relative to the

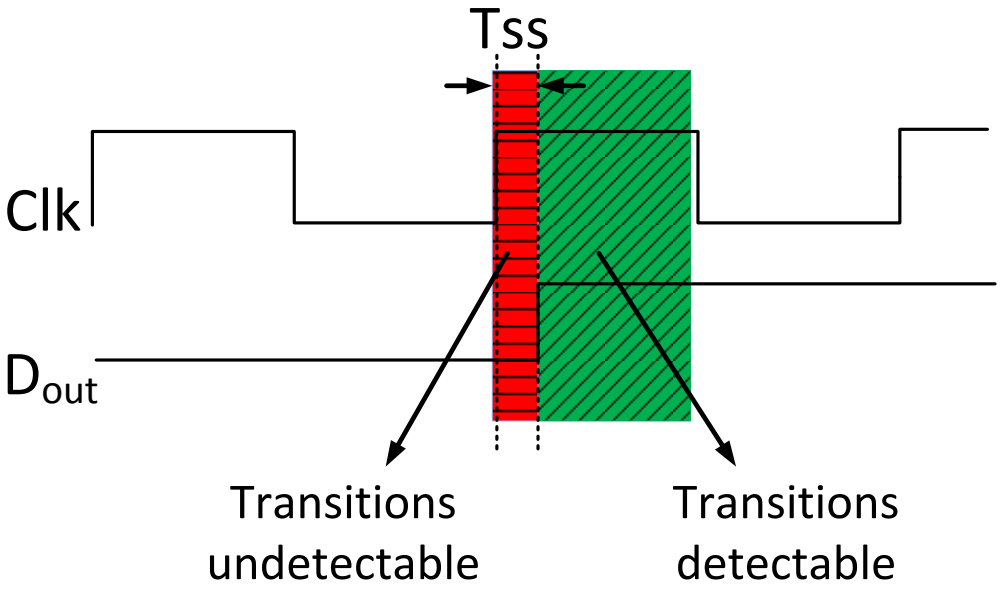


Figure 8.5.: The sensitivity of the sensor (T_{ss}), negative to the left of the clock edge, and positive to the right.

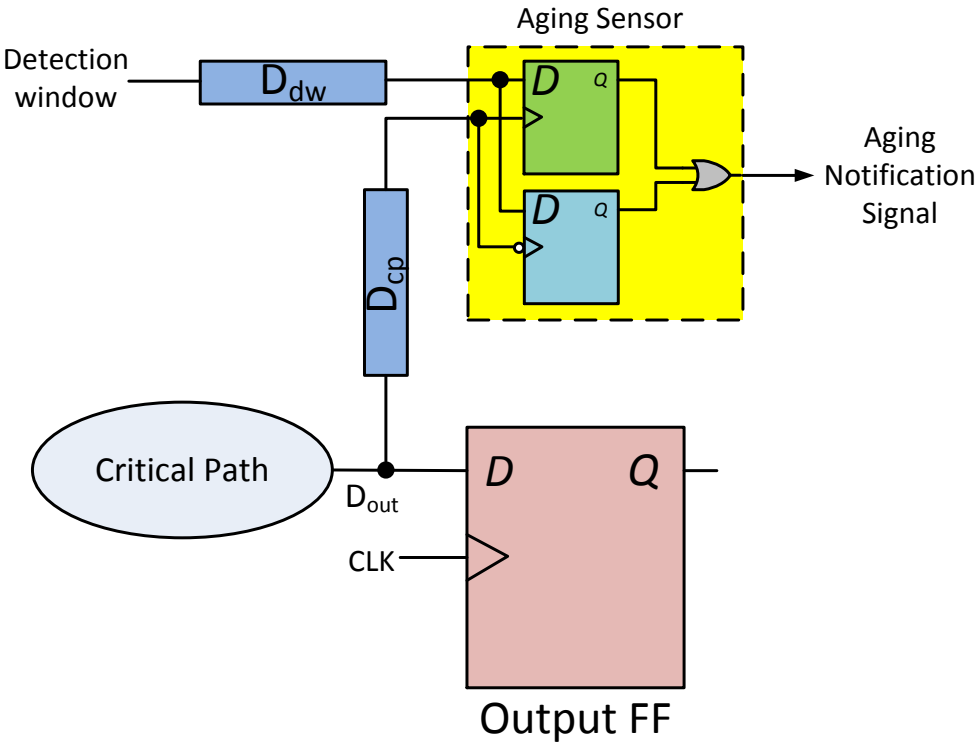


Figure 8.6.: The relative delays of the sensor inputs

timing of the output FF (at which the actual data is latched). Now, if we consider there is no skew between the inputs of the sensor's FFs, and the inputs of the output FF (Figure 8.6), then the sensitivity can be calculated as follows.

$$T_{ss} = T_{Sensor\_set} - T_{Output\_set} \tag{8.1}$$

where T_{Sensor_set} and T_{Output_set} are the setup times of the sensor's FFs and the output FF, respectively. If no process variation is considered, the FFs will have equal setup times, and as a result T_{ss} will be simply zero.

In the real case there are routing delays that lead to variable amounts of skew. If the two delays are different, the signals will reach the sensor at different timing with respect to the output FF (Figure 8.7). Considering these delays, the sensitivity equation (Equation (8.1)) will become:

$$T_{ss} = (T_{Sensor\_set} - T_{Output\_set}) + (D_{dw} - D_{cp}) \tag{8.2}$$

where D_{dw} and D_{cp} are the path delays of the sensor inputs (from the detection window and the critical path) relative to the inputs of the output FF.

Since the setup times cannot be altered, the term $(D_{dw} - D_{cp})$ is used to control the amount of the required sensitivity. For an ideal late transition detector the sensitivity is zero, which means that the late transitions can be detected as soon as they exceed the clock edge. This can be achieved when both terms $(T_{Sensor\_set} - T_{Output\_set})$ and $(D_{dw} - D_{cp})$ are equal to zero, or when $(T_{Sensor\_set} - T_{Output\_set}) = -(D_{dw} - D_{cp})$.

To obtain a negative sensitivity (warning sensor), the term $(D_{dw} - D_{cp})$ must be negative. That means the amount of the delay from the output of the critical path is larger than that of the clock (Figure 8.7).

8.4. Sensor Mapping

Xilinx Virtex-6 is used, as one of the state-of-the-art FPGA platforms, for the mapping. As presented in section 8.3, the sensor uses two different edge triggered D-FFs to latch the actual state of the clock. It would be much useful if double edge triggered D-FFs are available on the FPGA, because then using one FF would be enough to detect both negative and positive transitions and there will be no need for the OR gate shown in Figure 8.3 anymore. Unfortunately, double-edge triggered FFs are not available on the Virtex-6 series; however, there are similar components that can be used instead, namely *Double-Data Rate Output Registers* (ODDR). The ODDR exists near the general purpose input/output pins, and one ODDR is enough to implement the functionality of the proposed sensor. However, because the aging notification signal has to be sent to one of the output pins outside the FPGA, this may cause problems since it might be necessary to handle and use this signal inside the FPGA. Furthermore, the ODDRs exist only in certain places on the FPGA, so the output of the critical path has to be routed there, which could be very long (through multiple switch matrices and buffers) depending on the original placement of the critical path.

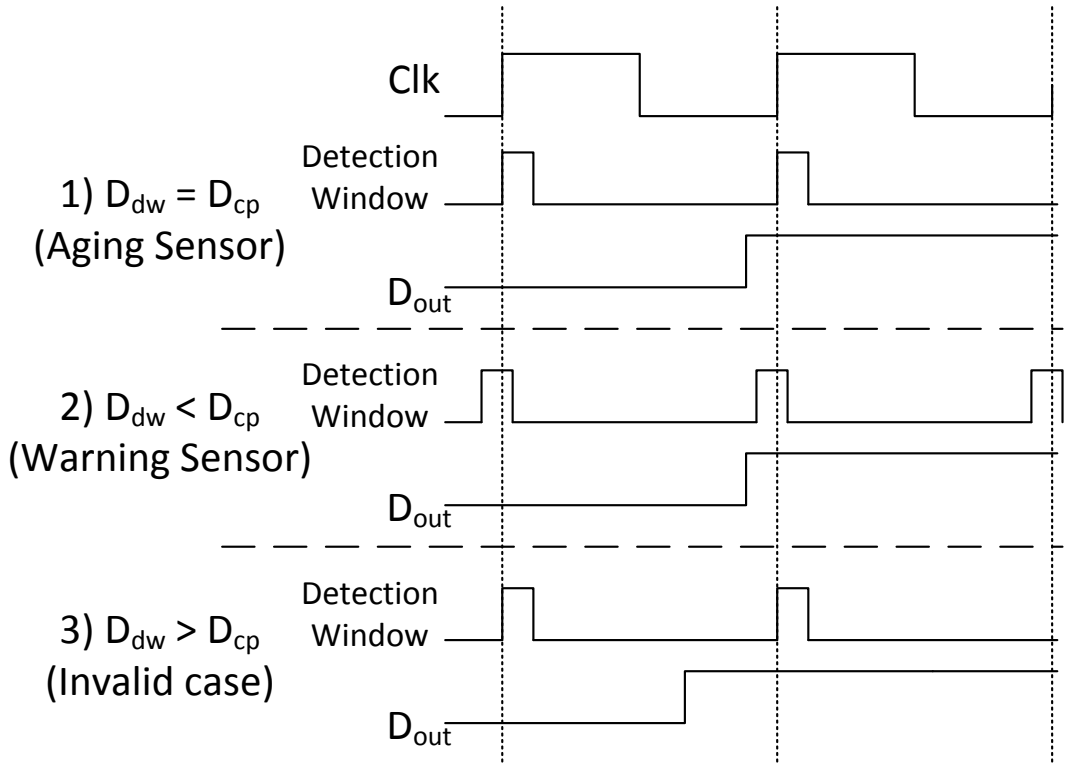


Figure 8.7.: Different relative values of input delays and the effects of how the sensor sees the inputs

8.4.1. Mapping to logic slices

The other option to map the proposed aging sensor is the normal logic slices that are distributed over the FPGA area. The fact that in each slice of the Virtex-6 FPGA only one type of clock edge can be defined, implies the need to use two different slices to implement the aging sensor: one for the positive edge D-FF and the other one for the negative edge. Although the sensor occupies two slices, it uses the D-FFs and one LUT, leaving the rest of resources in these slices available for mapping other circuit components.

The basic aging sensor (Figure 8.3) is able to detect any transition happens during the positive level (first half) of the clock cycle. This can generate false notification if an early transition happens in this period in the fault-free operation of the circuit. To alleviate this problem, it is necessary to reduce the window in which the sensor latches the transition. In the basic sensor implementation, this window was generated by the positive level of the clock. Since the clock typically has a 50% duty cycle, this latching window is 50% of the clock period. This is an issue for many functional paths as they make early transitions during the clock cycle. However, if this latching window is reduced such that the monitored path in the non-aged state does not make any transition during this latching window, no false notification can happen.

The *Mixed-Mode Clock Managers* (MMCM) component in Virtex-6 allows the generation of a controlled-duty-cycle clock. This option fits the need for a flexible aging sensor. By generating a smaller duty cycle period, the latching window can be reduced proportion-

ally. In this case, it is enough to use the controlled duty cycle clock in place of the functional clock in the sensor. However, the MMCM is unable of generating a small duty cycle signal when the functional clock frequency is relatively high (300-600 MHz). This makes this method unsuitable for high-frequency designs. A suitable method for high-frequency designs is introduced in the next section.

8.4.2. Detection window generation

Using the MMCM, a phase shift version of the clock can be generated, which can be then combined with the original clock using an AND gate to generate the required detection window (Figure 8.8). The amount of the phase shift can specify the width of the detection window. It should be noted that the width of the detection window cannot be less than the allowed minimum signal width of the FPGA as described in section 8.4.3, otherwise, the detection window will be absorbed by the internal buffers and will not reach the FFs.

This combination (the original clock with the phase shifted one) must be done before passing the two generated clocks (The functional clock and the generated detection window) to global clock buffers, in order to assure minimum skew. Furthermore, the two clock paths must be balanced using the same amount of components, to avoid large delay differences between them. That is the reason why there is a NOT gate on the functional clock path in Figure 8.8, so that the two clocks have nearly the same propagation delay of one LUT each.

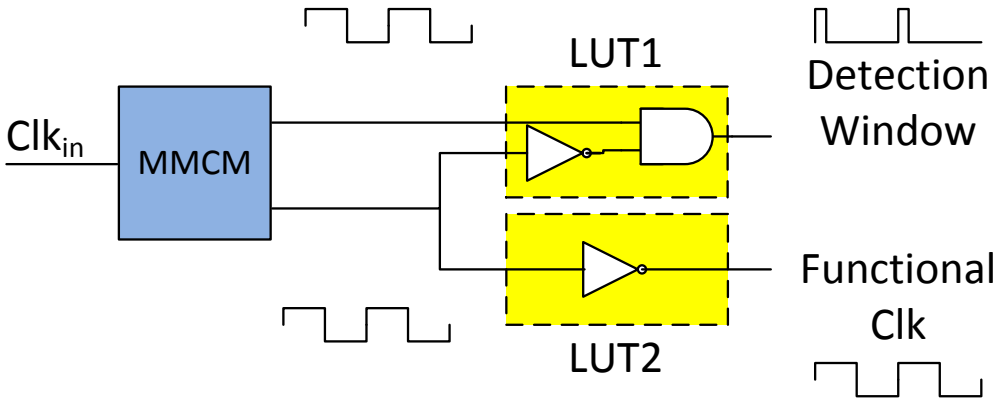


Figure 8.8.: The generation of the detection window using the MMCM

8.4.3. Glitches in FPGA

As the proposed sensor uses the data path as a clock source, it may face too many transitions, and hence the power consumption of the sensor maybe high. Before going further with the analysis of this information, an interesting point must be highlighted. In modern FPGAs, special types of buffers are used for the clocks and the inputs of the FFs inside

the slices. These buffers allow only pulses with a width greater than a certain amount to be further propagated. Thus, glitches that are very small are internally absorbed by the buffer [92]. The timing simulations that have been done in the scope of this chapter prove that fact as well. Table 8.1 shows different simulated FPGAs, with the minimum pulse-width that can be propagated through the buffers.

Table 8.1.: Minimum allowed pulse widths in different FPGAs

Simulated device	Minimum allowed pulse width
Spartan-6 LX45	240 ps
Virtex-5 LX110t	369 ps
Virtex-6 LX75t	450 ps

The values in Table 8.1 are actually defined in the generated post-place-and-route simulation models, under "PATHPULSE => xxx ps" constraint. The possible generated small glitches cannot reach the output FF (The monitored FF, where these glitches are supposed to be latched and cause errors) as they will be absorbed by the buffers, also these glitches cannot reach the sensor's FF. Actually, this is the idea, because the sensor is supposed to generate errors only if the latched data is incorrect, so as long as the output FF does not see these glitches, the sensor won't see them as well. Furthermore, as the possible number of glitches that can reach the sensor clock input within one clock cycle is minimum, its power consumption would be minimum as the results show in section 8.5.

8.4.4. Aging sensor placement and calibration

Choosing the appropriate place (CLB slices) for mapping the sensor with respect to the placement of the *aging-critical paths* (the paths that have the highest post-aging delay) to be monitored, and the number of the sensors for the entire circuit are important issues to be addressed.

8.4.4.1. Selection scheme of the paths to be monitored

The correct choice of where to place the sensors in the circuit, plays an important role for achieving a highly reliable monitoring. The method presented in Chapter 7 is used here to select the most top aging-critical paths to be monitored. This can be summarized in the following steps:

1. Using the timing analyzer tool, sort the paths based on their timing slacks in the decreasing order. This way, the critical and near-critical paths are determined.
2. Estimate the aging of each path using the method proposed in Chapter 7.
3. Resort the paths according to their aged delays.

4. As the sensor is to be placed at the output of the path, if several paths share the same output node, keep the one with the highest post-aging delay and remove the others.

5. Depending on the criticality of the application, and the available space left on the FPGA chip; the number of the paths to be monitored can be determined.

6. Place the sensor circuits at the outputs of selected paths.

7. A calibration to the sensors should then be done to set their sensitivity (negative for warning, and near-zero positive for late transition detector) as detailed below.

8.4.4.2. Sensor calibration

As mentioned in section 8.3.3, the sensitivity can be calibrated using the delays at the inputs of the sensor. An option to use programmable delay elements would consume too many resources. Therefore, the proposed approach is to control the routing to specify the amount of the delay in a simple way.

The first knob to calibrate sensitivity is the delay on the detection window path, which can be controlled using a relative location constraint. The locations of the AND and the NOT gates (The LUT to which they are mapped in Figure 8.8) can be chosen such that one of them is farther than the other one, with respect to the clock buffers. Thus specifying one of them affects the relative delay of the other one, accordingly. For example, suppose that the names of the two LUTs in Figure 8.8 are `MMCM/Detection_window` and `MMCM/Functional_clock`, the relative location constraints for them can then be written as

```
INST "MMCM/Detection_window" U_set="clock_0";
INST "MMCM/Functional_clock" U_set="clock_0";

INST "MMCM/Detection_window" RLOC = X0Y0;
INST "MMCM/Functional_clock" RLOC = X10Y0;
```

which means that the LUT from which the functional clock is passing, is 10 slices farther than the other, with respect to the clock buffers of both, and hence the detection window advances the functional clock. Modifying this distance changes the amount of delay between the two clocks.

The second knob is the relative delay from the critical path output to the sensor. Again, the relative location constraints can be used to determine the place of the sensor relative to the place of the output FF, and thus increasing or decreasing the delay difference between the output FF inputs and the sensor inputs. For example, suppose that the name of the output register in Figure 8.6 is `Output_reg`, and the names of the sensor's FFs are `sensor0/pos_edge` and `sensor0/neg_edge`, the relative location constraints can then be written as

```
INST "Output_reg" U_set="sensor0";
INST "sensor0/pos_edge" U_set="sensor0";
```

```
INST "sensor0/neg_edge" U_set="sensor0";

INST "Output_reg" RLOC=X0Y0;
INST "sensor0/pos_edge" RLOC=X0Y1;
INST "sensor0/neg_edge" RLOC=X1Y1;
```

By controlling these two knobs, the sensor sensitivity can easily be calibrated using Equation 8.2.

8.4.4.3. Pre-used FPGAs

In reconfigurable applications, some other configurations may have been loaded in the FPGA and because of that, the FPGA device is aged based on that usage. When the new configuration is loaded in the FPGA, the effect of pre-aging due to previous configurations has to be considered (in which some FPGA resources are aged with different rates). To handle such cases in the proposed approach, instead of considering d_0, as the path delay of "fresh FPGA", one can use d_0^u as the delay of that path in the "used FPGA" (see Equation 8.3).

$$d_0^u = d_0 + \Delta d_0 \tag{8.3}$$

where Δd_0 is the aging of the path so far. In other words, d_0 is updated with the delay increase so far, and the delta delay in future, and hence, the post-aging path delay d becomes $d = d_0^u + \Delta d$.

8.5. Experimental Results

In order to evaluate, validate and analyze the proposed aging sensor, experimental analysis for representative high frequency FPGA designs have been performed.

8.5.1. FPGA design tool experiments (simulation results)

The simulations are done using Xilinx ISE 12.2, together with Modelsim SE 6.5c for Xilinx Virtex-6 FPGA devices. Post-place and route simulation model from ISE is generated for the simulation of each experiment and used together with the generated Standard Delay Format (SDF) file in Modelsim. The maximum delay values for all components were always considered in the following results, to reflect the worst-case results. Virtex-6 series FPGAs are chosen for the simulated device since they are one of state-of-the-art FPGA devices from Xilinx fabricated using a 40nm copper CMOS process technology.

To have a fair analysis of the sensor, three circuits with typical operational frequency have been firstly chosen to reflects real usage of the FPGA. The first circuit is a pipelined 32-bit square root circuit [93]. It contains 5 pipelined stages, and operates at a frequency of about 320 MHz. The second circuit is a pipelined AES encoder [94], which contains 30 stages, and operates at a frequency of about 550 MHz. The third circuit is a self-built typical 8-bit FIR filter with 32 stages, has LUT-based implementation, and operates at a

frequency of about 275 MHz. In addition, the sensor has been tested on a set of several circuits from the ITC'99 testbench with different sizes and frequencies.

To assign the sensors to the critical paths in the circuits, a logic-level implementation is necessary, because the behavioral description contains internally generated paths which may not easily be extracted to be monitored. The logic-level descriptions for the circuits were generated using Xilinx ISE synthesis tool. Different numbers of sensors are placed to the top aging-critical paths of the circuit as described in section 8.4.4. The sensors are then calibrated as warning sensors. The suitable device size to efficiently fit the requirements of most of the tested circuits is chosen to be XC6VLX75T-FF484. For the AES encoder circuit, XC6VLX240T-FF784 is chosen. To simulate the aging phenomena, the frequency of the circuit is increased gradually, and the outputs of critical paths are reported together with the sensors behavior. The sensors were calibrated to work with a sensitivity of almost -50 ps. When the critical path transitions fall within 50 ps prior to the clock edge, the sensors generate the aging_notification signal. By using the detection window adjustment technique presented in Section 8.4.2, no false notifications happened for the inserted sensors.

