CRANFIELD UNIVERSITY

SOHAIB ASLAM

DESIGN FOR PROGNOSTICS AND SECURITY IN FIELD PROGRAMMABLE GATE ARRAYS (FPGAs)

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING

Doctor of Philosophy
Academic Year: 2017 - 2020

Supervisor: Prof. Ian K Jennions
Associate Supervisor: Dr. Mohammad Samie
March 2020

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING

Doctor of Philosophy

Academic Year 2017 - 2020

SOHAIB ASLAM

DESIGN FOR PROGNOSTICS AND SECURITY IN FIELD PROGRAMMABLE GATE ARRAYS (FPGAs)

Supervisor: Prof. Ian K Jennions
Associate Supervisor: Dr. Mohammad Samie
March 2020

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

ABSTRACT

There is an evolutionary progression of Field Programmable Gate Arrays (FPGAs) toward more complex and high power density architectures such as Systems-on-Chip (SoC) and Adaptive Compute Acceleration Platforms (ACAP). Primarily, this is attributable to the continual transistor miniaturisation and more innovative and efficient IC manufacturing processes. Concurrently, degradation mechanism of Bias Temperature Instability (BTI) has become more pronounced with respect to its ageing impact. It could weaken the reliability of VLSI devices, FPGAs in particular due to their run-time reconfigurability. At the same time, vulnerability of FPGAs to device-level attacks in the increasing cyber and hardware threat environment is also quadrupling as the susceptible reliability realm opens door for the rogue elements to intervene. Insertion of highly stealthy and malicious circuitry, called hardware Trojans, in FPGAs is one of such malicious interventions. On the one hand where such attacks/interventions adversely affect the security ambit of these devices, they also undermine their reliability substantially. Hitherto, the security and reliability are treated as two separate entities impacting the FPGA health. This has resulted in fragmented solutions that do not reflect the true state of the FPGA operational and functional readiness, thereby making them even more prone to hardware attacks. The recent episodes of Spectre and Meltdown vulnerabilities are some of the key examples. This research addresses these concerns by adopting an integrated approach and investigating the FPGA security and reliability as two inter-dependent entities with an additional dimension of health estimation/ prognostics. The design and implementation of a small footprint frequency and threshold voltage-shift detection sensor, a novel hardware Trojan, and an online transistor dynamic scaling circuitry present a viable FPGA security scheme that helps build a strong microarchitectural level defence against unscrupulous hardware attacks. Augmented with an efficient Kernel-based learning technique for FPGA health estimation/prognostics, the optimal integrated solution proves to be more dependable and trustworthy than the prevalent disjointed approach.

Keywords:

Reliability, Hardware Trojans, Kernel Learning, Negative and Positive Bias Temperature Instability (N/PBTI), Cybersecurity, Threshold Voltage.

ACKNOWLEDGEMENTS

I dedicate this valuable piece of intensive research and intellect to my father, who has always been the prime source of motivation and inspiration for me towards undertaking this arduous journey of immense worth. His unconditional love and support throughout my life have kept me sailing smoothly during moments of distress and disbelief.

Dr. Mohammad Samie, my associate supervisor, has been an influential part of my academic development, and I owe him my special gratitude for persevering my sluggishness all these years of research. He has been much more than an academic advisor to me – a good friend, fantastic human being, and a lively caring person to share thoughts with. I thank him for his guidance and selfless help during this time.

Prof. Ian K Jennions, my supervisor, has always been a source of inspiration for me. His wisdom and supportive nature are second to none. I thank him for his consistent encouragement.

In particular, I would like to specially thank my viva panel comprising, Dr. Lucian Prodan, Dr. Hongmei He, and Dr. Ioannis Giannopoulos for examining my research work in a highly professional manner and rendering extremely useful guidance on enhancing its impact.

Friendship with Alex, my Ph.D. mate, meant a lot to me, as I could always rely on him for help and good company. Lastly, the love and encouragement from my wife and children got me all the strength to carry on; thank you all.

Sohaib, Cranfield, UK, March 2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

AC	Alternating Current
ACAP	Adaptive Compute Acceleration Platform
ASIC	Application Specific Integrated Circuit
BTI	Bias Temperature Instability
CLB	Configuration Logic Block
CPS	Cyber Physical Systems
CPU	Central Processing Unit
DoE	Design of Experiment
EM	Electromigration
FIT	Failure In Time
FPGA	Field Programmable Gate Array
FFT	Fast Fourier Transform
HCI	Hot Carrier Injection
HT	Hardware Trojan
IFHM	Integrated FPGA Health Management
IC	Integrated Circuit
IOB	Input Output Block
LUT	Look Up Table
NBTI	Negative Bias Temperature Instability
NFET	Negative Field Effect Transistor
NMOS	Negative Metal Oxide Semiconductor
NoC	Network on Chip
PBTI	Positive Bias Temperature Instability
PFET	Positive Field Effect Transistor
PMOS	Positive Metal Oxide Semiconductor
PV	Process Variation
PHM	Prognostics and Health Management
RUL	Remaining Useful Life
SRAM	Static Random Access Memory
TDDB	Time Delay Dielectric Breakdown
VLSI	Very Large Scale Integration