The area overhead of the sensors for the tested circuits is reported in Table 8.2. The area is chosen as an optimization goal in the synthesis phase to have a fair comparison. For power and performance overhead reported in Table 8.3 the speed is chosen as an optimization goal. The area overhead of the sensor is very small, as each sensor needs only 2 FFs and 1 LUT. The performance overhead is also very small. The power overhead is mainly caused by using the MMCM to generate the detection window. This can be seen from the results in table 8.3 where adding extra sensors does not scale the power linearly. Actually the power can be further reduced by selectively activating the sensors from time to time. The MMCM element is used in all circuits, the original circuits (i.e. without sensors) and with-sensors circuits, to generate the 200+ MHz clock frequencies; therefore, the MMCM is not considered in the area comparison. The negative values of area overhead in table 8.2 are related to LUTs which have been optimized; however the area overhead of the registers is always positive.

It needs to be noted that the transistor aging happens at a very large time scale. Therefore, the critical path is not required to be monitored all the time: a periodic (e.g. once every week or month) monitoring of the critical path is enough. Given the runtime reconfigurability of FPGAs, it is possible to turn off the sensor most of the time and only activate it at very infrequent rates. Alternatively, a control signal can be easily asserted to the sensor to enable/disable it. The Clock Enable (CE) ports of the D-FF in the sensor can be used for that purpose. Since the sensor circuitry would be used (switched) much less frequently than the functional path, the aging rate of the sensor circuitry would be multiple times less than the original circuit. Therefore, the aging of the sensor circuitry can be neglected compared to the aging of the original circuit. This periodic activation of the sensor can also reduce the power associate with the sensor circuitry considerably.

8.5.2. FPGA board experiment (emulation results)

To validate the sensor functionality in real-time environment, an XUP-5 board equipped with a Virtex5-LX110t is used. The square root circuit with five attached sensors is tested.

Table 8.2.: Area overhead for different number of sensors

Tested circuit	Resource type	Original	With 5 sensors	With 10 sensors	With 20 sensors
b04	Slice registers	62	77 (24.19%)	90 (45.16%)	110 (77.41%)
	Slice LUTs	108	111 (2.77%)	113 (4.62%)	116 (7.41%)
b05	Slice registers	44	54 (22.72%)	64 (45.45%)	84 (90.90%)
	Slice LUTs	132	135 (2.27%)	138 (4.55%)	141 (6.82%)
b12	Slice registers	126	136 (7.94%)	146 (15.87%)	166 (31.75%)
	Slice LUTs	240	243 (1.25%)	245 (2.08%)	248 (3.33%)
b14	Slice registers	165	175 (6.06%)	185 (12.12%)	205 (24.24%)
	Slice LUTs	782	785 (0.38%)	787 (0.64%)	790 (1.02%)
b17	Slice registers	1334	1344 (0.75%)	1354 (1.50%)	1374 (3.00%)
	Slice LUTs	5643	5650 (0.12%)	5652 (0.16%)	5651 (0.14%)
Square root	Slice registers	924	934 (1.08%)	944 (2.16%)	964 (4.33%)
	Slice LUTs	997	1009 (1.20%)	1007 (1.00%)	1014 (1.70%)
AES encoder	Slice registers	7748	7758 (0.13%)	7768 (0.26%)	7788 (0.52%)
	Slice LUTs	9698	9754 (0.58%)	9697 (-0.01%)	9657 (-0.42%)
FIR filter	Slice registers	499	509 (2.00%)	519 (4.01%)	540 (8.22%)
	Slice LUTs	962	966 (0.42%)	968 (0.62%)	971 (0.94%)

Table 8.3.: Power and performance overhead for different number of sensors

Tested circuit	Power / Performance	Original Design	With 5 sensors	With 10 sensors	With 20 sensors
b04	Power at 229 MHz	0.795 W	0.845 W (6.29%)	0.871 W (9.56%)	0.876 W (10.19%)
	Performance	4.332 ns (230 MHz)	4.347 ns (230 MHz) (0.35%)	4.350 ns (230 MHz) (0.42%)	4.365 ns (229 MHz) (0.76%)
b05	Power at 338 MHz	0.807 W	0.828 W (2.60%)	0.837 W (3.72%)	0.850 W (5.33%)
	Performance	2.877 ns (347 MHz)	2.942 ns (340 MHz) (2.26%)	2.912 ns (343 MHz) (1.22%)	2.956 ns (338 MHz) (2.75%)
b12	Power at 400 MHz	0.812 W	0.838 W (3.20%)	0.846 W (4.19%)	0.868 W (6.70%)
	Performance	2.458 ns (406 MHz)	2.491 ns (401 MHz) (1.34%)	2.484 ns (402 MHz) (1.05%)	2.499 ns (400 MHz) (1.67%)
b14	Power at 146 MHz	0.800 W	0.807 W (0.88%)	0.807 W (0.88%)	0.811 W (1.38%)
	Performance	6.588 ns (151 MHz)	6.835 ns (146 MHz) (3.75%)	6.810 ns (146 MHz) (3.37%)	6.825 ns (146 MHz) (3.60%)
b17	Power at 180 MHz	0.819 W	0.822 W (0.37%)	0.829 W (1.22%)	0.829 W (1.22%)
	Performance	5.513 ns (181 MHz)	5.527 ns (180 MHz) (0.25%)	5.501 ns (181 MHz) (-0.21%)	5.511 ns (181 MHz) (-0.04%)
Square root	Power at 310 MHz	0.951 W	0.978 W (2.84%)	0.976 W (2.62%)	0.985 W (3.58%)
	Performance	3.130 ns (319 MHz)	3.144 ns (318 MHz) (0.44%)	3.173 ns (315 MHz) (1.37%)	3.184 ns (314 MHz) (1.73%)
AES encoder	Power at 500 MHz	3.901 W	3.900 W (-0.03%)	3.928 W (0.69%)	3.977 W (1.95%)
	Performance	1.856 ns (538 MHz)	1.857 ns (538 MHz) (0.05%)	1.896 ns (527 MHz) (2.16%)	1.865 ns (536 MHz) (0.48%)
FIR filter	Power at 270 MHz	1.566 W	1.581 W (0.96%)	1.584 W (1.15%)	1.597 W (1.98%)
	Performance	3.619 ns (276 MHz)	3.620 ns (276 MHz) (0.03%)	3.662 ns (273 MHz) (1.19%)	3.633 ns (275 MHz) (0.39%)

The maximum frequency for the mapped circuit is reported as 340 MHz. For the generation of the detection window, two DCMs are used, because the MMCM element is not available in Virtex5. The outputs of the sensors are passed to an OR gate and connected to a LED.

Two input clock frequencies were tested: 1) under the maximum frequency (325 MHz), to test the sensor in normal circuit operation mode, and 2) above the maximum frequency (400 MHz), in order to emulate the aged circuit case. The LED was OFF during the first case, which means that no late transitions were detected, and ON during the second case, which proves that a late transition has been detected.

8.6. Summary

In this chapter, the design and mapping of a low-cost logic-level aging sensor for FPGA-mapped designs have been presented. It has been taken advantage of FPGA resources to design an efficient aging sensor (controlled to be warning or late-transitions detector) which not only detects transistor aging, but can detect erroneous glitches due to intermittent and transient faults. The implementation of such sensors for representative designs shows very low area, performance and power overhead. ($\approx$ 1.3% area, $\approx$ 1.6% performance, and $\approx$ 1.5% power overhead when 10 sensors are placed)

Chapter 9.

Self-Heating Thermal-Aware Testing of FPGAs

9.1. Introduction

Testing of FPGA chips is of great importance, not only at the manufacturing phase, but also during the in-field operation to ensure the correct functionality throughout the system lifetime. As many failure mechanisms are accelerated with high chip temperatures, thermal-aware methods of testing are usually used to detect such failures, and also to ensure the quality of the devices for a required thermal specification [95]. In this type of testing, the chip is first heated to a certain temperature and then the test is applied. External devices such as thermal chambers and ovens are used for the purpose of heating up the chip. Burn-in testing is another type of testing in which the chip is exposed to high temperatures and/or voltages for certain periods in order to accelerate the aging and stress the chip before applying the test. Again, in this type of testing external devices are used for heating up the chip. However, the use of these devices for thermal-aware testing has various limitations. For instance, they cannot provide on-chip thermal gradient for thermal testing; they are large and expensive, and have limited usability for in-field test; there is a possibility of heating up unwanted components on the board; and the number of chips that can be simultaneously tested is limited.

In this chapter, an approach for self-heating the FPGA chips is presented together with several techniques for integrating this self-heating with the test scheme for the purpose of thermal-aware testing. Thus, no external devices are needed. In the proposed approach, the internal logic resources of FPGA are used to build controlled *Self-Heating Elements* (SHEs). These SHEs are very flexible in both size and quantity, they can be either distributed across the FPGA to heat up all the FPGA at once, or they can be concentrated in a certain part of the FPGA to generate the required thermal gradient across the chip. Furthermore, a simple programmable controlling scheme has been developed to tune the amount of generated heat according to the test requirements.

Two different categories of SHEs integration techniques are presented to fit different testing purposes; the first category is for *Built-In Self-Test* (BIST) in application-independent testing. Here the SHEs are integrated with the BIST scheme for FPGAs, which allows any specific thermal profile during test by specific placement of SHEs throughout the chip as well as programming SHEs for particular heat generation. In that way, the required testing temperature is maintained throughout the test. The second category of the SHEs integration techniques is for application-dependent FPGA testing, in which the SHEs are integrated within the specific user application with minimum overhead to provide the self-heating for the thermal-aware testing.

The proposed approach has many advantages. First, it eliminates the need for any ex-

ternal devices for heating, and thus reducing the testing cost and assuring heating only the FPGA chip and not any other components on board. Unlike the external heating devices, it can generate uniform temperature or particular thermal gradient on the chip for specific test plans. Furthermore, it can be used for in-field thermal-aware testing by end users. Additionally, there is no limitation on the number of FPGA chips to be heated in parallel and hence reducing the test time. Moreover, this technique can primarily be used for thermal-aware testing as a part of manufacturing and production test flow. Also, it can be used as a part of acceptance test by the user in the field. The presented approach is demonstrated on a Virtex-5 FPGA and integrated with both a particular FPGA BIST scheme and an application-dependent test. The experimental results show that a wide range of maximum chip temperatures can be achieved (from 50°C up to 125°C) with a high accuracy ($\pm$ 1°C).

The rest of this chapter is organized as follows: Section 9.2 discusses the related work. Then, Section 9.3 illustrates the concept of self-heating and introduces the proposed SHEs. Afterwards, Section 9.4 discusses the generation of the desired thermal profile for testing. Later, Section 9.5 describes the integration of the SHEs with BIST structures, which is followed by a demonstration for the implementation of the proposed self-heating BIST on Xilinx Virtex-5 FPGAs. Then, Section 9.6 describes the thermal-aware application-dependent testing, where several SHEs integration methods are discussed. Section 9.7, shows the experimental results, and finally Section 9.8 concludes this chapter.

9.2. Related Work

The thermal-aware testing of chips is studied in literature with the focus on test procedure and schedule. For example, in [95, 96, 97] the thermal-aware test scheduling is handled. However, there is no focus on chip heating methods, and external heating devices are mostly considered.

The concept of self-heating in FPGA is discussed in the literature mostly as an undesired property that should be tolerated, e.g., in ring oscillators when used as thermal sensors [98, 99]. Some other works refer to self-heating in FPGA implicitly when the thermal issues of FPGA such as hot-spots are discussed [74, 100, 101]. Explicit methods for heat generation in FPGAs are presented in [102, 103].

The concept of self-heating for test purposes in *Programmable Logic Devices* (PLDs) is discussed in [104], which tests a performance-failing PLD by isolating thermal effects from AC effects on circuit performance. This is done by isolating the critical paths and placing a type of thermal generators around them to heat them for testing. The heat generated by each of the thermal generators is a result of creating an intentional short circuit between the output of two different logic blocks, one with logic-0 output and the other with logic-1 output. The same technique for generating the heat is also adopted in [105]. Obviously, this technique cannot be used in the production test as it may damage the chip permanently if the created short circuits are not controlled correctly. In [106], another self-heating method based on toggling flip-flops is introduced. The temperature is controlled by changing the frequency of the clock signal that feed the toggling flip-flops. To generate high chip temperatures (above 100°C), a high frequency clock signal is needed, which

might not be possible in FPGAs.

The main difference between the work presented in this chapter and the aforementioned references is that the focus here is not the self-heating itself but rather the integration of the self-heating with different testing schemes for efficient thermal-aware testing.

9.3. FPGA Self-Heating

9.3.1. The concept of self-heating

The idea of self-heating is to make the FPGA able to generate heat by increasing the power consumption of its elements. This can be achieved in several ways e.g., creating intentional short circuits, forcing high toggling rates at the inputs/outputs of the FPGA resources, etc. The approach that includes forcing of high toggling rates is safer and straightforward to implement and control, and thus it is preferred. Forcing high toggling rates can be applied to most FPGA resource types such as Block-RAMs or DSP elements. However, most of these resources are concentrated in a few locations on the FPGA die and may not be available at all in different FPGA chips. Therefore, if an even temperature distribution across the FPGA is desired, these types of resources are unsuitable to implement self-heating. On the other hand, logic resources are well distributed across the FPGA, which makes them a good candidate for being used for the implementation of self-heating modules. Usually the logic resources consist of a set of LUTs and FFs as well as switching/routing resources and multiplexers.

9.3.2. Self-Heating Elements (SHEs)

In this chapter, the main components to implement the proposed SHEs are LUTs and FFs. As these resources are available in a large number and evenly distributed across the FPGA, the SHEs can be freely placed in any desired location on the FPGA and thereby their density and quantity can be controlled, which determines the amount of heat generated in order to achieve the desired final on-die temperature distribution.

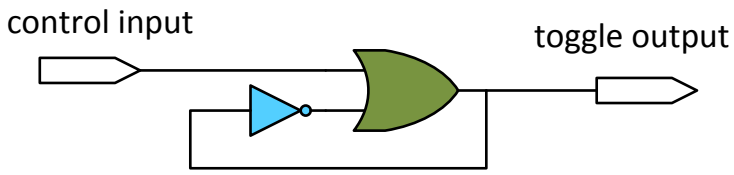


Figure 9.1.: Basic toggle logic

The basic circuit for controlling input/output toggling is depicted in Figure 9.1. It consists of a basic ring oscillator with an additional signal to control the toggling rate. This circuit can be realized using an LUT with at least two inputs. Usually, the feedback signal

from the LUT output to its input goes through switching resources, which cause these resources to toggle as well and hence their dynamic power consumption will increase in turn, enabling more heat to be generated. For larger LUTs (e.g. with 6-inputs), it is preferable to toggle all the inputs in order to maximize the power consumption. This can be achieved by connecting the feedback signal to all inputs.

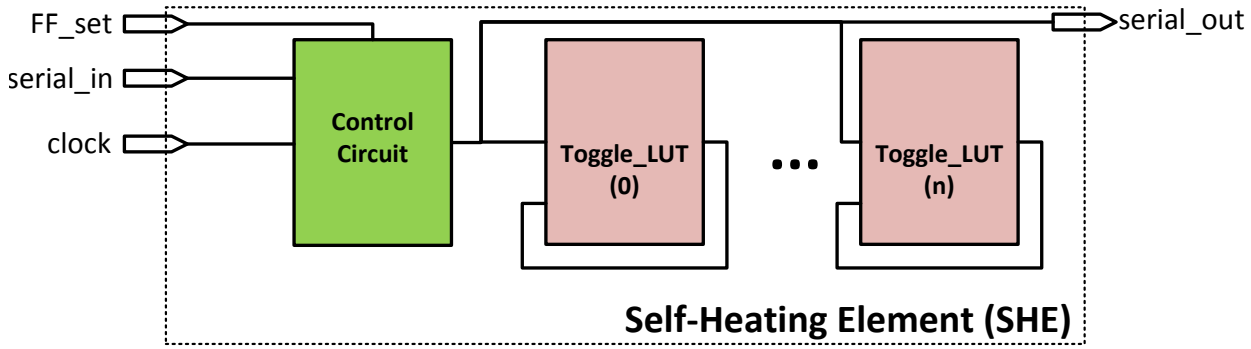


Figure 9.2.: Components inside an SHE

To maximize the power consumption, each of the proposed SHEs consists of several toggling LUTs. Figure 9.2 shows an overview of a single SHE. It consists of a set of toggling LUTs in addition to a control circuit. The logic function implemented by each toggling LUT is depicted in Figure 9.3. Basically, this is an extension to the circuit in Figure 9.1 to achieve toggling on the rest of LUT inputs by connecting them to feedback signals from other LUTs.

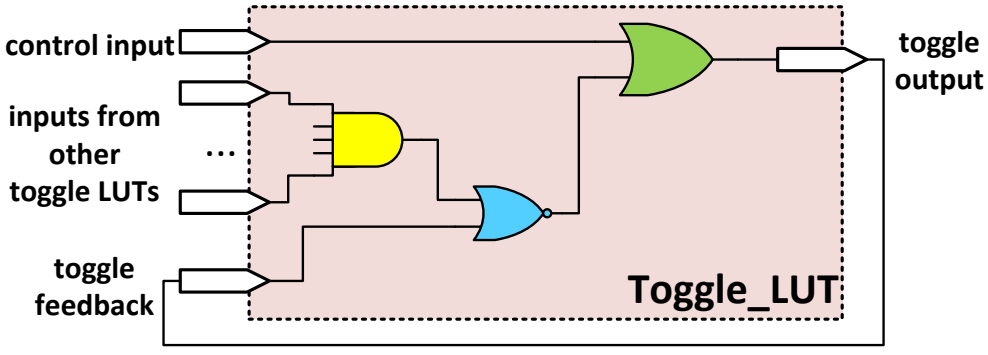


Figure 9.3.: Toggle logic with additional inputs

The control circuit depicted in Figure 9.4 sends the (`serial_out`) signal to all (`control_input`) ports of toggling LUTs inside the SHE. The circuit consists of a FF

working as a shift register to store the toggling control signal (`serial_in`) for one clock cycle. The (`FF_set`) signal is used to stop the toggling asynchronously by forcing the output of the FF to '1'. The "optimization preventing inputs" are connected to the feedback outputs of other toggling LUTs inside the SHE to prevent any possible deletion of their signal by the FPGA tools.

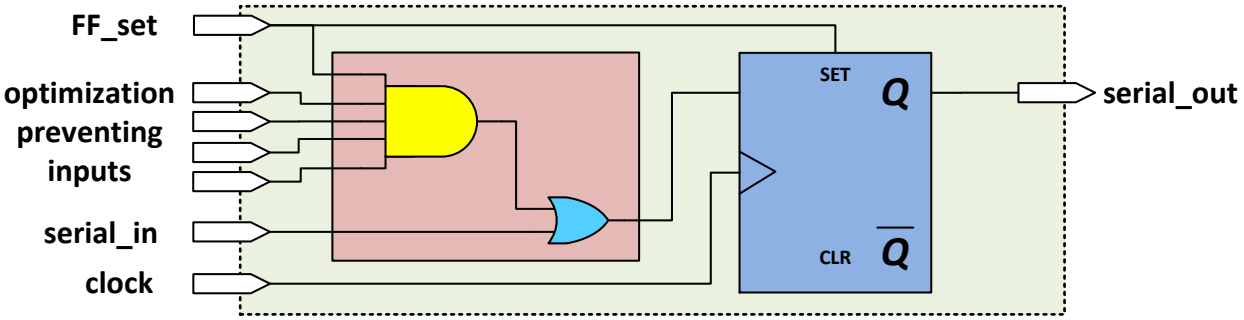


Figure 9.4.: The control circuit of the proposed SHE

9.3.3. Chain of SHEs: easier controlling

A single SHE can have as many toggling LUTs as possible. However, the larger the SHE the more routing complexity it poses, which can restrict its placement on the FPGA. Furthermore, in a single SHE, all of the toggling LUTs can be either in toggling mode or stopped, all at the same time. Thus, if an SHE contains too many LUTs, a fine control of the generated temperature cannot be achieved.

Based on the above statement, small SHEs (e.g., each with up to 10 toggling LUTs) are more preferable. Moreover, local routing can be used instead of global routing resources, and the `serial_out` signal has less routing overhead. Consequently, the footprint of each SHE is less and more space is left for any BIST circuitry to be implemented concurrently with the SHEs. Furthermore, it is easier to distribute small SHEs across the FPGA than the larger ones for better heat distribution and control.

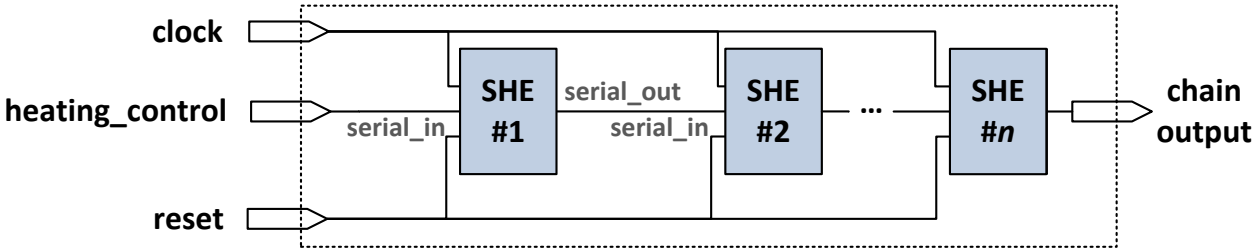


Figure 9.5.: Self-Heating Chain (SHC)

In order to have a simple control for the SHEs across the FPGA, the `serial_in` and `serial_out` ports of multiple SHEs are connected together resulting in a *Self-Heating Chain* (SHC), as depicted in Figure 9.5. The advantage of this connection is having the ability to control how many SHEs are toggling or stopped inside the SHC using a single control signal and hence, controlling the amount of heat generated from that SHC. This process is explained next. Furthermore, the SHC can contain an arbitrary number of SHEs, which gives more flexibility in its placement on the FPGA.

In each SHE in the chain, the toggling of the LUTs and the propagation of the `control_input` signal are delayed by the control circuit for one clock cycle, as mentioned earlier. Therefore, in order to make all SHEs in the chain toggle, after a reset, the first control input of the SHC (`heating_control`) has to be '0' for as many clock cycles as there are SHEs in the chain. A timing diagram showing this process is depicted in Figure 9.6.

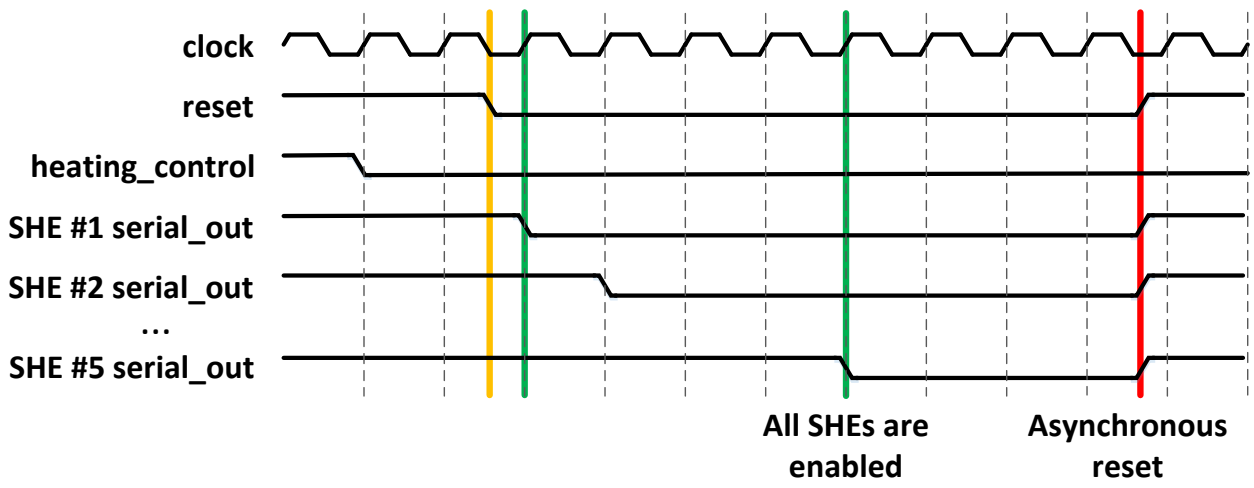


Figure 9.6.: Enabling the toggling of SHEs inside an SHC after reset

For controlling the amount of heat generated from the SHC, an input with a certain duty cycle can be applied to the `heating_control` signal, to switch the toggling of the individual SHEs inside the chain on and off. If, for example, only half of the SHEs are desired to toggle at the same time, the duty cycle of the `heating_control` signal should be set to 50%, as shown in Figure 9.7, using a chain with 5 SHEs.

While each chain could be driven by an individual control input in order to determine its power consumption and thereby the on-chip temperature distribution, a single input for all chains is used to control them. This simplifies the concurrent usage of self-heating with other BIST circuits on the FPGA.

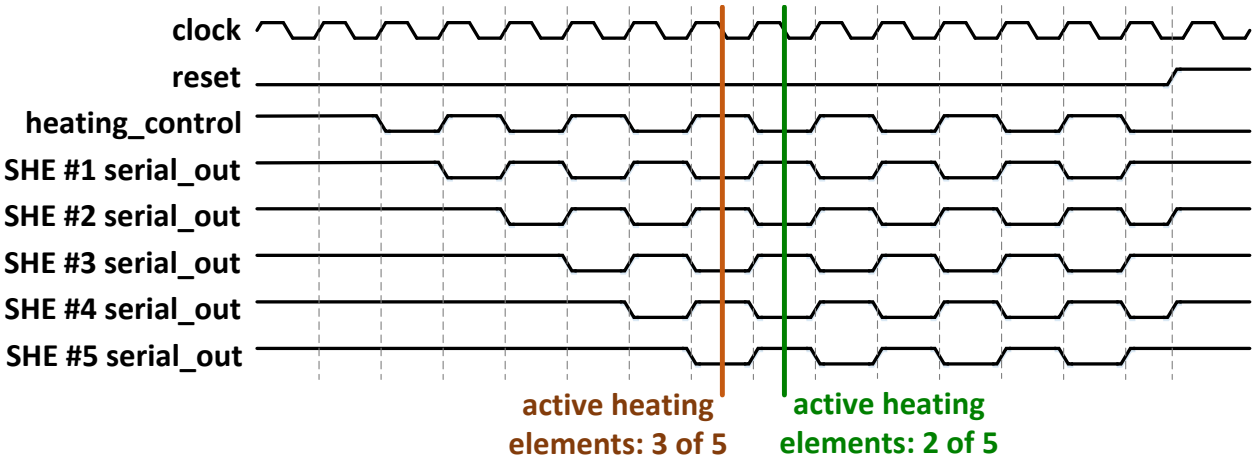


Figure 9.7.: Five self-heating elements of a SHC with 50% duty cycle

9.4. Thermal profile for testing

For applying a certain thermal-aware test on FPGA, a certain thermal profile is needed depending on the test requirements and type (e.g. application-independent vs. application-dependent). In this section, the process of arranging and controlling the SHEs on the FPGA to achieve a certain thermal profile is discussed.

9.4.1. Distribution of SHEs

The flexibility of SHEs and their arrangements in SHCs allow them to be distributed in different ways depending on the type of thermal-aware testing to be carried out on the FPGA components. If, for example, a certain thermal gradient is required to activate certain paths in particular regions of the chip, then the SHEs can be concentrated around these paths or regions. Figure 9.8 shows an example of using 359 SHEs concentrated in the upper-left corner of a Virtex-5 110T FPGA to generate a custom thermal profile. Each SHE consists of 7 toggling LUTs and additional LUT and FF for control. The thermal profile is obtained using the tool from Chapter 6. The internal thermal sensor of Virtex-5 FPGA is used additionally to verify the results of the thermal profile estimation tool. If other FPGAs than the Virtex-5 is to be used, a similar calibration to the one done in Chapter 6 must be carried out in advance. In Section 9.7, more details about the maximum temperature is discussed.

In case a homogeneous temperature profile over the entire FPGA is desired for the testing (which is usually the case for BIST schemes) a regular distribution of the SHEs is an important factor. However, the perfect regular distribution (i.e., with the same number of SHEs in each part of the FPGA) causes the temperature at the center of the FPGA to be higher than at the sides. Therefore, to generate a homogeneous thermal profile over the entire FPGA, it is necessary to balance the distribution of the SHEs by increasing the number of SHEs at the sides and decrease it in the center. By that, a homogeneous thermal profile over the entire FPGA is generated. This balancing is done gradually, where the increasing/decreasing of the SHEs at the sides/center is done linearly until a homogeneous

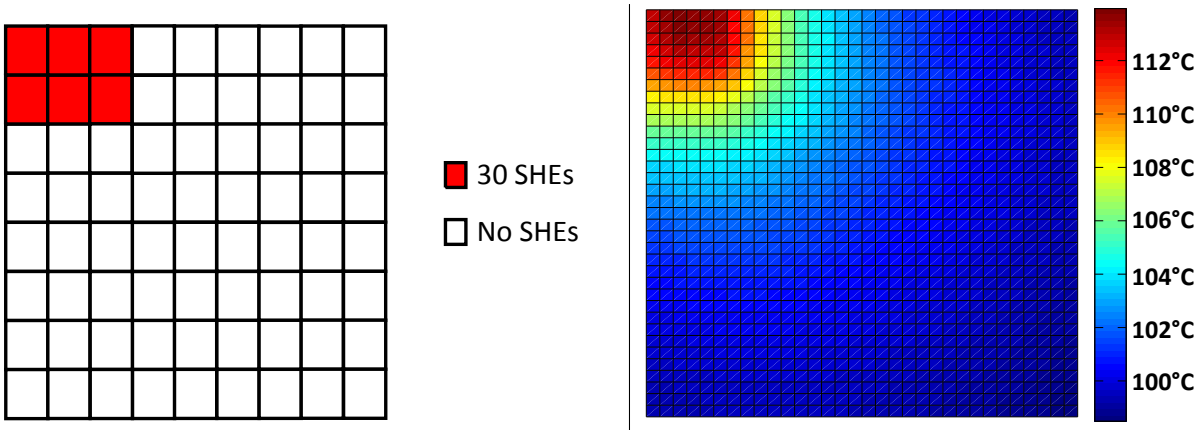


Figure 9.8.: A custom thermal profile generated by placing SHEs at a specified corner of the FPGA: (a) Distribution of the SHCs (Each rectangular box represents 2 SHCs. The reported number of SHEs is per each SHC) (b) The resulted thermal profile of the FPGA

thermal profile is achieved. Figure 9.9 shows an example of such balanced distribution of 1805 SHEs across the FPGA.

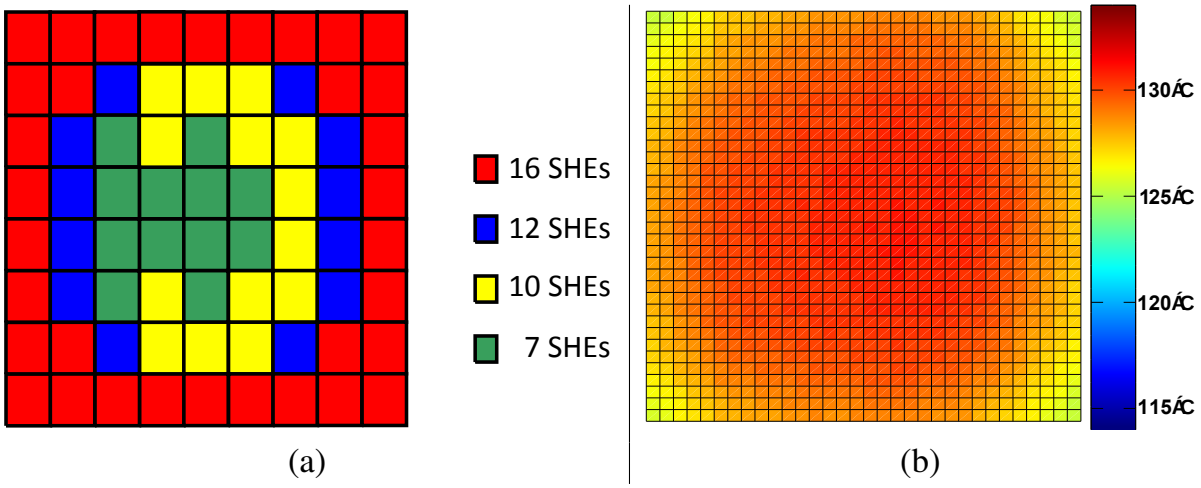


Figure 9.9.: A balanced distribution of SHEs: (a) Distribution of the SHCs (Each rectangular box represents 2 SHCs. The reported number of SHEs is per each SHC) (b) The resulted thermal profile of the FPGA

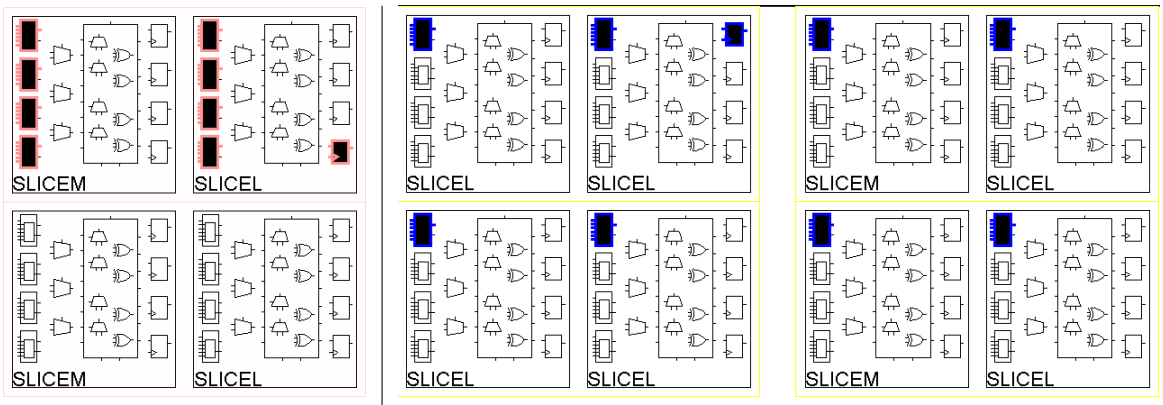
9.4.2. Calibration process

Based on the above discussion, the distribution of SHEs determines the shape of the thermal profile of the FPGA. However, the density of this profile (i.e. the maximum temperature) is

determined by the number of active SHEs in the design. The relation between the number of active SHEs and the desired maximum temperature depends mainly on the target FPGA device. Therefore, a pre-calibration must be carried out on the target FPGA to determine this relation. This can be done by testing different number of SHEs with different distributions and observe the resulted temperature. This information is used then in the design of the heating controller (Section 9.5.3.3). It should be noted that it is usually advantageous to integrate the maximum possible number of SHEs in the FPGA. The job of the heating controller is then to activate/deactivate a subset of SHEs according to the desired maximum temperature.

9.4.3. Integration of SHEs

Whether a homogeneous or custom thermal profile, in both cases it might be necessary to modify the test structure to release logic resources for the integration of the SHEs. This release process must take the desired distribution of the SHEs into account. This is discussed in details in Section 9.5.3 for a BIST case and in Section 9.6 for an application-dependent testing case. In fact, the structure of the proposed SHEs is very flexible for such integration. The way how the toggling LUTs inside the SHEs are connected, allows them to be distributed freely, leaving place for the original application to be routed around the SHEs. Figure 9.10 shows examples of such integration, where placement constraints are used to place the LUTs of the SHEs at certain locations inside the logic blocks, allowing the application to be placed freely at the other unused resources.



(a) (b)

Figure 9.10.: An SHE with 7 toggling LUTs and control circuit (additional LUT and FF) on Virtex-5 FPGA: (a) dense placement (higher heat generation) (b) spread placement (higher flexibility)

It should be noted that the distribution shown in Figure 9.10 is the distribution of the toggling LUTs inside a single SHE, which helps in integrating the SHEs in the test circuit. This process is independent of the process of distributing the SHEs themselves across the

FPGA to determine the shape of the desired thermal profile, which is discussed previously in Section 9.4.1.

9.4.4. Other thermal constraints

There are several other factors that should be considered before integrating the self-heating in the FPGA. These factors are related mainly to the environment in which the target FPGA is deployed. For example, it should be checked whether the board is able to supply the required current for raising the temperature of the FPGA, which can reach several amperes depending on the target FPGA and the desired temperature. Also, it should be checked whether raising the temperature of the FPGA will affect any other components on the board or in the surroundings. These constraints could be quite different for production test compared to in-field board test.

9.5. Self-heating application-independent BIST

The proposed self-heating approach can be combined with various BIST schemes for FPGA testing to create a thermal-aware BIST. In this section, the main steps for integrating the proposed self-heating for FPGA devices with BIST structures are explained. This is followed by a working example of such integration with a BIST structure for testing the logic resources in Xilinx Virtex-5 FPGAs.

Usually the BIST structure for FPGA devices consists of three main sets of components: i) *Test Pattern Generators* (TPGs) for generating the required input patterns for testing the targeted circuit, ii) *Blocks Under Test* (BUTs) which represent the blocks of the FPGA to be tested and iii) *Output Response Analyzers* (ORAs) which determine whether the outputs of the BUTs are valid (see Figure 9.11(a)). In order to test all the resources of the FPGA and achieve 100% coverage, several *test configurations* are generated in such a way that each resource appears in at least one test configuration as BUT. The integration of the self-heating with the BIST structure can be done in two ways:

9.5.1. Sequential method

In this method, the self-heating and the self-testing are done in two consecutive steps. First the FPGA is loaded with the SHEs (*heat configuration*) for heating up the chip to the desired temperature. Afterwards, the BIST test configuration is loaded on the FPGA for testing. In case the test requires multiple reconfigurations, this process of consecutive heating and testing is repeated before the application of each test configuration. The main advantage of this method is that there is no need to modify the original BIST structure. Additionally, the whole FPGA resources are available to the SHEs during the heating step. However, the controlling of the temperature can be done only during the execution of the heating step. During the testing step, the temperature will vary. Thus, depending on both the configuration time and execution time of each test configuration on the FPGA, the precision of the thermal control may decrease. Although the FPGA can be heated above the desired temperature during the heating step to compensate the cool down that may happen,

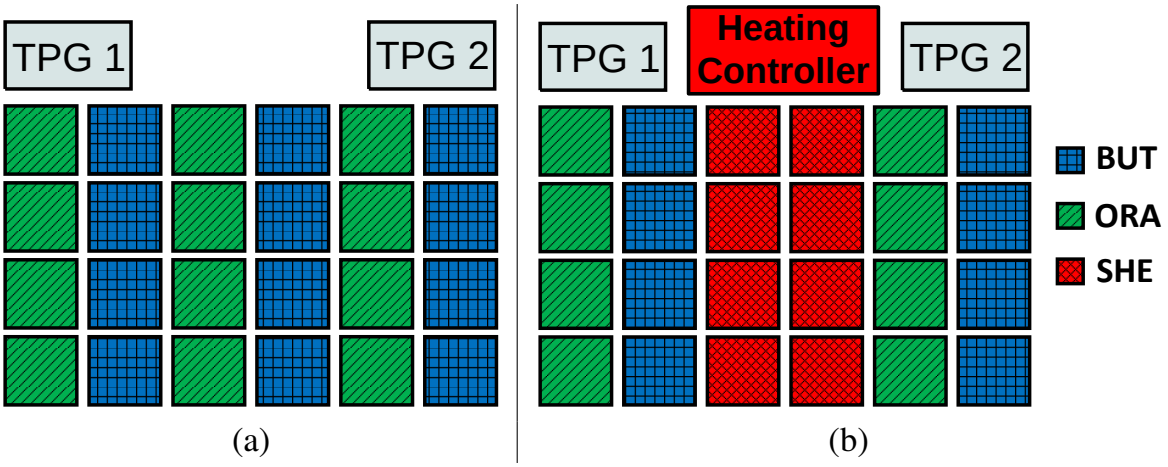


Figure 9.11.: (a) BIST components alone and (b) with integrated self-heating

this still does not allow accurate temperature control during the test steps. Using the sequential method, we have measured about 10°C cool down under the desired temperature, during the test steps, for the same experiment explained in Section 9.7.1.

9.5.2. Concurrent method

In this method, the SHEs are integrated with the BIST structure into a single *Heat&Test configuration*. This is done by adding a fourth set of components to the BIST structure representing the SHEs as shown in Figure 9.11(b). The main advantage of this method is the accurate control of the test temperature. Furthermore, the total number of required Heat&Test configurations, and hence the total test time, is less compared to the sequential method. However, this integration requires additional design efforts depending on the original BIST structure. The original BIST structure has to be modified in order to free up enough logic resources for the SHEs. This can be done in several ways, again depending on the targeted BIST structure: i) by reducing the number of BUTs that are tested at the same time and scaling down the other BIST components accordingly. This will also result in an increase to the total number of configurations, compared to those in the original BIST method, ii) by decreasing the size of the ORAs in such a way that the fault detection coverage is preserved but the diagnosis resolution is diminished. It should be noted that the process of modifying the BIST structure to release logic resources, must take the distribution of the SHE resources into account to generate the desired thermal profile.

Because of the main advantages of this method over the sequential method, the concurrent method is adopted in the rest of this chapter.

9.5.3. Case study BIST implementation

As a proof of concept, the proposed SHEs are integrated into an existing BIST scheme for Virtex-5 FPGAs. The selected BIST scheme from [107] can detect 100% of all stuck-at faults in every CLB.