1 INTRODUCTION

1.1 Motivation

The evolutionary progression of ‘Field Programmable Gate Arrays’ (FPGAs), from the traditional architectures to more complex and functionally capable heterogeneous platforms like ‘System-on-Chip (SoC)’ and ‘Adaptive Compute Acceleration Platform (ACAP)’ has been phenomenal. This reconfigurable class of integrated circuits is enabling real-time AI inference and adaptive compute acceleration in numerous sensitive applications across wide-ranging industrial sectors [1]. Recent statistical surveys envisage the global FPGA market to reach USD 14.2 billion by 2024 [2]. Against this backdrop, the functional as well as the operational reliability of these miniaturized nano-systems becomes highly relevant and worth probing. More importantly, with the episodes of Meltdown and Spectre casting shadows on the security fabric of FPGAs [3,4] and the vulnerability of its supply chain to hardware threats like hardware Trojans; the safe operation, confidentiality of the sensitive data and reliable performance of FPGAs may be jeopardised. For instance, when deployed in aerospace and defence systems operating under harsh environmental conditions for prolonged duration, the continual health assessment of FPGAs coupled with security is highly desired.

1.1.1 Research Gaps

1.1.1.1 Lack of Integrated Approach and Framework for FPGA Health Management

The systematic literature review has revealed that the researchers and academicians have been treating the vital elements of security and reliability in FPGAs as two separate entities [5], [6], [7], [8], [9], [10]. Such an approach is fragmented in nature and does not provide complete health assessment of an FPGA device, which is essential for the optimal functioning and operation of existing and future industrial, health-care, aerospace, and energy systems.

Instead, this has led to a clutter of non-composite solutions which do not provide an effective methodology and framework to build trust and ensure reliability as well as health management in FPGAs. Hence, to meet this challenge, there is a requirement of not only an in-depth study and pragmatic research on these two vital domains in a composite manner but to formulate a high-level integrated FPGA health management framework.

1.1.1.2 Lack of Design for Prognostics and Security in FPGAs

The existing 'Designs for Testability and Manufacturability' are not optimized to assess the remaining useful life (RUL) of an FPGA at a nano-system level. These designs are focused more on performing testability analysis of FPGAs (and other VLSI devices) using different scan design methods such as scan-based logic built-in self-test (BIST) [11] and JTAG boundary scan, which are themselves vulnerable to hardware attacks [12], [13]. They are not designed and optimised to provide the prognostics and security assessments through a controllable built-in mechanism.

Especially, in the event of a hardware Trojan attack that may accelerate the ageing process (with subsequent delay degradation) in an FPGA (triggered by Negative Bias Temperature Instability - NBTI) [14] , its health estimation that encompasses both the prognostics and security elements becomes critically essential. It is, therefore, deemed essential to bridge this gap and build 'Design for Prognostics and Security' that augments the controllability and observability regime in FPGAs for a highly reliable and security-hardened performance.

1.1.1.3 Frequency/Delay Degradation Measurement Sensors are Resource-intensive

Intra-die/process variation, which causes performance inconsistency across different process technologies of FPGAs (90nm to 7nm), is a greater challenge. In particular, the literature confirms that the degradation mechanisms of Bias Temperature Instability (BTI) (consisting of Negative and Positive components - NBTI and PBTI), Hot Carrier Injection (HCI), Electromigration (EM) and Time Delay Dielectric Breakdown (TDDB) continue to pose reliability, ageing and performance issues with shrinking process technologies (28nm and below). This can be exploited by a rogue element to design and implement hardware Trojan to de-functionalise FPGA and affect the systems' performance [14]. One of the key performance parameters that is highly sensitive to process variation and is the direct consequence of N/PBTI mechanisms is the propagation delay, which is the function of frequency degradation, threshold voltage shift, and reduction in the drain current – the device ageing parameters. In other words, delays are the function of transistors' ageing. Therefore, precise measurement of these critical transistor parameters propagation delays of arbitrary signal paths through logic blocks, interconnects and heterogeneous elements on an FPGA, is critical for the accurate prediction of its performance.

The evidence from literature reveals that the existing propagation delay measurement sensors and methods are largely based on synchronous (using system clocks) designs and suffer from performance issues, namely clock jitter and flip flop metastability [15]. They consume more FPGA resources and are prone to reliability issues [16]. Therefore, building asynchronous sensor designs and characterising them to utilise in-field timing slack and transition probability measurement methods for determining propagation delay, can help improve the detection of hardware Trojans embedded in FPGA and provide vital parametric data for its health estimation.

1.2 Research Aim and Objectives

1.2.1 Aim

In order to address the above-mentioned research gaps, we defined our overall scientific aim as follows:

‘Design for Prognostics and Security in Field Programmable Gate Arrays (FPGAs) that facilitates their reliability and security enhancement, enables NBTI-based hardware Trojan detection and mitigation within their reconfigurable fabric and helps estimate their health’.

1.2.2 Objectives

Based on the above, we defined the following key objectives to achieve the aim:

1.2.2.1 Integrated FPGA Health Management (IFHM) Framework

Devise a high-level integrated framework for FPGA health management that provides guidance to the researcher, an FPGA manufacturer, and an expert end-user on the process flows and methods required to develop a ‘Design for Prognostics and Security in FPGAs’ in a composite manner.