9.5.3.1. The structure of the selected BIST

The CLBs in Virtex-5 FPGAs can either contain two SliceL or a pair of SliceL and SliceM as shown in Figure 9.12. In each slice there are a set of four 6-input LUTs and four FFs in addition to carry logic circuits. The SliceM incorporates all the logic functionality of SliceL and has additional circuitry to be used as distributed memory.

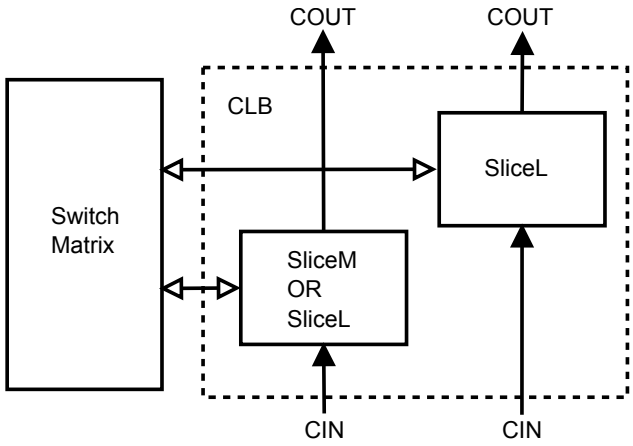


Figure 9.12.: Virtex-5 CLB [108]

The original BIST divides the test into three steps. First it configures half of the slices as BUTs and the other half as ORAs for the logic test. In the next step, it reverses the configuration in such a way that the slices, which were configured as ORAs are now configured as BUTs and vice versa, thus all the slices are tested. Finally, special configurations are loaded to test the additional functionality of all SliceM at once. Each of the first two steps uses 6 consecutive configurations to test the logic functionality of both SliceL and SliceM, and the final step uses 5 other configurations to test the additional functionality of SliceM. This makes the total number of required configurations equal to 17.

The TPGs that drive the inputs of BUTs for testing the logic functionality are realized using DSPs. The BUTs are divided into two partitions; each partition is fed by a different TPG. The outputs of each BUT from each partition are compared to an identical BUT from the other partition by an ORA as depicted in Figure 9.13. For testing the memory functionality of SliceM, the test patterns are stored in Block-RAMs and DSPs are used as address counters. For sake of simplicity, the integration of the SHEs is shown only for the logic testing steps (the first 12 configurations).

9.5.3.2. Modifications to integrate the SHEs

As all available slices on the FPGA are already used in the original BIST, the fraction of tested slices in each test step must be reduced for integrating the SHEs. As discussed before, half of the slices are configured to be BUTs, and the other half as ORAs in each step. In order to integrate the SHEs, the BIST is restructured in such a way that one third

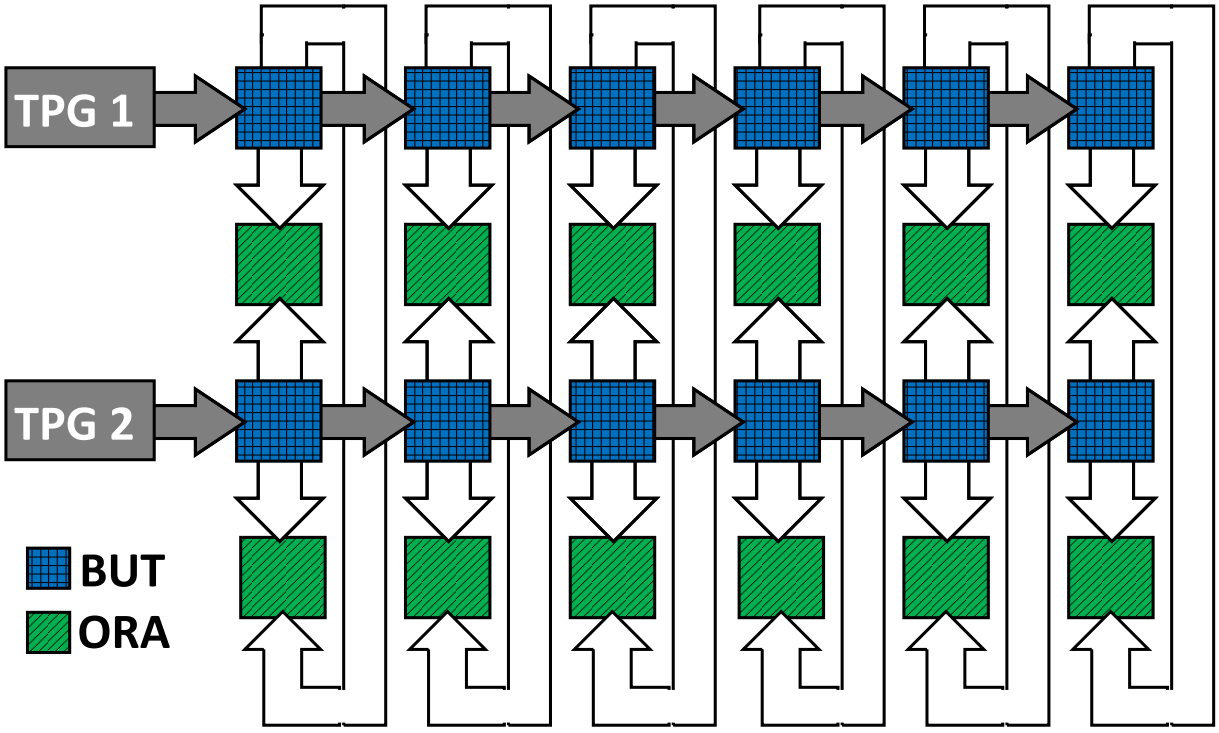


Figure 9.13.: BIST layout from [107]

of the slices are configured as BUTs, the second third as ORAs and the last third is used for the SHEs as shown in Figure 9.14.

This integration increases the number of required steps for testing all the slices on the FPGA from two steps to three. In each step a different set of slices are tested, which increases the total number of required reconfiguration from 12 to 18 (i.e. 6 reconfigurations in each step $\times$ 3 steps).

9.5.3.3. Heating control

The integrated SHEs are arranged in SHCs as explained in Section 9.3 to simplify the control of the heating. The heating controller shown in Figure 9.14 is basically a dynamic duty cycle signal generator. It takes as an input the desired testing temperature from the user and compares it to the FPGA chip temperature, which can be either read from the built-in thermal sensor available on the FPGA, or predetermined using FPGA thermal analysis tools (e.g. the tool presented in Chapter 6). The duty cycle of the generated `heating_control` signal is adapted according to the difference between the desired testing temperature and the measured chip temperature. The heating controller can be implemented either inside the FPGA or outside as an external component. As the proposed SHEs do not require the entire resources in their utilized slices (see Figure 9.10), the heating controller can be easily implemented inside the FPGA without posing any additional area overhead.

The distribution of the generated heat across the FPGA is controlled by the placement of SHEs on the FPGA. For this BIST example, a homogeneous temperature distribution across the FPGA is desired for thermal-aware testing of all the slices. Therefore, the SHEs

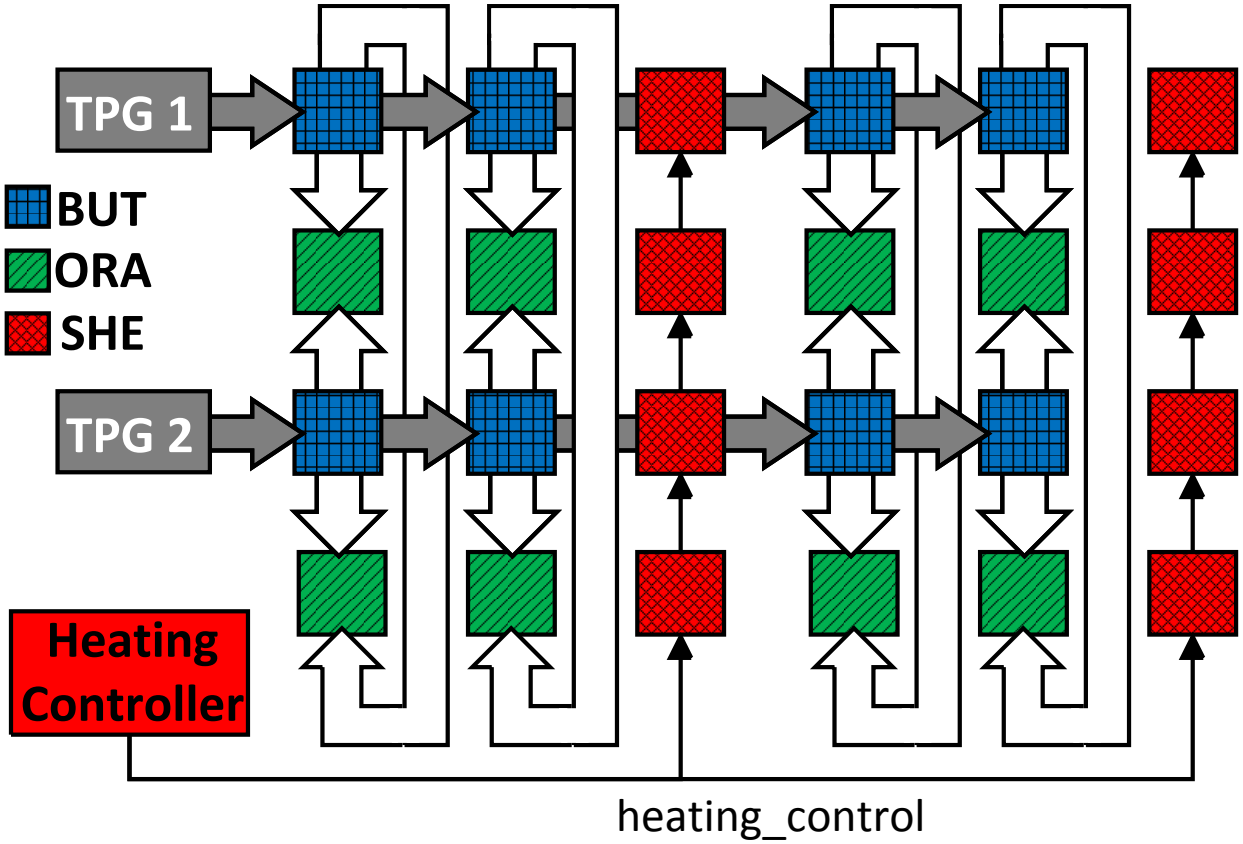


Figure 9.14.: Concurrent self-heating and BIST layout

are spread evenly as shown in Figure 9.15. In fact, this spreading causes the temperature on the edges of the FPGA to be slightly lower than at the center. To handle this, the length of the SHCs in the center is reduced to lower their heat generating capability and thus balancing the temperature across the FPGA.

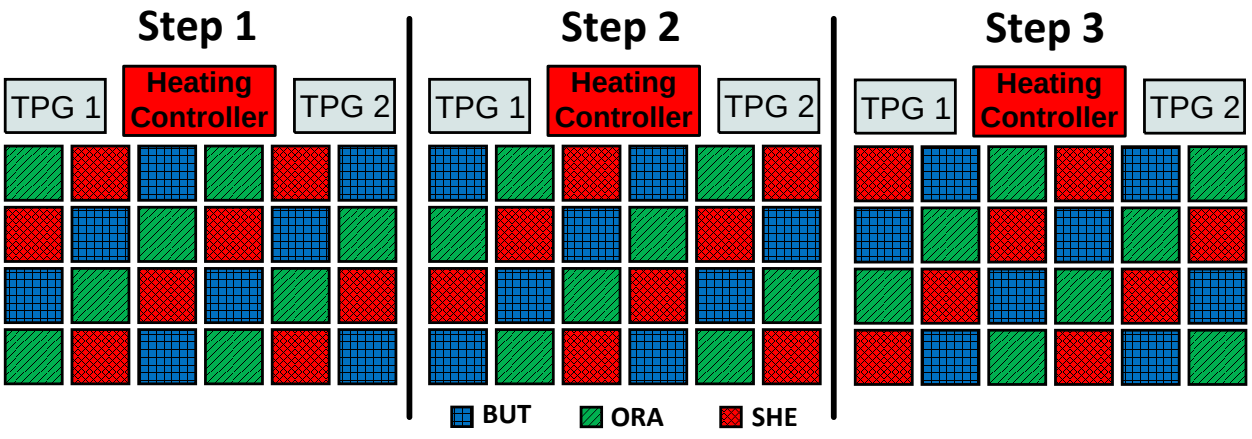


Figure 9.15.: Distribution of BUTs, ORAs and SHEs during the three different configuration steps for a homogeneous temperature distribution across the FPGA

9.6. Self-heating application-dependent testing

In addition to BIST, which is primarily used for application-independent testing, any application-dependent test can be integrated with the proposed SHEs for thermal-aware testing. This is particularly very useful for high-volume customers to check whether their design meets the thermal requirements and each chip is functional under different temperature corners. As mentioned previously in Section 9.4.3, the structure of the proposed SHEs is very flexible for such integration. In this section, different methods for this integration are discussed.

9.6.1. Application-dependent testing

Functional tests of the FPGA resources (e.g., the BIST example in Section 9.5.3) can be used to determine if the hardware complies with its specification. Sometimes it is more important to find out whether a single application can be executed correctly under certain conditions. Such specific conditions cannot be activated during the general application-independent manufacturing test due to test time constraints. For example, if a given application does not use the CLBs SliceM functionality, it seems pointless to run the additional memory test configurations to verify the SliceM's behavior. One alternative approach would be to do a set of functional tests for all used resources so that all utilized functionality can be verified.

Another approach (which is targeted in this section) is to verify that the implemented application delivers the correct output values for all possible input combinations. It may be the case that the application functions perfectly at low (room) temperature but fails at high temperatures, for example, resistive opens or reduced transistor mobility aggravated at high temperature leading to intermittent delay faults. The advantage of application-dependent testing is that fewer reconfigurations are needed to apply the test compared to a complete functional test of all resources. Later in Section 9.7, the total test time will be discussed. There, it will become clear, that the time needed for the reconfigurations of the FPGA can be the dominant factor in total test time, if a lot of test configurations have to be executed.

In order to point out the flexibility of the proposed self-heating scheme, the proposed approach has been adapted to heat up only the areas of the FPGA where the application is placed. As the implementation of the application is not stripped down to just the critical path, the application is still able to be used in a normal way. This is unlike the method in [104] where just the critical paths of failing PLDs are heated up by using short-circuits, which is done in order to determine more easily if elevated temperature levels are the sole cause for the application failures on these devices.

The goal here is to show that it is possible to augment an application with SHEs without affecting the application's performance considerably. This way, the application can either run normally, without the SHEs being active, or with active SHEs providing concurrent self-heating. Using the latter case, the part of the FPGA that is used by the application is heated up. Then, the impact of high temperature levels on the behavior of the application is analyzed.

9.6.2. SHEs integration methods

The first condition to integrate the SHEs in a certain FPGA-mapped circuit (i.e., a certain application) is that enough free resources should be available on the FPGA for this purpose. The redundancy of the FPGA resources together with the complexity of placing and routing highly utilized FPGAs; yield the fact that there exists almost no practical application that can utilize 100% of the FPGA resources. It is even recommended to limit the FPGA utilization to 50% if a peak performance is desired [109]. This means that for many practical applications, there are always free resources on the FPGA chip that can be used for the self-heating integration. However, the integration methods may differ from one application to another depending on the availability of unused resources.

In the following, different methods are presented for integrating the SHEs in a given application on the FPGA. It should be noted that these proposed methods are not all suitable for all sorts of applications. There are two measures to decide whether a certain proposed method is suitable for a particular application. The first measure is, as discussed before, the amount of available unused resources and the distribution of these resources across the FPGA. The second measure is the impact of the self-heating on the original application performance. The performance is measured by the minimum delay that is necessary to run the application. Lower minimal delays mean the application can be executed at higher clock speeds.

9.6.2.1. Ring placement

In this method, the mapped-circuit is constrained to be placed at the center of the FPGA, while the SHEs are mapped to the sides (see Figure 9.16). The heat generated by the SHEs at the sides raises the temperature of the circuit in the middle. The advantage of this method is that the SHEs and the application are separated, which causes the minimum overhead to the application performance. It is obvious that this method is ideally applicable for applications that utilize less than 50% of the FPGA resources. This is to allow enough number of SHEs to be integrated for a wider range of testing temperatures.

9.6.2.2. Rows or columns placement

This method constrains the SHEs to pre-specified columns or rows with regular spacing, leaving the free gaps between them to map the application (see Figure 9.17). The advantage here is to allow a more custom thermal profile within the application. However, this reduces the routability of the FPGA-mapped application, which in turn affects the application performance.

9.6.2.3. Stripping placement

As discussed previously in Section 9.4.3, this method exploits the structure flexibility of the proposed SHEs to strip the SHEs down to their toggling LUTs, where the placement freedom is higher. The toggling LUTs of each SHE are distributed across several slices as required, which leaves the rest of the resources in each slice available for the mapped design

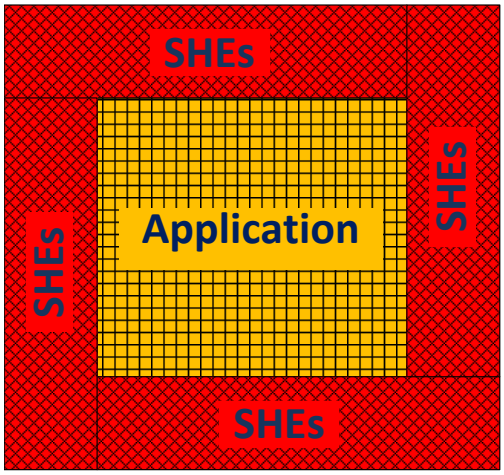


Figure 9.16.: Ring placement of the SHEs around the FPGA-mapped application

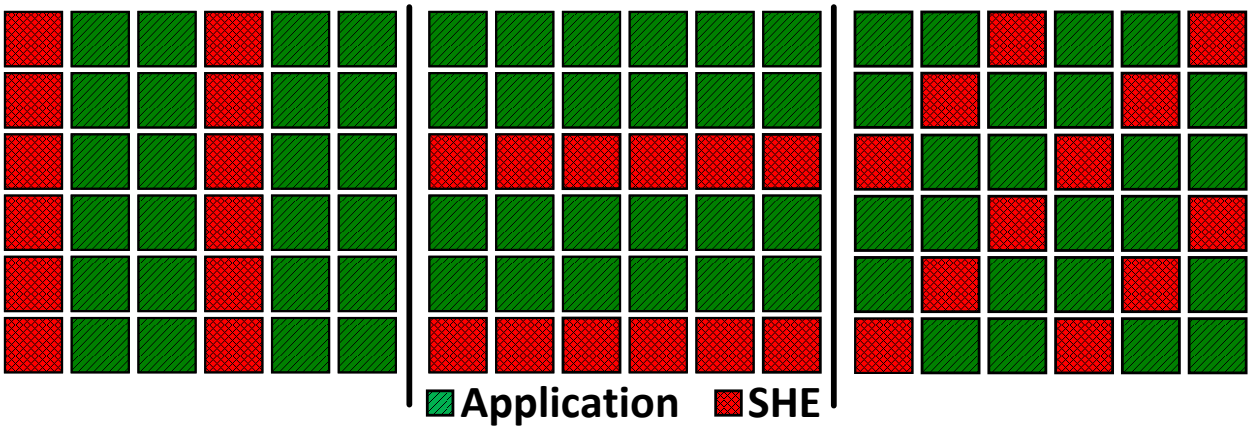


Figure 9.17.: Rows, columns and checkerboard placement of the SHEs through the FPGA-mapped application

(see Figure 9.10). In a different perspective, the self-heating elements are distributed and mapped to any unused resources (down to individual LUTs within partially-used slices) to be as non-intrusive as possible to the original mapped design. The advantage of this method is that it reduces the performance overhead significantly in comparison to the previous method (rows or columns placement). However, this comes at the cost of more design efforts.

It should be noted that since the individual delay time of a transistor is affected by process variations during its fabrication, and the delay time also increases with higher temperatures, the temperature at which an application actually begins to fail can also differ between devices of the same make and model. Therefore, the self-heating should be able to generate a wider range of temperatures, thus enough number of SHEs (i.e., depending on the calibration process) must be integrated in the design.

9.7. Experimental Results

9.7.1. Self-heating application-independent BIST

The self-heating BIST example discussed in Section 9.5.3 is implemented on a Xilinx XUPV5 development board. This board features a Virtex-5 LX110T FPGA, as well as several interfaces and peripherals. The 18 configurations for the self-heating BIST were generated as discussed in Section 9.5.3.2 using ISE 12.2 tool-chain. It should be noted that no modifications are made to the default packaging and heat-sink delivered from the vendor.

The heating controller is implemented inside the FPGA using the free resources left by the SHEs, and designed to heat the FPGA to different target testing temperatures. These can be selected through user input that is fed to the controller through on-board switches. The clock frequency for both BIST and SHEs circuits is set to 100 MHz. The built-in thermal sensor of the FPGA is also observed using the Xilinx ChipScope software to validate the controller operation. ChipScope is also used to load the 18 configurations onto the FPGA via a USB-to-JTAG interface cable. A single pass/fail bit is sent to one of the LEDs on the board to indicate if faults have been detected. In case of positive feedback, the status of the ORAs is read through an *Integrated Logic Analyzer* (ILA) core for fault diagnosis and localization.