1.2.2.2 Frequency/Delay Degradation Measurement Sensor

Design and implement a small footprint on-chip sensor (in the target 28 nm FPGA technology node) with low area and power overheads and high sensitivity to detect frequency degradation and delay variations. Characterise the sensor under nominal and stressed Temperature and Voltage conditions for the variations in frequency, delay, and threshold voltage, followed by collation of data to prognosticate FPGA health.

1.2.2.3 FPGA Security Scheme

Design and implementation of an FPGA Security Scheme capable of detecting a hardware Trojan and providing effective mitigation. It comprises investigation and measurement of the degradation/ageing impact of Negative Bias Temperature Instability (NBTI) on the target FPGA (***resulting in threshold voltage shifts***) by conducting highly accelerated stress tests in a controlled environment. Based on the acquired results and collated data, design and implement Threshold Voltage Shift-triggered HT inside the target FPGA for the payload and detection analysis *(observing threshold voltage shifts, and corresponding frequency degradation and delays),* followed by detailed data analytics to build HT-infected FPGA profile. In addition, the hardware Trojan mitigation scheme would form an integral part of this scheme.

1.2.2.4 FPGA Health Estimation

Develop FPGA health estimation/prognostics method using Kernel-based machine learning technique. The validated data from the healthy and HT-infected FPGA profiling experiments will be used to evaluate the method – resulting in the culmination of the ‘Design for Prognostics and Security’.

1.3 Research Methodology

The overall research methodology is constructed around the research objectives. The quantitative research methodology has been adopted and is augmented with the ‘Design of Experiment’ (DoE) approach for the experimentation phases, followed by the statistical analysis of the validated data. Quality assurance of the project is achieved by ensuring the correctness of the processes developed to conduct design and implementation tests, the collation of experimental data and ‘a posteriori’ analysis.

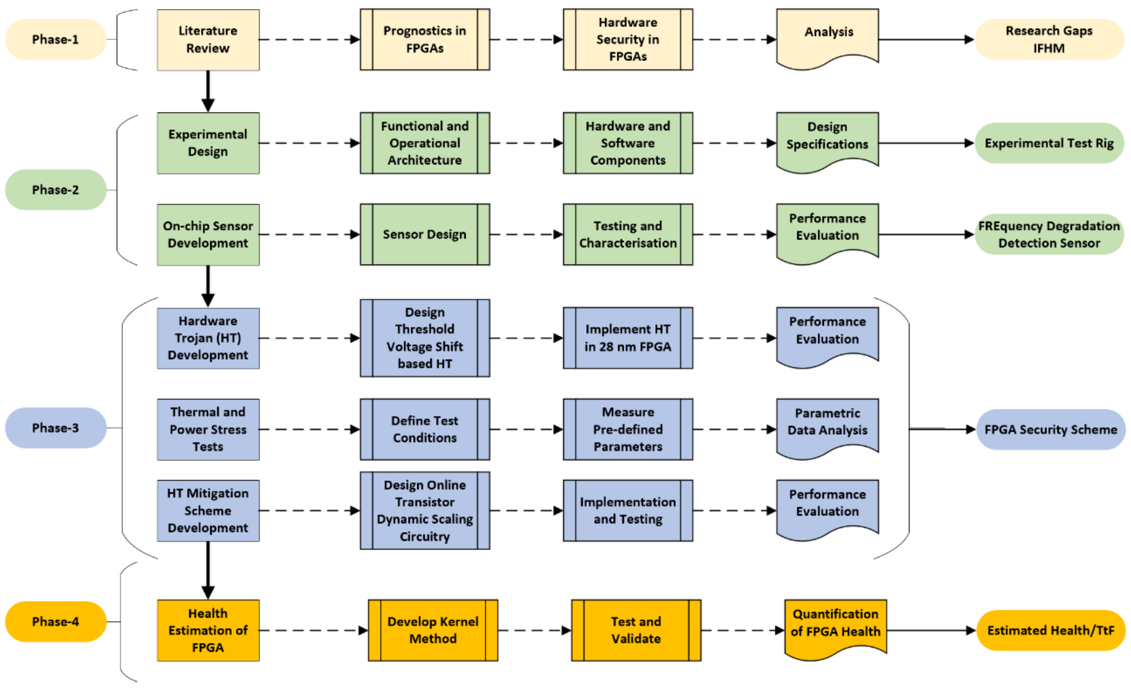
The details of research methodology are appended below. In addition, the research methodology process flow is given in Figure 1-1 for a quick overview.

1.3.1 Phase-1

This phase marked the beginning of the research process with the building up of the context of research on ‘Reliability & Prognostics in FPGAs’ and ‘Hardware Security in FPGAs’ through literature review. It was aimed at understanding the existing research made to date in the aforementioned areas and excavating gaps that can be worked upon for an intelligible contribution and improvement in the field of FPGA prognostics and health management coupled with security. The main deliverables of this phase were the **‘Research Gaps’** and the high-level ‘**Integrated FPGA Health Management (IFHM)**’ framework.