A total number of 1805 SHEs are integrated with the BIST, each of which consists of 7 toggling LUTs in addition to the control circuit (as in Figure 9.10(a)). The simulated toggling frequency of the SHEs is reported as 500 MHz by Xilinx power tool. This integration increases the total power consumption of the FPGA from ≈2.5 W for the original BIST circuit to ≈14 W for the self-heating BIST. The resulted thermal profile using the SHEs is estimated, using the tools from Chapter 6. The thermal profile of the original BIST circuit (without the SHEs) is depicted in Figure 9.18(a). By integrating the SHEs as shown in Figure 9.15 for two of the three steps, a maximum temperature range between 126°C and 132°C is reported across the FPGA as seen in Figure 9.18(b). However, as Virtex-5 LX110T is rated to work under a maximum temperature of 125°C, the board is switched off when the built-in thermal sensor reports 120°C to avoid any possibility of damaging the chip.

The measurement of the total test time depending on the targeted testing temperature is shown in Figure 9.19. The gaps in this temperature measurement on x-axis are the result of ChipScope tool not being able to capture the output of the built-in thermal sensor during FPGA reconfiguration.

The total test time can be divided into 4 parts: i) the initial heat-up time to the targeted BIST temperature, ii) the BIST run-time, iii) the reconfiguration time after each Heat&Test configuration and iv) the re-heating time to the targeted testing temperature again after the temperature dropped during the previous reconfiguration. While the time to reconfigure and to run the BIST is independent from the targeted temperature, the initial heat-up time rises with the temperature. The re-heating time after each reconfiguration also rises with the temperature but as the re-heating has to be done 18 times, its impact on the total test time increases more considerably with increasing targeted BIST temperature. In Figure

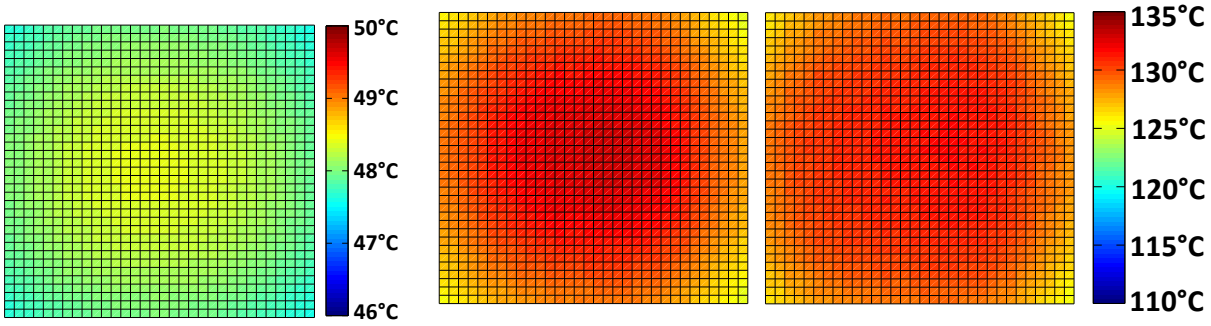


Figure 9.18.: Thermal profile of the Virtex-5 FPGA: (a) the original BIST and (b) the modified BIST (with SHEs)

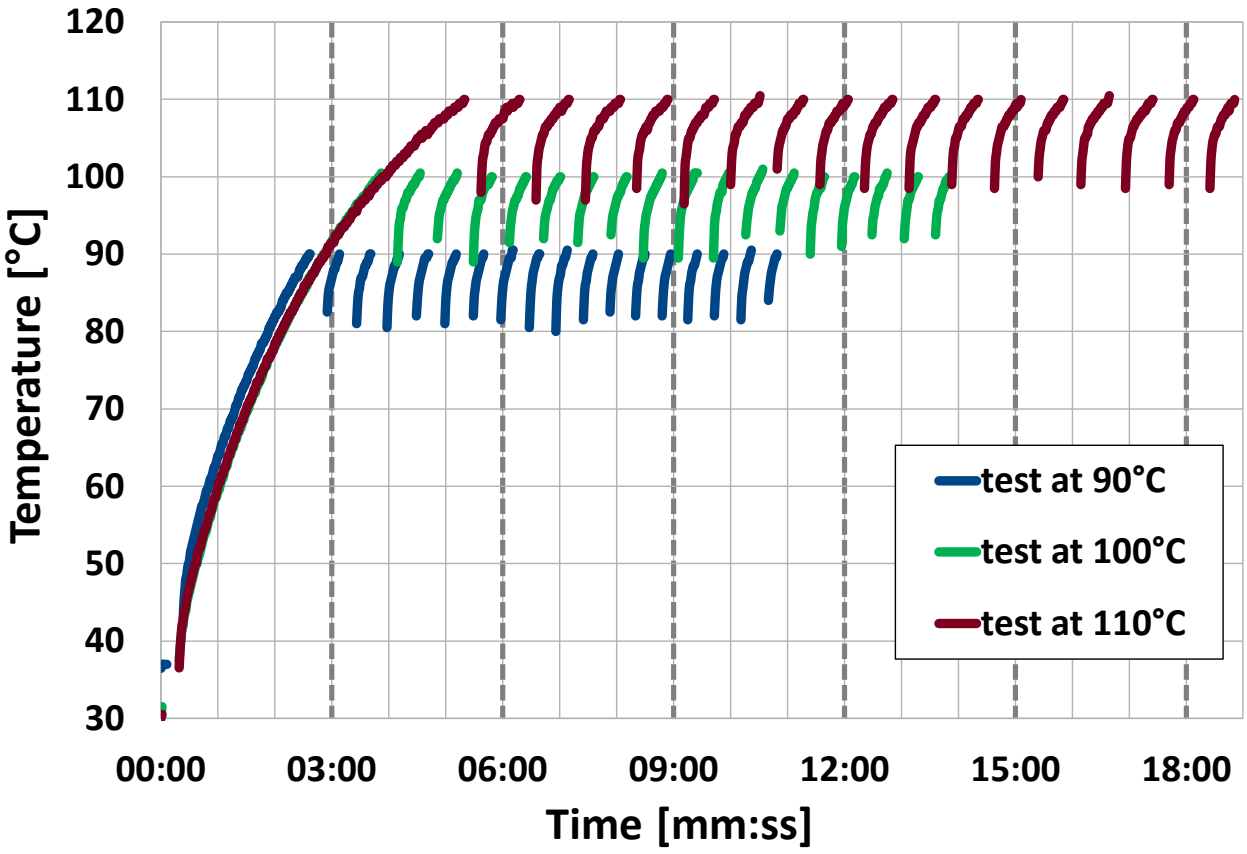


Figure 9.19.: Total test time depending on targeted BIST temperature

9.20 the re-heating time of the 17 configurations after the first configuration is shown at different targeted BIST temperatures. Later configurations generally need less time to re-heat, because the thermal capacity of the heat-sink is saturated. This can be seen more clearly at higher temperatures.

Assuming the original BIST was programmed to the FPGA the same way over JTAG, the overhead caused by merely adding the SHEs to the BIST can be compared. Even if the heating time is ignored, the configuration time is 50% more, because now 18 configurations are required for testing all the CLBs instead of 12 in the original BIST. Depending on the

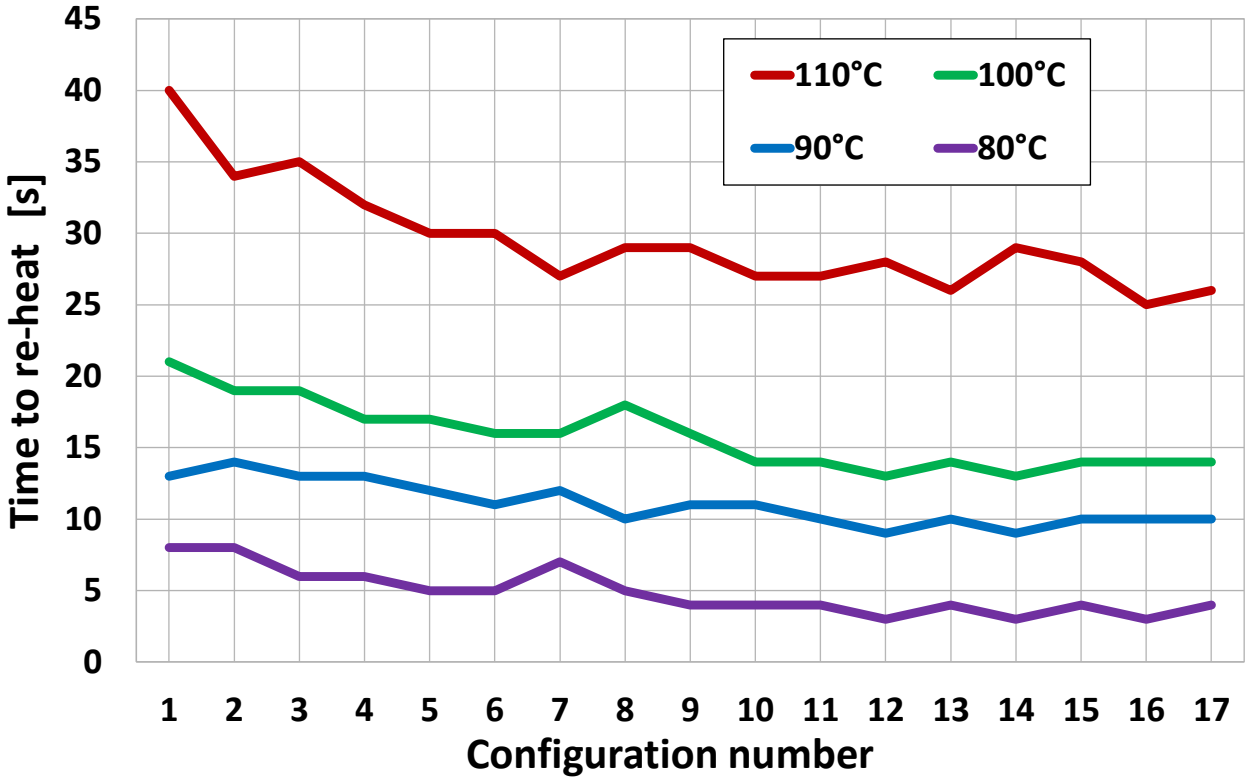


Figure 9.20.: Time to re-heat after each reconfiguration depending on targeted BIST temperature

targeted BIST temperature, the increase in the total test time may grow further. This is because the test application now is not executed right after the completion of reconfiguration but just after the FPGA has reached the desired temperature. This increase is reported in Table 9.1. It should be noted that for a fair comparison, the testing time should be compared with that of a real heating chamber, which also needs time for heating up the FPGA. Although the increase in testing time looks large, the advantages of this method over using the external heating devices (e.g., the ability to apply the test to large number of FPGAs simultaneously, the possibility for in-field testing by the user, the cost reduction, etc.) make this method more feasible in many domains.

Table 9.1.: Increase in testing time caused by implementing the proposed SHEs alongside the BIST in connection to the targeted BIST temperature

Target temperature	Increase in testing time
Reconfiguration time alone	50%
80°C	57%
90°C	80%
100°C	88%
110°C	115%

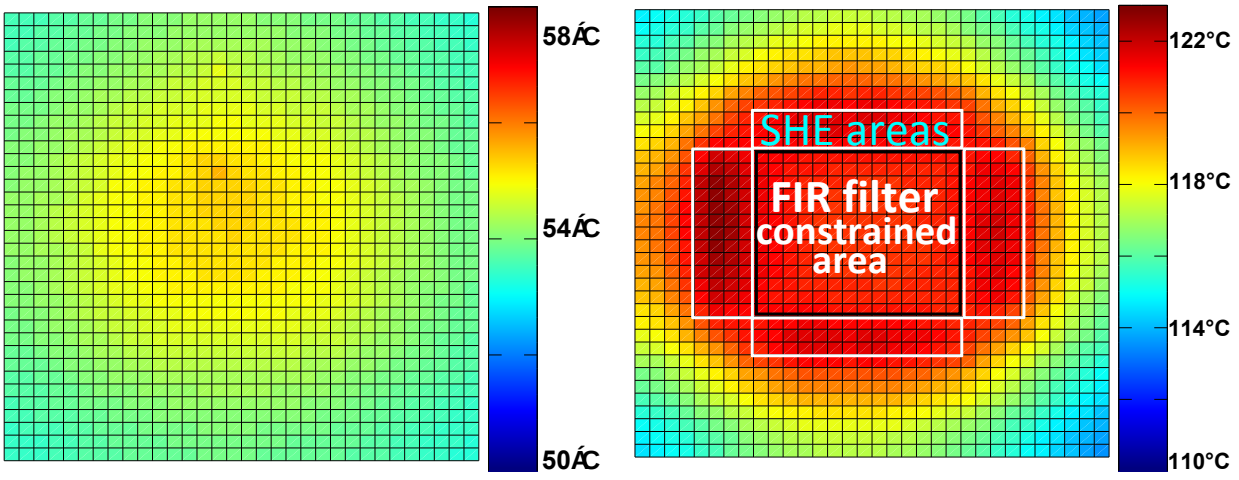
9.7.2. Self-heating application-dependent test

To compare the different self-heating integration methods for application-dependent testing, discussed in section 9.6.2, a 16-bit FIR filter is used as a test case. The chosen FIR filter has 128 *taps* that are concatenated to form the 128 stages of a digital 128_{th}-order discrete-time FIR filter.

To implement the FIR filter on the Virtex-5 FPGA, 5,888 LUTs and 2,048 FFs are needed. The option that allows the ISE tool-chain to use DSP resources for faster summation and multiplication is turned off to allow a fair comparison between the different integration methods. Another 917 LUTs and 23 FFs are used to implement a *Linear Feedback Shift Register* (LFSR). The LFSR acted as stimulus process for the FIR filter that feeds the filter inputs with pseudo random data.

The FIR filter is placed in a constrained area in the middle of the chip. The achievable clock speed that is still safe to run the filter correctly is between 144 MHz and 156 MHz. When constrained to an area with significantly less than 11,400 free LUTs (i.e., a utilization rate higher than 52%), the ISE tool-chain is no longer able to implement the FIR filter.

72 SHCs are added to heat up the application. Each SHC contains 20 SHEs and every element consists of seven toggle LUTs and one control LUT plus a FF. A total of 1,440 SHEs are used that require 11,520 LUTs and 1,440 FFs to be implemented. The integration methods discussed in section 9.6.2 are implemented and their respective impact on the application performance is analyzed. Figure 9.21 shows the result of implementing the ring placement method.



(a) (b)

Figure 9.21.: Thermal profiles: (a) FIR filter alone (b) FIR filter and active SHEs (ring placement)

The initial thermal profile of the FIR filter alone executing at 150 MHz is shown in Figure 9.21(a). There, the central area where the filter is placed is 56°C, which is only

slightly warmer than the rest of the chip. The absolute temperature deviation is below 2.3°C. By using all SHCs with the maximum duty cycle, the area in which the application was placed can be heated up to 120°C. This can be seen in Figure 9.21(b). The center where the FIR is located is heated in a rather homogeneous way, which is exactly as desired. However, in some of the SHE areas there are hot spots. Therefore, an extra care has to be taken to avoid overheating in some parts of the FPGA during any self-heating application-dependent testing.

In order to have a fair comparison among all the integration methods, the delay overhead of these methods has been compared at different utilization rates. This is because higher utilization rates have a negative influence on the application performance, and this role should be overridden in the comparison. To achieve that, the size of the constrained area, where the FIR filter is mapped, is changed in such a way that different levels of utilization rate for the mapped FIR filter on the constrained area are reached (between 25% and 52%, see Figure 9.22). Then, in each case, the exact same number of resources as needed to implement the SHEs in the ring placement is added to this constrained area. The utilization rate is calculated based on the free resources left for the FIR filter in the constrained area after subtracting the resources needed by the SHEs.

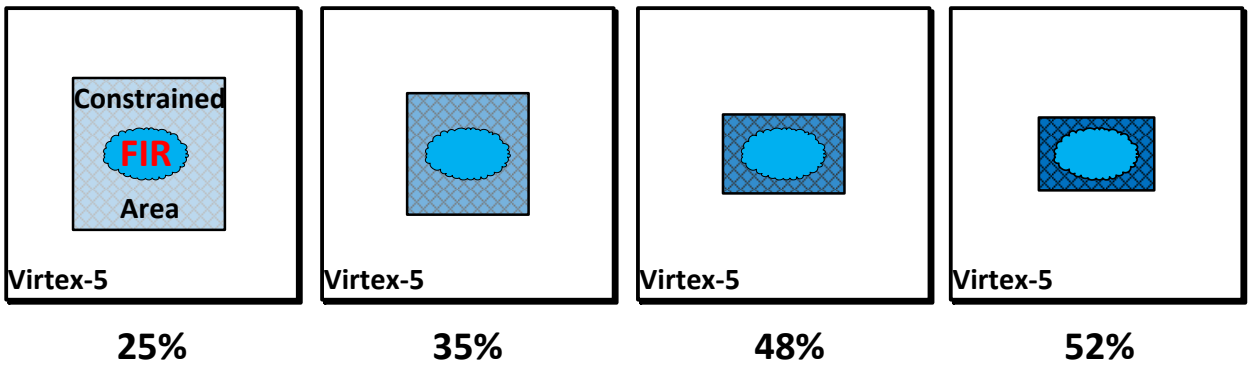


Figure 9.22.: Different sizes for the constrained area on Virtex-5 FPGA to achieve different utilization rates for the mapped FIR filter into this area

Figure 9.23 shows the worst case delays for the different integration methods and Table 9.2 shows the associated relative performance degradation. For each utilization rate the scenario where the FIR filter alone is implemented is used as reference.

As anticipated, the maximum clock speed of the application drops slightly with increasing the utilization rate, even without the self-heating circuitry. The impact of the ring placement method on the FIR performance is the minimum among all the methods for various utilization rates. This is obvious because of the separation of the SHEs and the application as discussed previously in Section 9.6.

For the rows and columns placement methods, where the SHEs are constrained as fixed rows or columns inside the application area, they result in the highest performance impact

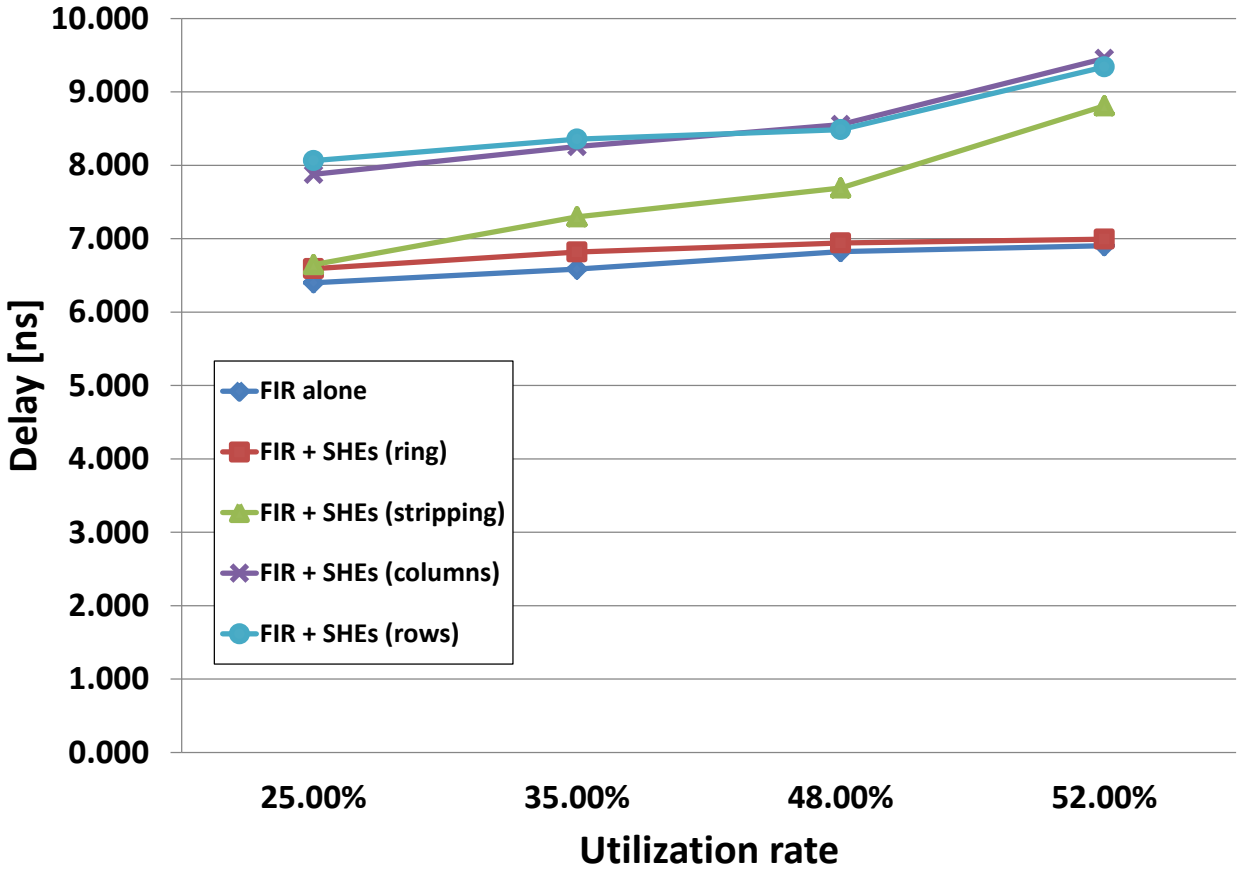


Figure 9.23.: Impact of different SHE integration method and utilization rates on the application delay

Table 9.2.: Relative performance degradation of the FIR filter for different SHE integration methods

Utilization rate (Max frequency)	**SHE integration method**			
	Ring	**Stripping**	**Columns**	**Rows**
25% at (156MHz)	3.00%	3.84%	23.10%	26.02%
35% at (151MHz)	3.54%	10.84%	25.38%	26.89%
48% at (146MHz)	1.76%	12.74%	25.36%	24.40%
52% at (144MHz)	1.30%	27.62%	36.95%	35.28%

among all these methods. Especially at utilization rate of 52% where the performance drops considerably compared to the ring placement method.

For the stripping placement method, when the utilization rate is low, the performance impact is acceptable. However, this impact is much less than the rows and the columns placement methods. While this method is not the best in terms of performance overhead, it allows the design to be placed and routed at much higher utilization rates (using this method, it was possible to map the FIR filter at a utilization rate of up to 73%). This means that in terms of area requirements, particularly when the utilization rate is very high, this is the most efficient method.

9.8. Summary

In this chapter, self-heating integration techniques for thermal-aware testing of FPGA devices have been presented, in which the internal resources of FPGA are used to build controlled SHEs. Thus, no external devices for heating up the FPGA are needed. Both application-independent and application-specific thermal-aware BIST-based testing of FPGAs are presented. The techniques are applied on representative test cases. The experimental results show that a wide range of maximum chip temperatures can be achieved (from 50°C up to 125°C on Virtex-5 FPGA) with a high accuracy ($\pm$ 1°C).

Part III.

Mitigation

Chapter 10.

Aging Mitigation in LUTs

10.1. Introduction

As discussed in Chapter 2, LUTs form the basic reconfigurable blocks for mapping and implementing logic circuits in FPGAs. Due to this important role, it is very necessary to keep the aging rates of the LUTs as low as possible. In Chapter 3, the impact of BTI on different implementations of 2-input and 4-input LUTs was studied. The analysis revealed that BTI-induced aging on LUTs has a direct relationship with: 1) actual LUT configuration, 2) the configuration history of the LUT, 3) input signal probabilities, and 4) the architecture of the LUT. Based on that study, changing the LUT configuration and/or its input signal probabilities can change the degradation rate significantly.

In this chapter, it is made use of these results to propose a method for mitigating BTI-induced aging in partially and fully used LUTs. The main idea is to alter LUT inputs and LUT configuration to select the configuration with the minimum aging effect while preserving the logic functionality of FPGA-mapped circuit. The proposed method has been implemented and validated using the academic tool VTR [110]. The results show that for different benchmark circuits with different sizes and functionality, the proposed method can mitigate BTI aging effect by about 20% on average (translated to 2X improvement in lifetime of FPGA-mapped designs).

The rest of the chapter is structured as follows: Section 10.2 reviews the related work. Section 10.3 gives the motivation for this chapter. Then, Section 10.4 presents the proposed methods for mitigating the aging of LUTs. Afterward, Section 10.5 details the implementation flow. Later, Section 10.6 presents the observations and the results of the study. Finally, Section 10.7 concludes the chapter.

10.2. Related Work

There exists few related work for degradation mitigation in LUTs. The authors in [111] proposed inverting the configuration bits of LUTs on a periodic base to relax the transistors and mitigate the NBTI effect. This technique however, requires additional overhead due to the need for reconfiguration and rerouting. The authors in [53] proposed exploiting unused and partially used LUTs to replace the worn-out LUTs. This can be applied only to a certain set of the LUTs that have such unused and partially used LUTs in the same cluster where they exist.

10.3. Motivation

In Chapter 3, an investigation is made for BTI-induced aging on three different 2-input and 4-input LUT architectures. These are: PT-based, TG-based and LG-based structures. Table 10.1 shows an example of the results obtained for the TG-based LUT.

Table 10.1.: Normalized BTI-induced Δdelays for 4-input LUT with TG structure

LUT Config.	LUT Input Signal Probabilities				Normalized ΔDelays		
	A	B	C	D	NBTI	PBTI	BTI
0000	0	0	0	0	8.01%	7.19%	10.50%
0000	0	0	0	0.1	7.97%	7.19%	10.46%
...							
0000	1	1	1	1	7.98%	7.17%	10.48%
0001	0	0	0	0	3.60%	3.17%	5.68%
...							
FFFF	1	1	1	0.9	4.45%	6.47%	6.10%
FFFF	1	1	1	1	4.46%	6.49%	6.12%

The strong dependency of LUT delay degradation on the mapped configuration as well as input signal probabilities motivates the idea of the work in this chapter. If one can change the configuration of each LUT in the circuit to another configuration that has minimum aging effect, while preserving the logic functionality of each LUT, aging of the whole circuit will be mitigated. This idea is described in details in the next section.

10.4. Methodology

According to the observations of Chapter 3, LUTs with different configurations and different input signal probabilities have different aging-induced delay degradation. The objective here is to transform a LUT configuration from one pattern to another, to mitigate aging without affecting the logic function that it implements. In order to do so, two methods are proposed that are described next:

10.4.1. Method 1: Manipulating partially-used LUTs

In FPGAs, not all mapped logic functions can fully utilize LUTs. This is because some functions need fewer inputs to be implemented. Therefore, there are some LUTs which are partially used. An LUT that has unused inputs will naturally contain *don't-care* configuration bits. The idea here is to change the signal probabilities of any unused LUT input (i.e. in partially used LUT) and also to alter the don't-care bits in it to obtain better

aging-induced delay degradation for that LUT. The methodology to change an LUT configuration bit-pattern using "Method 1" (unused inputs and don't care bits manipulation) is illustrated in Figure 10.1 and 10.2. In both figures, the scenario chosen is a 4-input LUT with 2 unused inputs. While Figure 10.1 only clarifies the availability of the don't-care bits (denoted by X) that can be exploited for the given scenario in the configuration bitstream, Figure 10.2 illustrates how the signal probabilities of the unused inputs (denoted by Y) and the don't care bits together can be manipulated iteratively to generate a wide range of new LUT configurations.

In Figure 10.1, it is assumed that both unused inputs (C and D) are connected to logic 0 and the default LUT configuration for this scenario is "XXXX XXXX XXXX 1000". As shown in this figure, there are $2^{12} = 4096$ different possible configurations supporting the same logic functionality, but each of them has a different post-aging* delay as described in Section 10.3. Among the wide range of possibilities available, the LUT configuration with the best post-aging delay is picked as the replacement configuration. There is a possibility to also exploit the effect of input signal probabilities in the optimization of delay degradation. By connecting an unused input permanently to logic 0 (Gnd) or logic 1 (Vdd), its signal probability is set to 0.0 and 1.0, respectively. This provides a wider search space for optimization, as shown in Figure 10.2. This will increase the search space to about $4 * 2^{12}$ different combinations. It should be noted that, the new LUT configuration still supports the same logic functionality as the original LUT configuration. Therefore, the functionality of the design does not change by using this method. The only restriction of this method is that it requires partially-utilized LUTs. Thus, "Method 1" cannot be applied on fully used LUTs to mitigate aging effect.

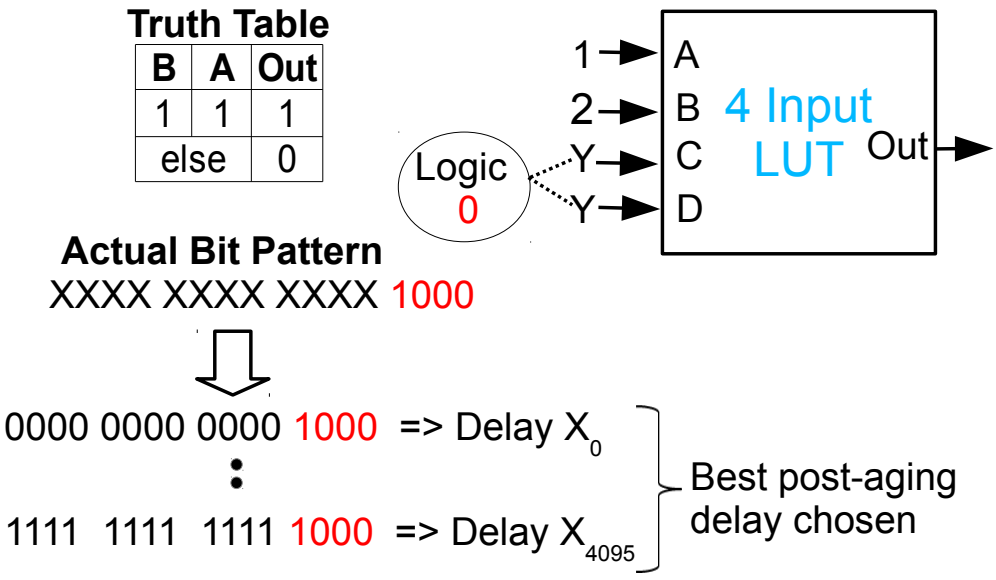
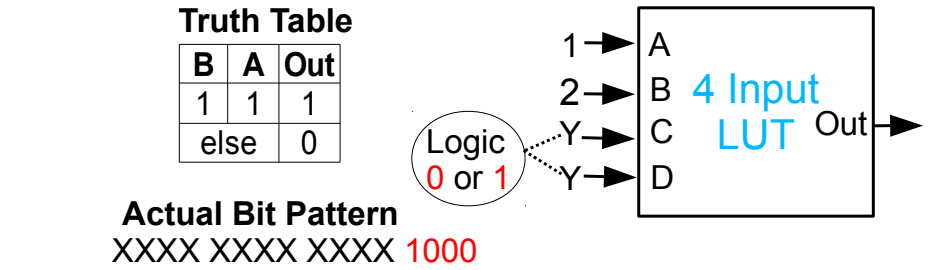


Figure 10.1.: Manipulating LUT configuration based on the available don't-care bits

*"Post-aging" delay refers to LUT delay after 3 years, from start of operation.



Actual Bit Pattern
XXXX XXXX XXXX 1000

Port D Signal Logic	Port C Signal Logic	LUT Bit Pattern	Don't care Manipulation
0	0	XXXX XXXX XXXX 1000	0000 0000 0000 1000 to 1111 1111 1111 1000
0	1	XXXX XXXX 1000 XXXX	0000 0000 1000 0000 to 1111 1111 1000 1111
1	0	XXXX 1000 XXXX XXXX	0000 1000 0000 0000 to 1111 1000 1111 1111
1	1	1000 XXXX XXXX XXXX	1000 0000 0000 0000 to 1000 1111 1111 1111

Figure 10.2.: Manipulating LUT configuration based on the signal probability of the unused inputs and the corresponding don't-care bits

10.4.2. Method 2: Swapping LUT inputs

In this method the LUT inputs are swapped, mapped to different ports and accordingly the configuration of the LUT is changed to obtain same functionality but with better post-aging delay. This is illustrated in Figure 10.3 with an example scenario. The scenario shows how the original LUT configuration bit-pattern of "1000 1000 1000 0000" can be transformed to "1110 0000 0000 0000" by just connecting the LUT inputs to different ports. Unlike "Method 1", "Method 2" can be applied to both fully and partially used LUTs. The number of alternate LUT configuration bit-patterns that can be generated depends on the exact scenario at hand. For example, if a 4-input LUT is partially used with just one utilized input port, then there are only 4 ways to connect that input to the various LUT ports, and thus only 4 configuration bit-patterns can be generated (this of course if "Method 1" is not considered). However, for a 4 input fully utilized LUT, the total number of alternate configuration bit-patterns that can be generated is 4! = 24. As in "Method 1", the configuration bit-pattern that yields the best post-aging delay is chosen as the replacement one.

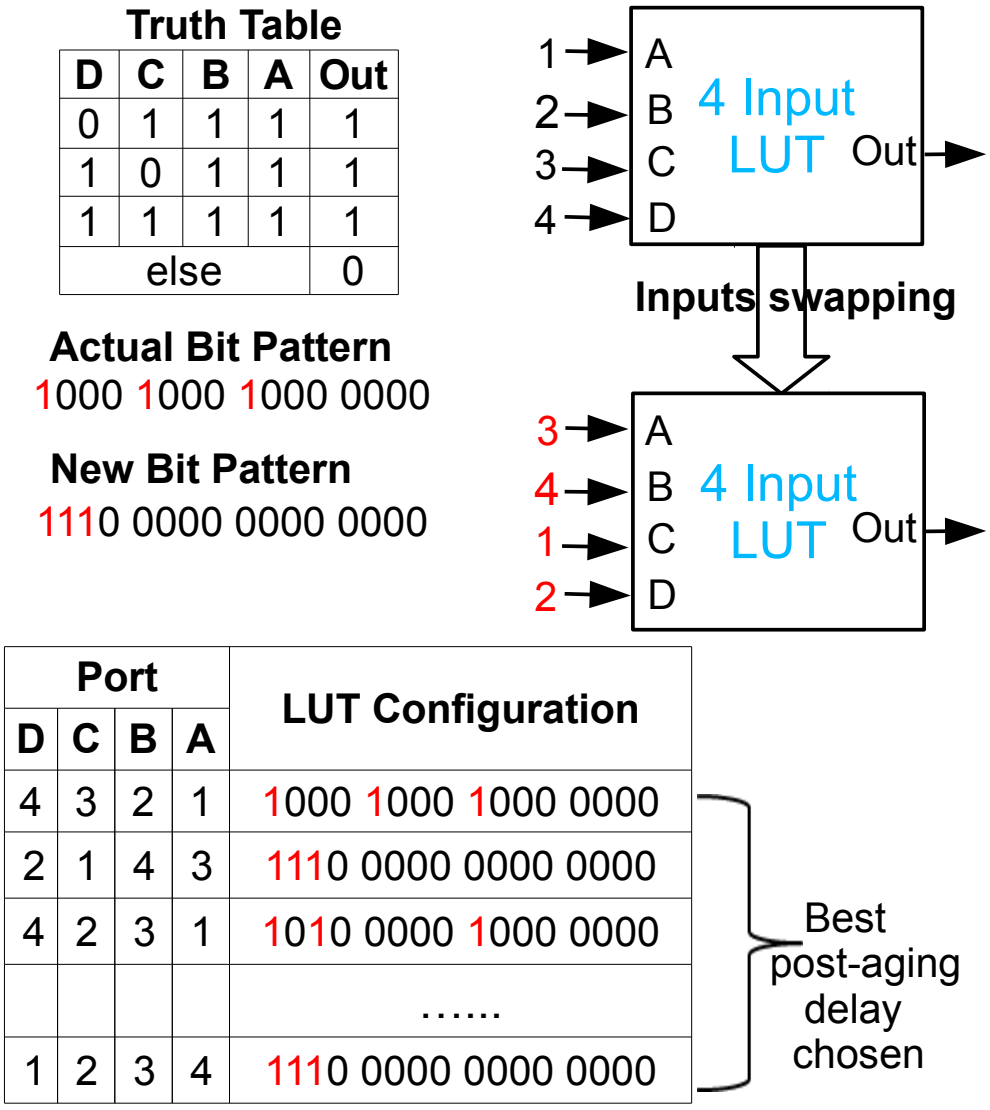


Figure 10.3.: LUT configuration manipulation by swapping LUT inputs

10.4.3. Application strategies

It should be noted that swapping the LUT inputs (Method 2) involves changes to routing, albeit mostly intra-cluster routing. In this chapter, it is presumed that this routing change will be supported by the cluster cross-bar configurations, with almost no impact on routing delay. In short, while employing "Method 2", this chapter presumes that the cost of local routing changes is negligible in architectures which employ full cluster cross-bars, which can easily connect a cluster input to any given LUT port. Nevertheless, keeping the local routing changes as a major constraint, this chapter classifies the application of the proposed two methods into the following strategies:

1. **Routing_intact** ($S_{noroute}$): using "Method 1" alone without any local routing changes, then estimating the benefits obtained with aging mitigation.

2. **Re-routing** ($S_{reroute}$): using the combination of "Method 1" and "Method 2" to in-

crease the search space and come up with alternate LUT patterns, which may potentially yield better aging mitigation, but however requires local routing changes.

10.5. Implementation Flow

In order to implement the proposed methods of Section 10.4, it is necessary to perform data mining on the results obtained in Chapter 3. These results are the delay degradation values for different LUT configurations and input signal probabilities (Table 10.1) for each of the three LUT transistor-level implementations (PT, TG & LG). The data mining is necessary for search space exploration of the proposed methods in order to make the searching for the best aging-aware configuration in each LUT a fast, static and one-time process. For more details about the data mining performed in this regard, the reader is referred to [112]. In addition to that, some modifications to the VTR tool are also necessary to include aging-mitigation as part of the flow, and also to make it aware of configuration-based delay calculation for LUTs.

The original VTR tool takes as inputs, in addition to the HDL circuit, a device architecture file that contains information about the FPGA components and their delays [110]. The device architecture file adopted by this chapter is provided by the VTR tool and is based on a Stratix© IV FPGA. Since only 4-input LUTs are targeted, the device architecture file is modified accordingly to use only this kind of LUTs, and the delays are modified according to the measurements of Chapter 3. Furthermore, to add optimization for aging as a part of the VTR flow and make VTR aware of the proposed strategies in Section 10.4, the following modifications are applied:

1. *Making VTR able to display the complete configuration patterns for each LUT at the final output of the flow*: The VTR tool, by default, is only capable of displaying the truth table for each LUT and does not have the provision to display LUT configuration bit patterns. The VTR tool is therefore modified to include configuration bits as part of each LUT and highlight the effects of configuration changes on individual LUTs by the proposed aging mitigation strategies. For the sake of simplicity, the VTR tool is considered to connect the LUT input signals to the LUT ports in the sequential order (i.e, signal 1 connected to LUT port 1, signal 2 connected to LUT port 2, etc.,) and the configuration bitstream is generated accordingly. Using this order, the data mining tool employs a signal probability of ”0” or ”1” for the unused inputs as discussed in section 10.4.

2. *Picking the post-aging delays for the LUTs according to their configurations from the hashing tables generated by the data mining tool*: Based on each LUT original configuration value, the best post-aging delay and the corresponding new LUT configuration value can be picked up from the respective hashing tables according to the LUTs architecture (PT, TG and LG), aging phenomenon (BTI and NBTI) and the mitigation strategies. Therefore, with this modification, the VTR tool is designed to generate circuit paths with pre-aging[†] and post-aging delays and thus enable the

[†] ‘Pre-aging” delay refers to fresh LUT delay at the start of operation.

comparison between them. This helps to analyze the impact of BTI-induced aging on the circuit paths and also the effects of aging mitigation offered by the two strategies proposed in this chapter.

3. *Adding a timing report utility to report all critical and near critical paths in the circuit based on the post-aging delays*: In this implementation, using the default critical path report generated by the VTR as the baseline, four additional critical path reports are generated by substituting 1) the pre-aging delays, 2) the unmitigated post-aging delays, 3) the post-aging delays based on $S_{noroute}$, and 4) the post-aging delays based on $S_{reroute}$, respectively. This is done for the purpose of comparing the critical paths before and after aging.

With these modifications, we have the ability to choose which aging optimization strategy to apply on all available LUTs in the design, and also to compare the efficiency of different strategies against the original case (with no aging optimization) for the overall circuit delay using an aging aware (i.e. pre-aging and post-aging) static timing analysis.

10.6. Experimental Results

Various standard VTR benchmark circuits are tested using the modified VTR tool (Section 10.5). Three different post-aging delays are captured for the critical path in each circuit: 1) without using any aging mitigation strategy ($D_{Default}$), 2) using $S_{noroute}$ strategy ($D_{noroute}$) and 3) using $S_{reroute}$ strategy ($D_{reroute}$). For each strategy, both NBTI-induced post-aging delay (to account for older devices without high-κ/metal gate materials), and BTI (NBTI + PBTI)-induced post-aging delay (to account for the newer devices) are considered.

The critical path delay of a circuit includes both the LUT delays and the routing delays. Since the proposed strategies target only LUTs, the effect of aging on routing resources is not considered in this work. Nevertheless, based on the results of Chapter 4, aging of routing resources is affected by their input signal probabilities. Since the functionality of LUTs and accordingly the signal probabilities of the internal nodes are not affected by using the proposed strategies, we believe that applying the proposed method will not affect aging of the routing resources.