1.3.2 Phase-2

The research gaps from phase-1 were analysed to develop the requirements for designing and outlining requisite experiments. This phase consisted of two sub-phases. In the first sub-phase, the functional and operational architectures for the series of experiments, required to develop ‘Design for Prognostics and Security in FPGAs’, were defined. It included the software and hardware components, their interfacing and optimization to ensure the development of an efficient and effective experimental set-up and test rig. The second sub-phase related to the design and



Phase-1
Literature Review
Prognostics in FPGAs
Hardware Security in FPGAs
Analysis
Research Gaps IFHM
Phase-2
Experimental Design
Functional and Operational Architecture
Hardware and Software Components
Design Specifications
Experimental Test Rig
On-chip Sensor Development
Sensor Design
Testing and Characterisation
Performance Evaluation
FREquency Degradation Detection Sensor
Phase-3
Hardware Trojan (HT) Development
Design Threshold Voltage Shift based HT
Implement HT in 28 nm FPGA
Performance Evaluation
Thermal and Power Stress Tests
Define Test Conditions
Measure Pre-defined Parameters
Parametric Data Analysis
FPGA Security Scheme
HT Mitigation Scheme Development
Design Online Transistor Dynamic Scaling Circuitry
Implementation and Testing
Performance Evaluation
Phase-4
Health Estimation of FPGA
Develop Kernel Method
Test and Validate
Quantification of FPGA Health
Estimated Health/TtF

Figure 1-1 Research Methodology Process Flow Diagram.

implementation of a digital sensor inside the target FPGA, followed by its simulated and real-time testing and characterisation. The main deliverable of this phase was '**FREquency Degradation Detection and Measurement Sensor – FRED**'.

1.3.3 Phase-3

This phase was aimed at designing and implementing FPGA Security Scheme. It consisted of the design and implementation of the Threshold Voltage Triggered hardware Trojan in the target FPGA, improving the FRED sensor design for accuracy and to enable detection of shifts in threshold voltage due to NBTI mechanism, building HT-mitigation sub-scheme, and eventually subjecting it to 'Thermal and Power Cycling' under pre-defined stress test conditions. The main deliverable of this phase was the **'FPGA Security Scheme'**.

1.3.4 Phase-4

Kernel-based Machine Learning method was studied and evaluated for prognosticating FPGA health in this phase. Accordingly, the method was developed and validated against FPGA fault dictionary. The main deliverable of this phase was the '**FPGA Health Estimation/Prognostics**'.

1.4 Organisation of Thesis

This thesis is organised into seven chapters in the '**Paper-Format**' and not as a 'Monograph Format' thesis. Accordingly, the papers have been reformatted into chapters with minor changes to maintain coherence and ensure format consistency. The thesis disposition is shown in Figure 1-2. A brief overview of the remaining chapters is given as follows:

1.4.1 Chapter 2

This chapter is the outcome of '**Objective 1**' and it delineates the integrated approach to prognostics and security in FPGAs by putting forth an '**Integrated FPGA Health Management (IFHM)**' framework. The architecture of a modern FPGA is explained inter alia its commercial and industrial significance. The chapter draws the canvas of the extant and the future technological revolution with FPGAs at the heart of it. Most significantly, it excavates the FPGA reliability issues, vulnerabilities, threats, and several counteractive research efforts made in the realms of FPGA reliability, prognostics, and security to enhance its dependability and build trust. A high-level framework, called **IFHM** is finally presented that provides guidance on managing prognostics and security in FPGAs as a composite entity for their security and reliability hardening.

This chapter is the reformatted version of the paper under submission to '**IEEE Transactions on Device and Materials Reliability**'.

1.4.2 Chapter 3

This chapter highlights the integrated circuit (IC) level threats with an emphasis on hardware Trojans that pose a significant threat to computational systems employing FPGAs, Systems-on-Chip (SoC) or Network-on-Chip (NoC). It describes the hardware Trojan phenomenon, its taxonomy, and gives a critical analysis of various hardware Trojan countermeasures.

This chapter is the reformatted version of the conference paper published in '**Advances in Manufacturing Technology XXXII – 2018**'.

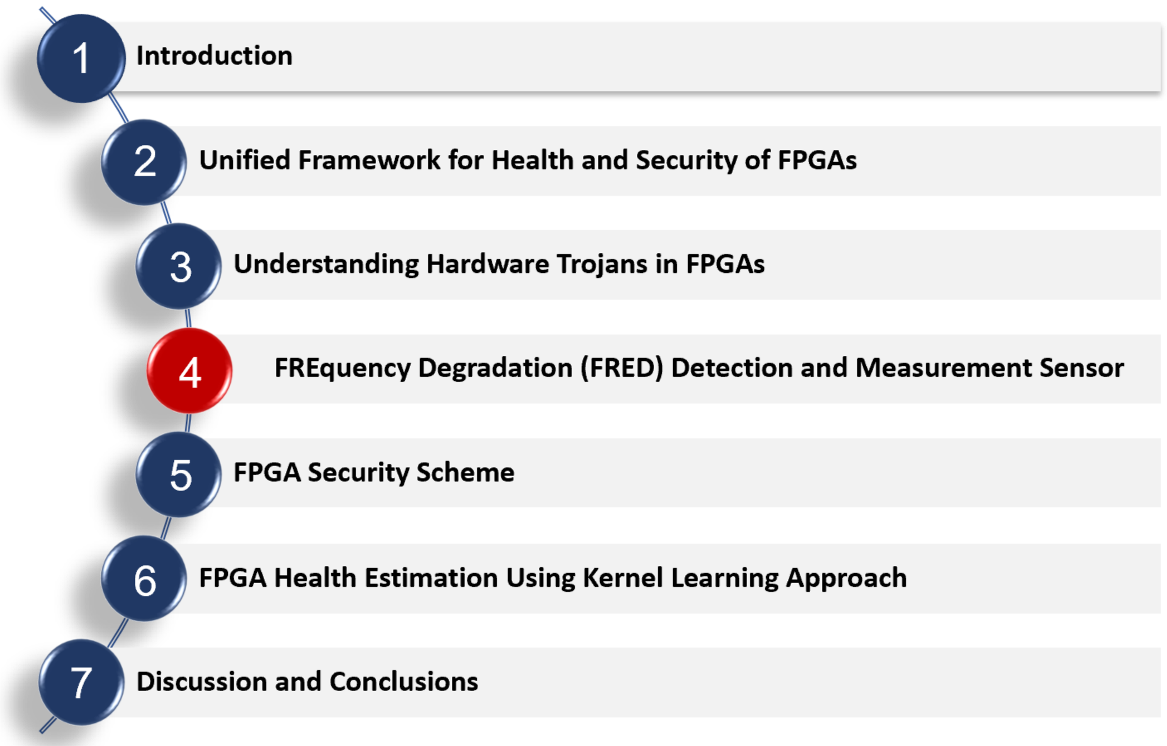

1
Introduction
2
Unified Framework for Health and Security of FPGAs
3
Understanding Hardware Trojans in FPGAs
4
FREquency Degradation (FRED) Detection and Measurement Sensor
5
FPGA Security Scheme
6
FPGA Health Estimation Using Kernel Learning Approach
7
Discussion and Conclusions

Figure 1-2 Thesis Organisation – Disposition of Chapters.

1.4.3 Chapter 4

This chapter is the outcome of '**Objective 2**'. It provides a deep insight into the current research on capturing delay variability in VLSI circuits, including FPGAs. The design and implementation of the FREquency Degradation detection and measurement sensor in a 28 nm process technology is elaborated. In addition, the simulation and real-time experimentation alongwith results and the proposed sensor's performance evaluation are presented.

This chapter is the reformatted version of the paper under submission to **Sensors** journal.

1.4.4 Chapter 5

This chapter is the outcome of '**Objective 3**'. It presents a comprehensive FPGA security scheme, comprising novel elements of hardware Trojan infection, detection, and mitigation, to protect FPGA applications against the hardware Trojan. Built around the threat model of a naval warship's integrated self-protection system (ISPS), this chapter proposes a threshold voltage-triggered hardware Trojan that operates in a threshold voltage region and remains stealthy with a very low area overhead. It delineates the hardware Trojan detection sub-scheme comprising a unique lightweight threshold voltage-aware sensor. An online transistor dynamic scaling (OTDS) to mitigate the impact of hardware Trojan is also presented as a hardware Trojan Mitigation sub-scheme.

This chapter is a reformatted version of the paper published in **IEEE Access** journal.

1.4.5 Chapter 6

This chapter is the outcome of '**Objective 4**'. It proposes an FPGA health estimation method that is developed using a unique kernel-based machine-learning approach. More specifically, this chapter focuses on estimating the health of an FPGA that is degraded as a result of NBTI initiated by hardware Trojans. A stochastic filtering optimization algorithm for accurate hyperparameter selection is also proposed to help improve the overall FPGA health estimation/prognostics accuracy. The chapter later presents the evaluation results of the developed method and the overall accuracy.

This chapter is a reformatted version of the paper under submission to **Microelectronics Reliability** journal.

1.5 List of Published/Submitted Work

1.5.1 Journal Publication

S. Aslam, I. K. Jennions, M. Samie, S. Perinpanayagam and Y. Fang, "Ingress of Threshold Voltage-Triggered Hardware Trojan in the Modern FPGA Fabric–Detection Methodology and Mitigation," in ***IEEE Access***, vol. 8, pp. 31371-31397, 2020.

1.5.2 Conference Publication

S. Aslam, M. Samie, I. Jennions; "Hardware Trojans and Smart Manufacturing – A Hardware Security Perspective", Advances in Manufacturing Technology XXXII: proceedings of the 16th International Conference on Manufacturing Research, University of Skövde, Sweden , 11–13 September 2018. (DOI: 10.3233/978-1-61499-902-7-305).

1.5.3 Virtual Conference Presentation

J. Buu-Sao, M. Samie, S. Aslam, et. al., "IoT Security – Hardware Perspective", December 2018, the IoT Day Slam 2018, VIRTUAL Internet of Things Conference: https://iotslam.com/session/iot-security-hardware-perspective/.