The experiments on the benchmark circuits are repeated for the three LUT architectures. However, for the sake of clarity and brevity, the detailed results are presented only for TG-based LUTs. For the PT and LG-based LUTs only average values are presented. Table 10.2 reports the TG-based critical-path normalized Δdelay for each benchmark circuit, taking into consideration, aging phenomena and the strategies used.

It should be noted that the calculation of input signal probabilities of used LUT inputs is not possible using VTR, because VTR currently does not have a provision to generate a post-place-and-route simulation model. Therefore, the worst case post-aging delays among all possible signal probabilities of used inputs for each LUT configuration are considered. This is of course very pessimistic; however, it still shows that the proposed strategies can successfully mitigate BTI-induced aging. Whenever the workload details and the post-place-and-route simulation capability are made available for the VTR utility, the precise

Table 10.2.: Worst-case normalized post-aging Δdelay for the critical paths in TG-based LUTs using different optimization strategies.

Circuits	**Size in LUTs**	**LUTs in Critical Path**	**$\Delta D_{Default}$ (%) No mitigation**		**$\Delta D_{noroute}$ (%) $S_{noroute}$ strategy**		**$\Delta D_{reroute}$ (%) $S_{reroute}$ strategy**	
			NBTI	**BTI**	**NBTI**	**BTI**	**NBTI**	**BTI**
ch_intrinsic	646	6	6.13	9.16	5.82	8.91	4.64	7.49
ode	10187	77	5.75	8.94	5.54	8.75	4.64	7.77
fir	16341	72	5.69	8.94	5.46	8.72	4.55	7.64
mm2	11767	74	5.67	8.91	5.44	8.70	4.55	7.64
bfly	17080	74	5.76	8.96	5.54	8.77	4.65	7.73
bgm	44216	66	5.69	8.94	5.51	8.79	4.67	7.85
or1200	4289	67	5.90	8.72	5.24	8.37	3.38	5.90
LU8PEEng	34280	397	6.02	9.01	5.41	8.50	3.90	6.47
s-vision0	16155	11	5.93	9.09	5.67	8.94	4.68	7.73
s-vision1	15262	15	6.39	9.20	5.54	8.47	3.70	5.89
s-vision3	328	8	6.00	9.14	5.86	9.02	4.68	7.97
Average			5.90	9.00	5.55	8.72	4.37	7.28
Average in case of PT-based LUTs			4.80	28.00	4.77	26.95	3.00	21.59
Average in case of LG-based LUTs			4.45	7.92	3.84	7.01	3.13	5.89

signal probabilities can be then used directly and exact post-aging delays can be considered instead of the worst case ones.

As it can be observed from the results in Table 10.2, the strategy $S_{reroute}$ offers better aging mitigation than $S_{noroute}$. This is because $S_{reroute}$ works on a larger search space and also considers both fully- and partially-used LUTs for aging mitigation as discussed in Section 10.4. The improvements that strategy $S_{reroute}$ can offer over $S_{noroute}$ for different benchmark circuits in TG-based architecture is shown in Figure 10.4 and 10.5 for NBTI and BTI-induced aging, respectively. The average BTI-induced aging mitigation using $S_{reroute}$ is about 20%.

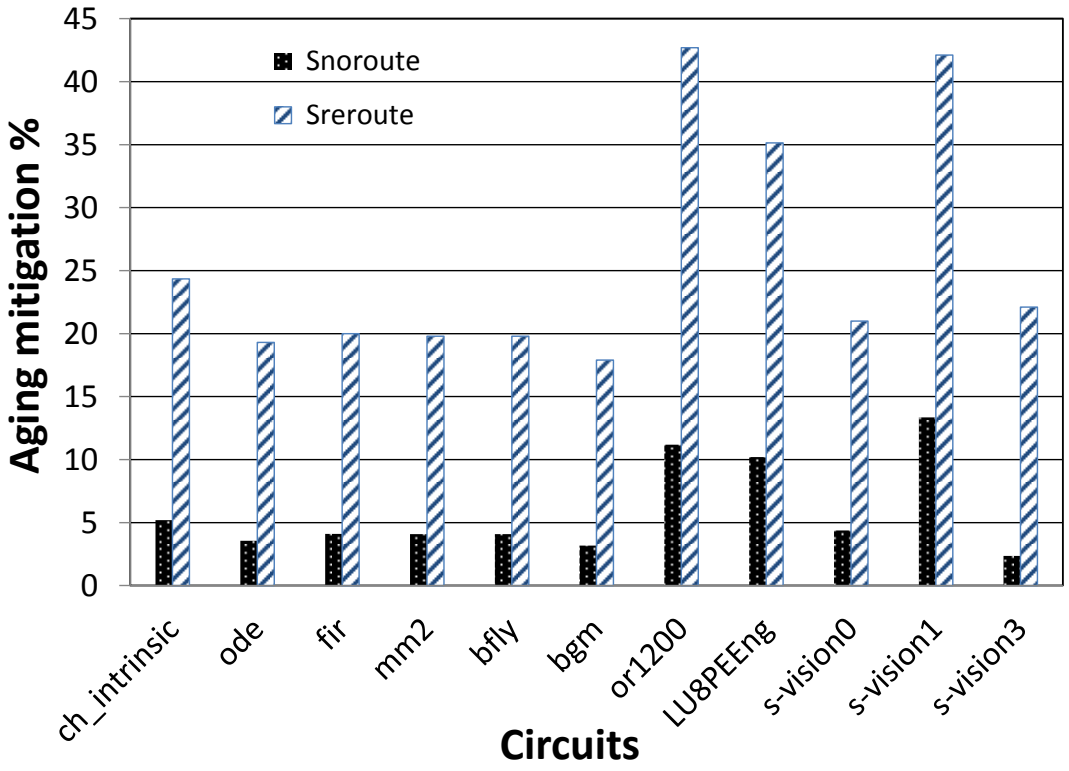


Figure 10.4.: NBTI aging mitigation for TG LUTs - Worst case delay

The degradation caused by BTI-induced aging has an obvious effect on the lifetime of the circuit. Different degradation rates can lead to different operational lifetime of the circuit. Even a small difference in the degradation rates can yield significant lifetime enhancements. Therefore, when the strategies for aging mitigation described in this chapter are translated into lifetime enhancements, a significant improvement to circuit lifetime is achieved. Table 10.3 reports the improvements in lifetime for each of the benchmark circuits using the two strategies. The average lifetime enhancement is about 200% using $S_{reroute}$ for BTI.

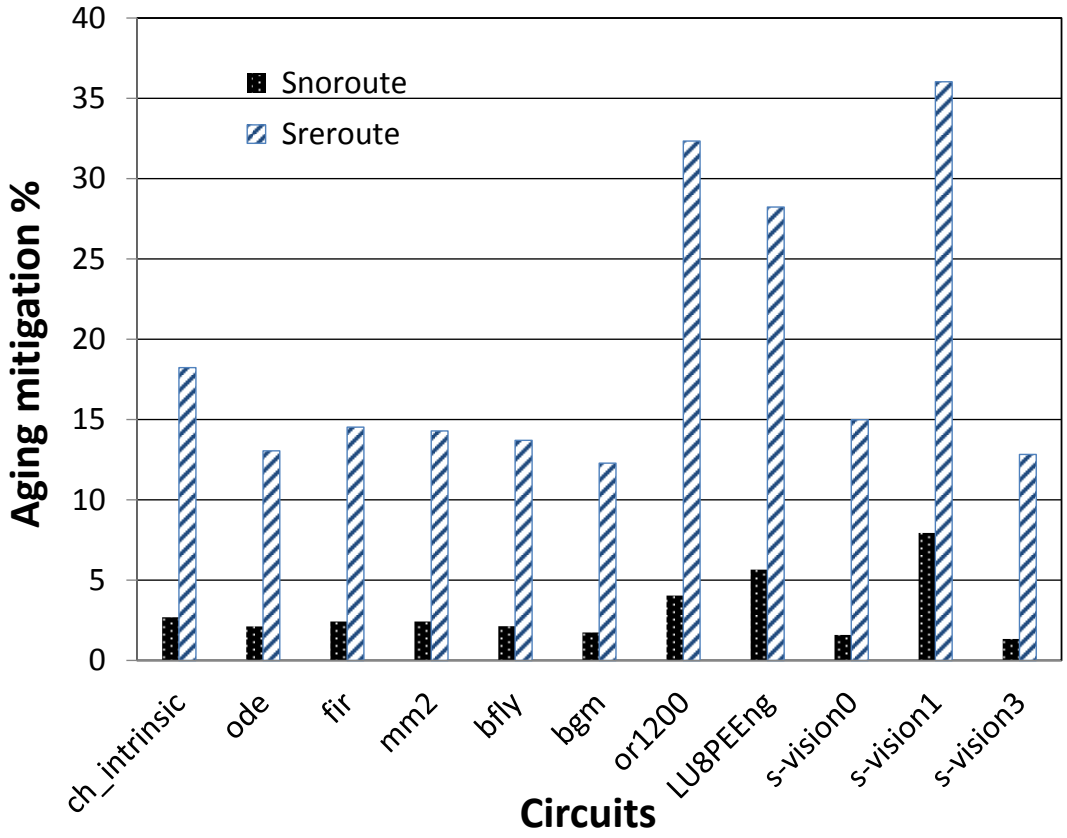


Figure 10.5.: BTI aging mitigation for TG LUTs - Worst case delay

Table 10.3.: Lifetime improvement

Circuits	$S_{noroute}$ (%)		$S_{reroute}$ (%)	
	NBTI	**BTI**	**NBTI**	**BTI**
ch_intrinsic	37.69	17.80	433.01	234.59
ode	24.24	13.72	262.01	131.42
fir	28.66	15.87	281.46	156.48
mm2	28.42	15.88	275.56	152.27
bfly	26.27	13.80	260.05	142.28
bgm	21.39	11.14	226.63	119.52
or1200	103.80	28.15	2722.79	942.89
LU8PEEng	90.73	41.81	1243.28	632.23
s-vision0	30.80	10.07	310.98	165.07
s-vision1	136.17	64.35	2556.22	1360.38
s-vision3	15.36	8.42	347.97	127.89
Average	45.25	20.85	509.23	257.79
Average for PT-based	4.59	25.65	1498.23	278.49
Average for LG-based	142.60	108.01	239.61	184.29

10.7. Summary

In this chapter, two novel strategies to mitigate BTI-induced aging in LUTs have been proposed and implemented. The proposed mitigation strategies have been implemented by manipulating the LUT configuration bits and input signal probabilities, while preserving the intended mapped logic functionality. Implementation is carried out by devising a new data mining tool and also by using the modified version of the VTR tool. The results presented in this chapter confirmed that there is a significant improvement on BTI-induced aging degradation based on the proposed strategies. The main conclusions of this chapter can be summarized as follows.

- Altering LUT configuration and input signal probabilities can significantly mitigate BTI-induced aging in FPGAs.
- The mitigation strategies proposed in this chapter are feasible and can be implemented without significant run-time overheads.
- The proposed strategies improve the lifetime of the FPGA-mapped designs, on average, by more than 200%.

Chapter 11.
Conclusions and Outlook

Transistor aging, as a main cause for the degradation in FPGAs, is one of the most important reliability challenges at nano-scale CMOS. Since FPGAs are used in many domains including very critical applications, it is extremely important to account for transistor aging through all design stages. Thus, appropriate mitigation and countermeasures have to be implemented at all levels of the design from system-level down to transistor-level. In this dissertation, investigation, modeling, monitoring and mitigation techniques are proposed to understand and deal with the aging of FPGAs at different abstraction levels.

11.1. Conclusions

In the first part of this dissertation (*the investigation and modeling part*), the BTI-induced aging effects on the basic building blocks of the FPGA (LUTs and switch matrices) are investigated in the means of simulation using transistor-level models. The results of investigating the aging of LUTs show that the LUT structure, the current configuration, the previously run configuration, and the inputs signal probabilities are parameters that have considerable effect on the NBTI and PBTI induced aging of LUTs, and all of them must be considered for accurate delay degradation analysis. It is also shown that the PT-based structure (the one with least area overhead) is the worst choice for high-κ/metal gates technologies. Furthermore, it is shown that the best configuration for the unused LUTs depends on the runtime configuration and the LUT-structure.

The results of investigating the aging of switch matrices revealed several knobs that can be used by the FPGA manufacturers and designers to mitigate the aging. The first knob is about the number of cascaded switches against the length of the connecting wires, where it is shown that less number of cascaded switches with longer wires have less aging effect than larger number of cascaded switches with shorter wires. The second knob is about the fan-out of the TS-buffer structure, where activating a higher fan-out leads to less aging. The third knob is about the supply voltage, where using higher V_{dd} for both the PT-keeper and TS-buffer structures leads to less aging. It is also shown that the structures based on NMOS transistor are the best for technologies susceptible to only NBTI, while the transmission gate based structure is the best when both NBTI and PBTI are important.

The aforementioned investigations are followed by an experimental analysis for the aging effect in FPGAs from a system-level point-of-view. This analysis reveals as well several important facts about the aging. It shows the extent of the aging-induced degradation in FPGA for the tested technology and the role of usage (input signal probabilities and switching activities) in this degradation. It also shows that there is no correlation between how the process variation affects the initial delay and how it affects the aged one. Furthermore,

it shows that the unused FPGA resources age significantly as well, in some cases more than some used/active blocks.

As temperature has an exponential relation with the aging effects, it was also necessary in this part to accurately estimate the thermal-profile of FPGA-mapped designs to allow for more accurate aging estimation. Therefore, a method for properly distributing the leakage power across the FPGA chip is presented. This method is based on a temperature-leakage loop estimation model for distributing the leakage power. It is shown that the presented approach can achieve accurate thermal-profile estimation, with average absolute estimation error of around 1°C across the FPGA chip.

All of the previous knowledge is then gathered to propose an aging estimation and prediction tool, which can estimate/predict the amount of aging-induced degradation for designs mapped to FPGA devices. The tool is based on RTL abstractions for both BTI and HCI device-level models, and extracts the necessary information from the implicit data existed in the power and timing reports provided from the FPGA's vendor tools. The proposed tool enables exploring the influence of different designs and mapping options on the amount of aging.

In the second part of this dissertation (*the monitoring part*), a low-cost logic-level aging sensor with very low area, performance and power overhead for FPGA-mapped designs is presented. The sensor can be controlled to be warning or late-transitions detector that not only detects transistor aging, but can also detect erroneous glitches due to intermittent and transient faults. Furthermore, a sensor placement methodology is presented to achieve a highly reliable aging monitoring.

In the second part as well, and for quickly test for thermally-activated faults in FPGAs, self-heating integration techniques for thermal-aware testing of FPGA devices is presented. These techniques eliminate the need for external devices for heating up the FPGA. Two different categories of these techniques are presented to fit different testing purposes; for BIST, and application-dependent testing.

In the third part of this dissertation (*the mitigation part*), two novel strategies to mitigate BTI-induced aging in LUTs are proposed. These strategies are based on the relation between the LUT configuration and the amount of BTI-induced degradation. The results of applying these strategies show that a significant improvement on BTI-induced aging degradation can be achieved by just altering the LUT configurations.

11.2. Outlook

As the main focus in this dissertation is on degradation in FPGAs, the degradation effects due to TDDB mechanism can be investigated in the future and modeled. Also, aside from the transistor-aging mechanisms, the degradation of the interconnects (i.e., the interconnect aging) due to EM and SM can be also targeted in any future work.

The high-level aging estimation methodology presented in Chapter 7 can be used in the placement and routing algorithms of FPGAs. In which a third optimization function can be proposed (i.e., in addition to the two existing optimization functions that can optimize for either the speed or the area) to optimize for aging reliability. Thus, aging-aware placement and routing of the FPGA-mapped design can be achieved.

Using the main conclusions of this dissertation, especially those related to the basic building blocks (LUTs and switch matrices), it is also possible to build an aging-aware FPGA chip, in which the FPGA building blocks are designed with aging effects in mind from the transistor-level up to the whole FPGA. Such aging-awareness should be straight forward to implement if the low-level details of the FPGA are available.

List of abbreviations

AR Switching Activity Rate of the transistor

D_{it} or N_{it} Interface Trap

LT Lifetime Extension

$SA(s)$ Switching Activity(ies)

V_D Drain voltage

V_G Gate voltage

V_S Source voltage

V_{DD} Positive supply voltage in Field-Effect Transistor (FET) technology

V_{gs} Gate-Source Voltage

V_{th} Threshold Voltage

AC Alternating Current

ASIC(s) Application Specific Integrated Circuit(s)

BIST Built-In Self-Test

BLE(s) Basic Logic Element(s)

BTI Bias Temperature Instability

CHE Channel Hot Electron

CLB(s) Configurable Logic Block(s)

CMOS Complementary Metal-Oxide-Semiconductor

DAHC Drain Avalanche Hot Carrier

DC Direct Current

DCM(s) Digital Clock Manager(s)

DSP(s) Digital Signal Processor(s)

EM Electro-Migration

FF(s) flip-flop(s)

FIR Finite Impulse Response

FPGA(s) Field Programmable Gate Array(s)

H Hydrogen

HCE Hot Carrier Effect

HCI Hot Carrier Injection

HM Hard Macro

I/O Input/Output

IC(s) Integrated Circuit(s)

IP(s) Intellectual Property(ies)

LFSR Linear Feedback Shift Register

LG Logic Gate

LUT(s) Look-Up Table(s)

MMCM Mixed-Mode Clock Managers

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor

MTTF Mean-Time-To-Failure

MUX Multiplexer

NBTI Negative Bias Temperature Instability

NCS Number of Cascaded Switches

NMOS N-type Metal-Oxide-Semiconductor

ODDR Double-Data Rate Output Register

PBTI Positive Bias Temperature Instability

PCI Peripheral Component Interconnect

PIP(s) Programmable Interconnect Point(s)

PLD(s) Programmable Logic Device(s)

PLL phase-locked loop

PMOS P-type Metal-Oxide-Semiconductor

PT Pass Transistor

PT-keeper Pass Transistor with keeper

PTM Predictive Technology Model

PUF(s) Physical Unclonable Function(s)

PV Process Variation

RAM Random Access Memory

RD Reaction-Diffusion theory

RO(s) Ring Oscillator(s)

SGHE Secondary Generated Hot Electron

SHC(s) Self-Heating Chain(s)

SHE(s) Self-Heating Element(s)

SHEl Substrate Hot Electron

Si Silicon

SiO2 Silicon Dioxide

SM Stress Migration

SP(s) Signal Probability(-ies)

SRAM Static Random Access Memory

TD Charge Trapping/Detrapping theory

TDDB Time-Dependent Dielectric Breakdown

TG Transmission Gate

TP(s) Transition Probability(ies)

TS-buffer Tri-State buffer

TTF Time-To-Failure

VLSI Very Large Scale Integration

WL Wire Length

XDL Xilinx Design Language

Bibliography

[1] Mike Santarini. Xilinx Ships Industrys First 20-nm All Programmable Devices. *Xcell journal*, 86:8–15, 2014.

[2] Nick Mehta. Xilinx UltraScale Architecture for High-Performance, Smarter Systems. *Xilinx White Paper WP434*, December 2013.

[3] Altera. Meeting the Performance and Power Imperative of the Zettabyte Era with Generation 10. *Altera White Paper WP-01200*, June 2013.

[4] Steve Leibson and Nick Mehta. Xilinx UltraScale: The Next-Generation Architecture for Your Next-Generation Architecture. *Xilinx White Paper WP435*, July 2013.

[5] Altera. The Breakthrough Advantage for FPGAs with Tri-Gate Technology. *Altera White Paper WP-01201*, June 2013.

[6] Jean-Marc Yannou. Xilinx's 3d (or 2.5 d) packaging enables the world's highest capacity FPGA device, and one of the most powerful processors on the market. 3D Packaging, November 2011.

[7] Virtex Ultrascale: Delivering ASIC-Class Advantages, http://www.xilinx.com.

[8] Stratix 10 FPGAs and SoCs: Delivering the Unimaginable, http://www.altera.com.

[9] FPGA applications, http://www.xilinx.com/applications/.

[10] Shekhar Borkar. Tackling variability and reliability challenges. *IEEE Design and Test of Computers*, 23:520, 2006.

[11] Joseph W McPherson. Reliability challenges for 45nm and beyond. In *Proceedings of the 43rd annual Design Automation Conference*, pages 176–181. ACM, 2006.

[12] Sang Phill Park, Kunhyuk Kang, and Kaushik Roy. Reliability implications of bias-temperature instability in digital ics. *IEEE Des. Test*, 26(6):8–17, 2009.

[13] Tanya Nigam, Kok-Yong Yiang, and Amit Marathe. Moore's Law: Technology Scaling and Reliability Challenges. *Microelectronics to Nanoelectronics: Materials, Devices & Manufacturability*, page 1, 2012.