1.5.4 Under Peer Review for Journal Publication

S. Aslam, I. Jennions, M. Samie, S. Perinpanayagam, : " Reliability, Security, and Prognostics in FPGAs – An Integrated Approach". to: **IEEE Transactions on Device and Materials Reliability (under peer review)**

S. Aslam, I. Jennions, M. Samie, S. Perinpanayagam, : "FREquency Degradation (FRED) Detection and Measurement Sensor for Reliable and Secure FPGAs". to: **Sensors (under peer review)**

S. Aslam, I. Jennions, M. Samie, S. Perinpanayagam, : "FPGA Health Estimation Using Kernel Learning Approach". to: **Microelectronics Reliability (under peer review)**

REFERENCES

[1] Chino, ‘Xilinx Unveils Their Revolutionary Adaptive Compute Acceleration Platform’, [Online]. Available at: https://www.techpowerup.com/242538/xilinx- [Accessed : 15 July 2018].

[2] Grand View Research, “*Field-Programmable Gate Array Market – Cruising Ahead,”* [Online]. Available at: https://www.grandviewresearch.com/blog/field-programmable-gate-array-fpga-market-size-share. [Accessed : July 2020].

[3] M. Lipp *et al.*, “Meltdown : Reading Kernel Memory from User Space,” Commun. ACM 63, pp 46-56, 2020.

[4] P. Kocher *et al.*, "Spectre Attacks: Exploiting Speculative Execution," *2019 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, pp. 1-19, 2019.

[5] Z. Zhang, Q. Yu, L. Njilla and C. Kamhoua, "FPGA-oriented moving target defense against security threats from malicious FPGA tools," *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, Washington, DC, pp. 163-166, 2018.

[6] S. Zamanzadeh and A. Jahanian, "Scalable security path methodology: A cost-security trade-off to protect FPGA IPs against active and passive tampers," *2017 Asian Hardware Oriented Security and Trust Symposium (Asian HOST)*, Beijing, pp. 85-90, 2017.

[7] Z. Zhang, L. Njilla, C. A. Kamhoua and Q. Yu, "Thwarting Security Threats From Malicious FPGA Tools With Novel FPGA-Oriented Moving Target Defense," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 3, pp. 665-678, March 2019.

[8] P. Maene, J. Götzfried, R. de Clercq, T. Müller, F. Freiling, and I. Verbauwhede, “Hardware-based trusted computing architectures for isolation and attestation,” Trans. Comp., vol. 67, no. 3, pp. 361–374, 2018.

[9] H. Zhang et al., “Architectural support for containment-based security,” in Proc. Arch. Supp. Programm. Lang. Op. Sys., pp. 361–377, 2019.

[10] X. Wang, S. Si, C. Gao and J. Huang, "A method of FPGA interconnect resources testing by using XDL-based configuration," *2014 Prognostics and System Health Management Conference (PHM-2014 Hunan)*, Zhangiiaijie, pp. 203-207, 2014.

[11] Laung-Terng Wang, Yao-Wen Chang, Kwang-Ting (Tim) Cheng, *"Design for testability- Electronic Design Automation",* Morgan Kaufmann, pp 97-172, 2009.

[12] H. B. Shashidhara, S. Yellampalii and V. Goudanavar, "Board level JTAG/boundary scan test solution," *International Conference on Circuits, Communication, Control and Computing*, Bangalore, pp. 73-76, 2014.

[13] L. Pierce and S. Tragoudas, "Enhanced Secure Architecture for Joint Action Test Group Systems," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 7, pp. 1342-1345, July 2013.

[14] Bhunia, Swarup, Tehranipoor, Mark M. (Eds.), *The Hardware Trojan War-Attacks, Myths, and Defenses*, Springer International Publishing, DOI 10.1007/978-3-319-68511-3, 2018.

[15] J. S. J. Wong and P. Y. K. Cheung, "Timing Measurement Platform for Arbitrary Black-Box Circuits Based on Transition Probability," pp. 1–14, 2013.

[16] Z. Ghaderi, M. Ebrahimi, Z. Navabi, E. Bozorgzadeh, and N. Bagherzadeh, "SENSIBle: A Highly Scalable SENsor DeSIgn for Path-Based Age Monitoring in FPGAs," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 919–926, 2017.

2 UNIFIED FRAMEWORK FOR HEALTH AND SECURITY OF FPGAs

This chapter presents an integrated (unified) approach to prognostics and security in FPGAs by putting forth an '**Integrated FPGA Health Management (IFHM)**' framework. It begins with a succinct introduction to the architecture of a modern FPGA, highlights its commercial and industrial significance, and draws the canvas of the extant and the future technological revolution with FPGAs at the heart of it. Sections 2 and 3 excavate the FPGA reliability issues, vulnerabilities, threats, and several counteractive research efforts made in the realms of FPGA reliability, prognostics, and security to enhance its dependability and build trust. A high-level unified framework, called **IFHM** is then presented in Section-4 that provides guidance on managing prognostics and security in FPGAs as a composite entity for their security and reliability hardening. The conclusion summarises this chapter and briefly outlines the next stage of work. Figure 2-1 depicts the arrangement of this chapter. The main contribution of this chapter besides propounding IFHM framework includes the incisive and critical evaluation of FPGAs' security and reliability realms to uncover the subtleties of this reconfigurable integrated circuit. It is a unique effort, not endeavoured previously.