[14] Alvin W Strong, Ernest Y Wu, Rolf-Peter Vollertsen, Jordi Sune, Giuseppe La Rosa, Timothy D Sullivan, and Stewart E Rauch III. *Reliability wearout mechanisms in advanced CMOS technologies*, volume 12. John Wiley & Sons, 2009.

[15] Wenping Wang, Shengqi Yang, S. Bhardwaj, S. Vrudhula, F. Liu, and Yu Cao. The Impact of NBTI Effect on Combinational Circuit: Modeling, Simulation, and Analysis. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 18(2):173 –183, feb. 2010.

[16] S. Zafar, YH Kim, V. Narayanan, C. Cabral, V. Paruchuri, B. Doris, J. Stathis, A. Callegari, and M. Chudzik. A comparative study of NBTI and PBTI (charge trapping) in SiO2/HfO2 stacks with FUSI, TiN, Re gates. In *VLSI Technology, Symposium on*, pages 23–25. IEEE, 2006.

[17] A. Tiwari and J. Torrellas. Facelift: Hiding and slowing down aging in multicores. In *Microarchitecture, International Symposium on*, pages 129 –140. IEEE/ACM, 2008.

[18] A. Bravaix, C. Guerin, V. Huard, D. Roy, JM Roux, and E. Vincent. Hot-carrier acceleration factors for low power management in DC-AC stressed 40nm nMOS node at high temperature. In *Reliability Physics Symposium, 2009 IEEE International*, pages 531–548. IEEE, 2009.

[19] T Nigam, A Kerber, and P Peumans. Accurate model for time-dependent dielectric breakdown of high-k metal gate stacks. In *Reliability Physics Symposium, 2009 IEEE International*, pages 523–530. IEEE, 2009.

[20] Ayse Kivilcim Coskun, Tajana Simunic Rosing, and Kenny C Gross. Proactive temperature management in MPSoCs. In *Proceedings of the 13th international symposium on Low power electronics and design*, pages 165–170. ACM, 2008.

[21] Shekhar Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *Micro, IEEE*, 25(6):10–16, 2005.

[22] Transparency Market Research. Field-Programmable Gate Array (FPGA) Market - Global Industry Analysis, Size, Share, Growth, Trends and Forecast, 2013 - 2019. Technical report, RESEARCH AND MARKETS, October 2013.

[23] Vaughn Betz, Jonathan Rose, and Alexander Marquardt. *Architecture and CAD for deep-submicron FPGAs*. Kluwer Academic Publishers, 1999.

[24] Deming Chen, Jason Cong, and Peichen Pan. FPGA design automation: A survey. *Foundations and Trends® in Electronic Design Automation*, 1(3):139–169, 2006.

[25] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula. Predictive modeling of the NBTI effect for reliable design. In *Custom Integrated Circuits Conference, 2006. CICC'06. IEEE*, pages 189–192. IEEE, 2006.

[26] Dieter K Schroder. Negative bias temperature instability: What do we understand? *Microelectronics Reliability*, 47(6):841–852, 2007.

[27] Tibor Grasser, Ben Kaczer, Wolfgang Goes, Hans Reisinger, Thomas Aichinger, Philipp Hehenberger, P-J Wagner, Franz Schanovsky, Jacopo Franco, M Toledano Luque, et al. The paradigm shift in understanding the bias temperature instability: from reaction–diffusion to switching oxide traps. *Electron Devices, IEEE Transactions on*, 58(11):3652–3666, 2011.

[28] S. Mahapatra, N. Goel, S. Desai, S. Gupta, B. Jose, S. Mukhopadhyay, K. Joshi, A. Jain, A.E. Islam, and M.A. Alam. A Comparative Study of Different Physics-Based NBTI Models. *Electron Devices, IEEE Transactions on*, 60(3):901–916, March 2013.

[29] J.J. Kim, R. Rao, S. Mukhopadhyay, and C.T. Chuang. Ring oscillator circuit structures for measurement of isolated NBTI/PBTI effects. In *Integrated Circuit Design and Technology and Tutorial, International Conference on*, pages 163–166. IEEE, 2008.

[30] JH Stathis, M. Wang, and K. Zhao. Reliability of advanced high-k/metal-gate n-FET devices. *Microelectronics Reliability*, 50(9):1199–1202, 2010.

[31] James H Stathis and Sufi Zafar. The negative bias temperature instability in MOS devices: A review. *Microelectronics Reliability*, 46(2):270–286, 2006.

[32] Tibor Grasser and Ben Kaczer. Evidence that two tightly coupled mechanisms are responsible for negative bias temperature instability in oxynitride MOSFETs. *Electron Devices, IEEE Transactions on*, 56(5):1056–1062, 2009.

[33] Muhammad Ashraful Alam and S Mahapatra. A comprehensive model of PMOS NBTI degradation. *Microelectronics Reliability*, 45(1):71–81, 2005.

[34] Muhammad Ashraful Alam, Haldun Kufluoglu, D Varghese, and S Mahapatra. A comprehensive model for PMOS NBTI degradation: Recent progress. *Microelectronics Reliability*, 47(6):853–862, 2007.

[35] Daniele Ielmini, Mariaflavia Manigrasso, Francesco Gattel, and Grazia Valentini. A unified model for permanent and recoverable nbti based on hole trapping and structure relaxation. In *Reliability Physics Symposium, 2009 IEEE International*, pages 26–32. IEEE, 2009.

[36] DS Ang, CJ Gu, ZY Tung, AA Boo, and Y Gao. Evolution of oxide charge trapping under bias temperature stressing. *Microelectronics Reliability*, 54(4):663–681, 2014.

[37] Takayasu Sakurai and A Richard Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *Solid-State Circuits, IEEE Journal of*, 25(2):584–594, 1990.

[38] Hong Luo, Yu Wang, Ku He, Rong Luo, Huazhong Yang, and Yuan Xie. Modeling of PMOS NBTI effect considering temperature variation. In *Quality Electronic*

Design, 2007. ISQED'07. 8th International Symposium on, pages 139–144. IEEE, 2007.

[39] M. Noda, S. Kajihara, Y. Sato, K. Miyase, X. Wen, and Y. Miura. On estimation of NBTI-Induced delay degradation. In *Test Symposium (ETS), 2010 15th IEEE European*, pages 107 –111, May 2010.

[40] Shyh-Chyi Yang, Hao-I Yang, Ching-Te Chuang, and Wei Hwang. Timing control degradation and NBTI/PBTI tolerant design for Write-replica circuit in nanoscale CMOS SRAM. In *VLSI Design, Automation and Test, 2009. VLSI-DAT '09. International Symposium on*, pages 162 –165, april 2009.

[41] Sanjay V. Kumar, Chris H. Kim, and Sachin S. Sapatnekar. Adaptive techniques for overcoming performance degradation due to aging in digital circuits. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, ASP-DAC '09, pages 284–289, Piscataway, NJ, USA, 2009. IEEE Press.

[42] Eiji Takeda, Cary Y-W Yang, and Akemi Miura-Hamada. *Hot-carrier effects in MOS devices*. Academic Press, 1995.

[43] Renesas. *Semiconductor Reliability Handbook*. Renesas Electronics Corporation, 2008.

[44] Paolo Magnone, Felice Crupi, Nicole Wils, Ruchil Jain, Hans Tuinhout, Pietro Andricciola, Gino Giusi, and Claudio Fiegna. Impact of Hot Carriers on nMOSFET Variability in 45-and 65-nm CMOS Technologies. *Electron Devices, IEEE Transactions on*, 58(8):2347–2353, 2011.

[45] E. Takeda and N. Suzuki. An empirical model for device degradation due to hot-carrier injection. *Electron Device Letters, IEEE*, 4(4):111 – 113, April 1983.

[46] Mihir Choudhury, Vikas Chandra, Kartik Mohanram, and Robert Aitken. Analytical model for TDDB-based performance degradation in combinational logic. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, pages 423–428. IEEE, 2010.

[47] A. Elhami Khorasani, M. Griswold, and T.L. Alford. A Fast I-V Screening Measurement for TDDB Assessment of Ultra-Thick Inter-Metal Dielectrics. *Electron Device Letters, IEEE*, 35(1):117–119, Jan 2014.

[48] David Dye. Partial Reconfiguration of Virtex FPGAs in ISE 12. *Xilinx White Paper WP374*, July 2010.

[49] Altera. Increasing Design Functionality with Partial and Dynamic Reconfiguration in 28-nm FPGAs. *Altera White Paper WP-01137*, July 2010.

[50] Rajat Chauhan and Rajesh Kaushik. Utilization of unused IO block for core logic functions, January 2 2007. US Patent 7,157,936 B2.

[51] E. Stott, P. Sedcole, and P. Cheung. Modelling degradation in FPGA lookup tables. In *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, pages 443–446, dec. 2009.

[52] Edward A. Stott, Justin S.J. Wong, Pete Sedcole, and Peter Y.K. Cheung. Degradation in FPGAs: measurement and modelling. In *FPGA '10: Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, pages 229–238, New York, NY, USA, 2010. ACM.

[53] E. Stott, J.S.J. Wong, and P.Y.K. Cheung. Degradation Analysis and Mitigation in FPGAs. In *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, pages 428–433, 2010.

[54] Charles Chiasson and Vaughn Betz. Should FPGAS abandon the pass-gate? In *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, pages 1–8. IEEE, 2013.

[55] Xinfei Guo, Wayne Burleson, and Mircea Stan. Modeling and Experimental Demonstration of Accelerated Self-Healing Techniques. In *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference*, pages 1–6. ACM, 2014.

[56] Tao Pi and Patrick J Crotty. FPGA lookup table with transmission gate structure for reliable low-voltage operation, December 23 2003. US Patent 6,667,635.

[57] Steven P Young. Programmable logic block having lookup table with partial output signal driving carry multiplexer, March 20 2007. US Patent 7,193,433.

[58] Predictive Technology Model (PTM). `http://ptm.asu.edu/`. model released in Oct. 2007.

[59] Xilinx. 7 Series FPGAs Configurable Logic Block. *Xilinx User Guide 474*, August 2014.

[60] Xilinx. Spartan-6 FPGA Configurable Logic Block. *Xilinx User Guide 384*, February 2010.

[61] I. Kuon, R. Tessier, and J. Rose. FPGA architecture: Survey and Challenges. *Foundations and Trends® in Electronic Design Automation*, 2(2):135–253, 2008.

[62] G. Lemieux and D. Lewis. Circuit design of routing switches. In *international symposium on Field-programmable gate arrays*, pages 19–28. ACM, 2002.

[63] M. Khellah, S. Brown, and Z. Vranesic. Modelling routing delays in SRAM-based FPGAs. In *Canadian Conference on VLSI*, page 6B. Citeseer, 1993.

[64] Florent Bruguier, Pascal Benoit, Philippe Maurine, and Lionel Torres. A New Process Characterization Method for FPGAs Based on Electromagnetic Analysis. In *FPL11 21st International Conference on Field Programmable Logic and Applications*, pages 20–23. Ieee, September 2011.

[65] A. Maiti, L. McDougall, and P. Schaumont. The Impact of Aging on an FPGA-Based Physical Unclonable Function. In *International Conference on Field Programmable Logic and Applications (FPL)*, pages 151 – 156, 2011.

[66] Spartan-6 Family Overview, 2011.

[67] Florent Bruguier, Pascal Benoit, and Lionel Torres. Investigation of Digital Sensors for Variability Characterization on FPGAs. In *Proceedings of the 5th International Workshop on Reconfigurable Communication-centric Systems on Chip 2010 ReCoSoC10*, pages 95 – 100, Karlsruhe, Germany, 2010.

[68] Tim Tuan and Bocheng Lai. Leakage power analysis of a 90nm FPGA. In *Custom Integrated Circuits Conference, 2003. Proceedings of the IEEE 2003*, pages 57–60. IEEE, 2003.

[69] P. Mangalagiri, S. Bae, R. Krishnan, Y. Xie, and V. Narayanan. Thermal-aware reliability analysis for platform FPGAs. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 722–727. IEEE Press, 2008.

[70] Jameel Hussein, Matt Klein, and Michael Hart. Lowering Power at 28 nm with Xilinx 7 Series FPGAs. *Xilinx White Paper WP389*, Aug. 2013.

[71] S. Velusamy, Wei Huang, J. Lach, M. Stan, and K. Skadron. Monitoring temperature in FPGA based SoCs. In *Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, pages 634 – 637, oct. 2005.

[72] Wei Huang, K. Sankaranarayanan, K. Skadron, R.J. Ribando, and M.R. Stan. Accurate, Pre-RTL Temperature-Aware Design Using a Parameterized, Geometric Thermal Model. *Computers, IEEE Transactions on*, 57(9):1277 –1288, sept. 2008.

[73] Y. Yang, Z. Gu, C. Zhu, R.P. Dick, and L. Shang. Isac: Integrated space-and-time-adaptive chip-package thermal analysis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(1):86–99, 2007.

[74] Priya Sundararajan, Aman Gayasen, N. Vijaykrishnan, and T. Tuan. Thermal characterization and optimization in platform FPGAs. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, ICCAD '06, pages 443–447, 2006.

[75] H.Y. Lui, C.H. Lee, and R.H. Patel. Power estimation and thermal budgeting methodology for FPGAs. In *Custom Integrated Circuits Conference, 2004. Proceedings of the IEEE 2004*, pages 711–714. IEEE, 2004.

[76] M. Happe, A. Agne, and C. Plessl. Measuring and predicting temperature distributions on FPGAs at run-time. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, pages 55 –60, dec. 2011.

[77] 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), the 21st IEEE International Symposium on*, pages 57–60, 2013.

[78] M. DeBole, K. Ramakrishnan, V. Balakrishnan, W. Wang, H. Luo, Y. Wang, Y. Xie, Y. Cao, and N. Vijaykrishnan. A framework for estimating NBTI degradation of microarchitectural components. In *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*, pages 455–460. IEEE, 2009.

[79] CAD group. CAD Group: ITC'99 Benchmarks (2nd release), 2010.

[80] Kenneth M. Zick and John P. Hayes. On-line sensing for healthier FPGA systems. In *FPGA '10: Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, pages 239–248, New York, NY, USA, 2010. ACM.

[81] Prasanth Mangalagiri, Sungmin Bae, Ramakrishnan Krishnan, Yuan Xie, and Vijaykrishnan Narayanan. Thermal-aware reliability analysis for platform fpgas. In *ICCAD '08: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 722–727, Piscataway, NJ, USA, 2008. IEEE Press.

[82] Miron Abramovici and Charles E. Stroud. BIST-Based Delay-Fault Testing in FPGAs. *J. Electron. Test.*, 19(5):549–558, 2003.

[83] J.S.J. Wong and P.Y.K. Cheung. Improved delay measurement method in FPGA based on transition probability. In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, pages 163–172. ACM, 2011.

[84] Suresh Srinivasan, Prasanth Mangalagiri, Yuan Xie, N. Vijaykrishnan, and Karthik Sarpatwari. FLAW: FPGA lifetime awareness. In *DAC '06: Proceedings of the 43rd annual Design Automation Conference*, pages 630–635, New York, NY, USA, 2006. ACM.

[85] John Keane, Tae hyoung Kim, Xiaofei Wang, and Chris H. Kim. On-chip reliability monitors for measuring circuit degradation. *Microelectronics Reliability*, In Press, Corrected Proof:–, 2010.

[86] Martin Oma na, Daniele Rossi, Nicolò Bosio, and Cecilia Metra. Novel low-cost aging sensor. In *CF '10: Proceedings of the 7th ACM international conference on Computing frontiers*, pages 93–94, New York, NY, USA, 2010. ACM.

[87] D. Ernst, Nam Sung Kim, S. Das, S. Pant, R. Rao, Toan Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: a low-power pipeline based on circuit-level timing speculation. In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pages 7 – 18, dec. 2003.

[88] Toshinori Sato and Yuji Kunitake. A Simple Flip-Flop Circuit for Typical-Case Designs for DFM. In *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on*, pages 539 –544, mar. 2007.

[89] M. Eireiner, S. Henzler, G. Georgakos, J. Berthold, and D. Schmitt-Landsiedel. In-Situ Delay Characterization and Local Supply Voltage Adjustment for Compensation of Local Parametric Variations. *Solid-State Circuits, IEEE Journal of*, 42(7):1583 –1592, jul. 2007.

[90] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D.M. Bull, and D.T. Blaauw. RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance. *Solid-State Circuits, IEEE Journal of*, 44(1):32 –48, jan. 2009.

[91] K.A. Bowman, J.W. Tschanz, Nam Sung Kim, J.C. Lee, C.B. Wilkerson, S.-L.L. Lu, T. Karnik, and V.K. De. Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance. *Solid-State Circuits, IEEE Journal of*, 44(1):49 –63, jan. 2009.

[92] Xilinx Synthesis and Simulation Design Guide, http://www.xilinx.com.

[93] J. Angermeier, A. Amouri, and J. Teich. General methodology for mapping iterative approximation algorithms to adaptive dynamically partially reconfigurable systems. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pages 302 –307, 31 2009-sept. 2 2009.

[94] Opencores, http://opencores.org.

[95] Chunhua Yao, Kewal K Saluja, and Parameswaran Ramanathan. Thermal-aware test scheduling using on-chip temperature sensors. In *VLSI Design (VLSI Design), 2011 24th International Conference on*, pages 376–381. IEEE, 2011.

[96] Chunsheng Liu, Kugesh Veeraraghavan, and Vikram Iyengar. Thermal-aware test scheduling and hot spot temperature minimization for core-based systems. In *Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on*, pages 552–560. IEEE, 2005.

[97] Zhiyuan He, Zebo Peng, Petru Eles, Paul Rosinger, and Bashir M Al-Hashimi. Thermal-aware SoC test scheduling with test set partitioning and interleaving. *Journal of electronic testing*, 24(1-3):247–257, 2008.

[98] Justin SJ Wong, Pete Sedcole, and Peter YK Cheung. Self-measurement of combinatorial circuit delays in FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2(2):10, 2009.

[99] Sergio Lopez-Buedo, Javier Garrido, and Eduardo I Boemo. Dynamically inserting, operating, and eliminating thermal sensors of FPGA-based systems. *Components and Packaging Technologies, IEEE Transactions on*, 25(4):561–566, 2002.

[100] Markus Happe, Andreas Agne, and Christian Plessl. Measuring and predicting temperature distributions on FPGAs at run-time. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, pages 55–60. IEEE, 2011.

[101] Adwait Gupte and Phillip Jones. Hotspot mitigation using dynamic partial reconfiguration for improved performance. In *Reconfigurable Computing and FPGAs, 2009. ReConFig'09. International Conference on*, pages 89–94. IEEE, 2009.

[102] Piotr Weber, Maciej Zagrabski, Przemyslaw Musz, Krzysztof Kepa, Maciej Nikodem, and Bartosz Wojciechowski. Configurable heat generators for FPGAs. In *Thermal Investigations of ICs and Systems (THERMINIC), 2014 20th International Workshop on*, pages 1–4. IEEE, 2014.

[103] Andreas Agne, Hendrik Hangmann, Markus Happe, Marco Platzner, and Christian Plessl. Seven recipes for setting your fpga on fire–a cookbook on heat generators. *Microprocessors and Microsystems*, 38(8):911–919, 2014.

[104] Siuki Chan and Steven HC Hsieh. Methods and apparatus for isolating critical paths on an IC device having a thermal energy generator, May 2005. US Patent 6,895,566.

[105] Steven HC Hsieh and Siuki Chan. Methods and circuits for measuring the thermal resistance of a packaged IC, Aug 2007. US Patent 7,257,511.

[106] Robert O Conn, Steven J Carey, Siuki Chan, and William H Pabst. Self-heating mechanism for duplicating microbump failure conditions in FPGAs and for logging failures, Apr 2008. US Patent 7,362,121.

[107] Bradley F Dutton and Charles E Stroud. Built-in self-test of configurable logic blocks in Virtex-5 FPGAs. In *System Theory, 2009. SSST 2009. 41st Southeastern Symposium on*, pages 230–234. IEEE, 2009.

[108] Xilinx. Virtex-5 FPGA User Guide. *Xilinx Documentation UG190*, March 2012.

[109] Doug Amos, Austin Lesea, and René Richter. *FPGA-Based Prototyping Methodology Manual*. Happy About, 2011.

[110] Jonathan Rose, Jason Luu, Chi Wai Yu, Opal Densmore, Jeff Goeders, Andrew Somerville, Kenneth B. Kent, Peter Jamieson, and Jason Anderson. The VTR Project: Architecture and CAD for FPGAs from Verilog to Routing. In *Proceedings of the 20th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 77–86. ACM, 2012.

[111] S. Srinivasan, R. Krishnan, P. Mangalagiri, Y. Xie, V. Narayanan, M.J. Irwin, and K. Sarpatwari. Toward increasing FPGA lifetime. *Dependable and Secure Computing, IEEE Transactions on*, 5(2):115–127, 2008.

[112] P.M.B. Rao, A. Amouri, S. Kiamehr, and M. Tahoori. Altering LUT Configuration for Wear-out Mitigation of FPGA-Mapped Designs. In *Field Programmable Logic and Applications (FPL), 2013 23nd International Conference on*, pages 1–8, 2013.