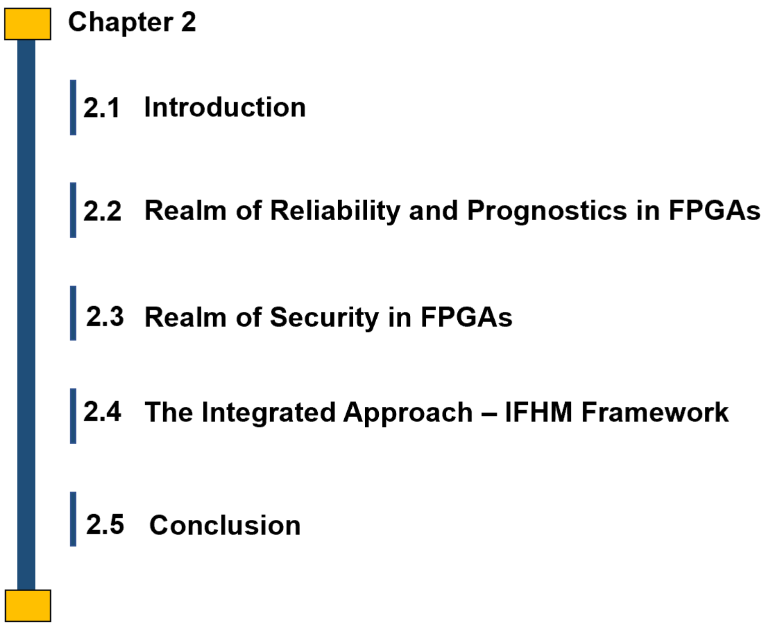

Chapter 2
2.1 Introduction
2.2 Realm of Reliability and Prognostics in FPGAs
2.3 Realm of Security in FPGAs
2.4 The Integrated Approach – IFHM Framework
2.5 Conclusion

Figure 2-1 The Disposition of Chapter 2.

2.1 Introduction

From simple Bluetooth devices to the NASA's Orion spacecraft, FPGAs have become the backbone of embedded system design. According to [1], the main driving factors for the exponential growth of the FPGA market are the increasing demand for advanced compute acceleration, autonomous and AI-based systems, the evolution of connectivity network to IoT, institutionalisation of cyber-physical systems (CPS), and the reduction in time-to-market. Through the merger of software and hardware properties, FPGAs provide an effective trade-off between the programmability of CPUs and the performance of application-specific hardware. Even though this flexibility helps developers to speedily prototype and deploy embedded systems with performance closer to Application Specific ICs, the programmability feature could be exploited to eavesdrop on encrypted communication, disrupt critical functionality, or even incur physical damage to the chip. Designing and developing systems that are both flexible and reliable, yet fundamentally sound from a security point of view, is an extraordinarily challenging venture for both researchers and practitioners. Quite often, the security facets of a reconfigurable entity, such as an FPGA, are not catered for until far too late in the design process, resulting in systems that are not reliable and hence, protected only by their obscurity.

2.1.1 The Increased Reliance on FPGAs

FPGAs are a vital element of many mission-critical systems, silently controlling and monitoring everything from wireless access points (WAP) to commercial face recognition systems. According to [1], the FPGA market is expected to reach 117.97 billion US dollars by 2026 growing at the compound annual growth rate (CAGR) of 7.2% during the forecast 2017-2026. This huge surge explains the growing significance of these massively parallel architectures.

As opposed to the sequential execution enabled by a general-purpose processor, modern FPGAs can carry out thousands of multiplies and adds each cycle, providing them the computational power to host numerous diverse logic modules simultaneously. For instance, an FPGA-hosted Wireless Access Point (WAP) application may use a packet scheduler with signal processing core and a protocol processing engine, all sharing the same FPGA primitives and silicon [2].

By virtue of this unique combination of computation power and flexibility, FPGAs are being regarded as the workhorses behind a wide variety of performance critical embedded systems [3]. They are capable of achieving high speedups and performance gain (100x) per unit of area as compared to a similar microprocessor [4]. High-end satellite systems, network-centric warfare equipment, intrusion detection systems, SMART grid, Industrial IoT devices, aircraft and avionics, and even the Mars Rover have great dependence on FPGAs to undertake their respective functional tasks. These devices help implement optimised circuitry for almost everything from encryption to FFTs, or even entire customized multi-processor systems by leveraging their bit-level reconfigurability. A gamut of such different domains leveraging FPGAs is shown in Figure 2-2. In order to understand this growing reliance on FPGAs, we need to examine the internals of the FPGA and its overall architecture.

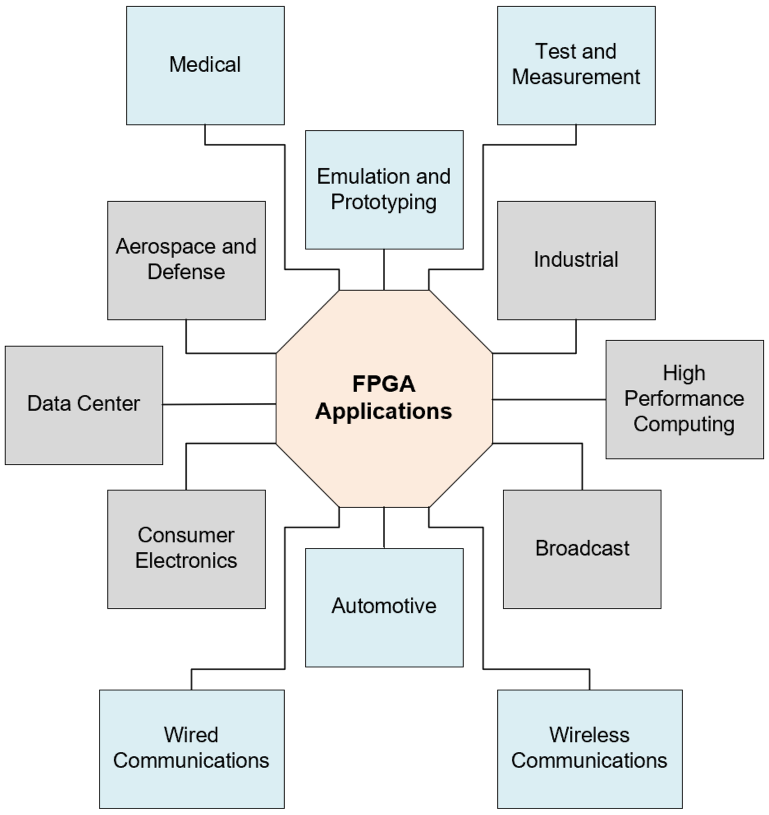


Figure 2-2 A Gamut of FPGA Applications.

2.1.2 The Internals of FPGA

An FPGA is a mesh of programmable logic gates embedded in a flexible interconnect, as shown in Figure 2-3. These logic gates are implemented by configuring look-up-tables (LUTs) for computational applications, flip-flops for managing timing across different applications, switching interconnects for laying routing network, and I/O blocks (IOB) for moving data/signal inside and outside of the FPGA. The logic gates are, basically, the structures with respective truth tables that enable the mapping of any circuit to an FPGA through LUT reconfiguration and by arranging bits in the switchboxes, which then specify the wires' connections through pass transistors. It is pertinent to mention here that the LUT and switchbox are programmed as defined by the configuration bitstream. The security of configuration bitstream, therefore, takes precedence and accordingly three different FPGA structures are commonly used, as shown in Figure 2-4. The ones that use EPROM/EEPROM or antifuse are write-once

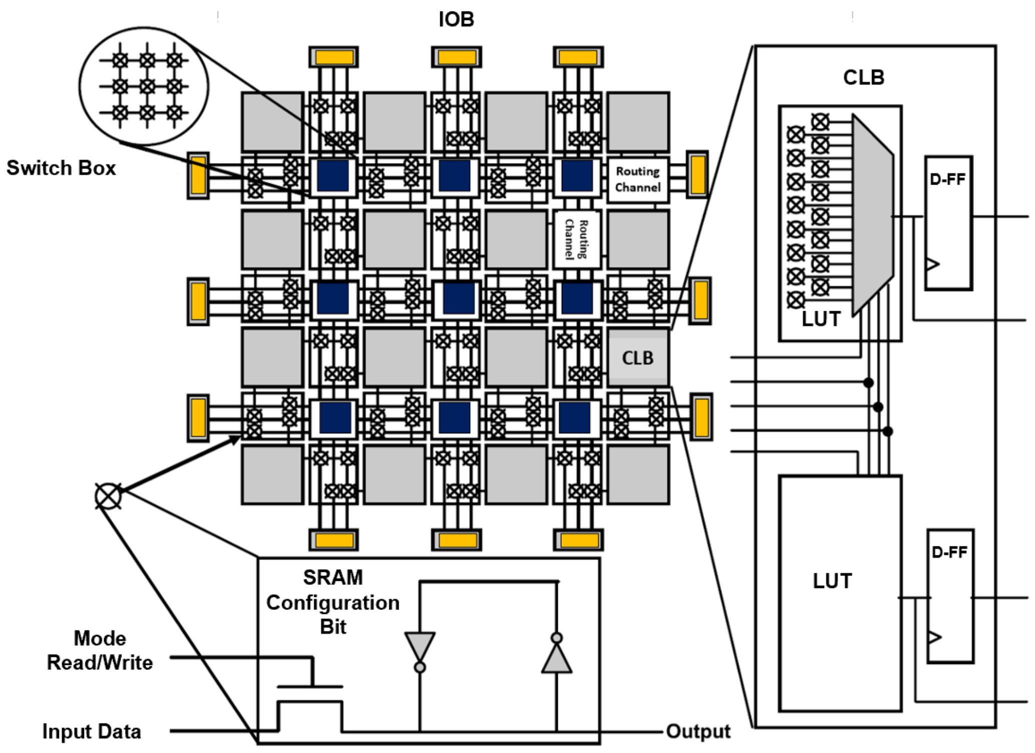


Figure 2-3 An architecture of a typical FPGA. The Configuration Logic Blocks (CLBs) are islands with a mesh of programmable interconnects around them. Each CLB houses a Lookup Table (LUT) that can be configured to implement any logic gate.

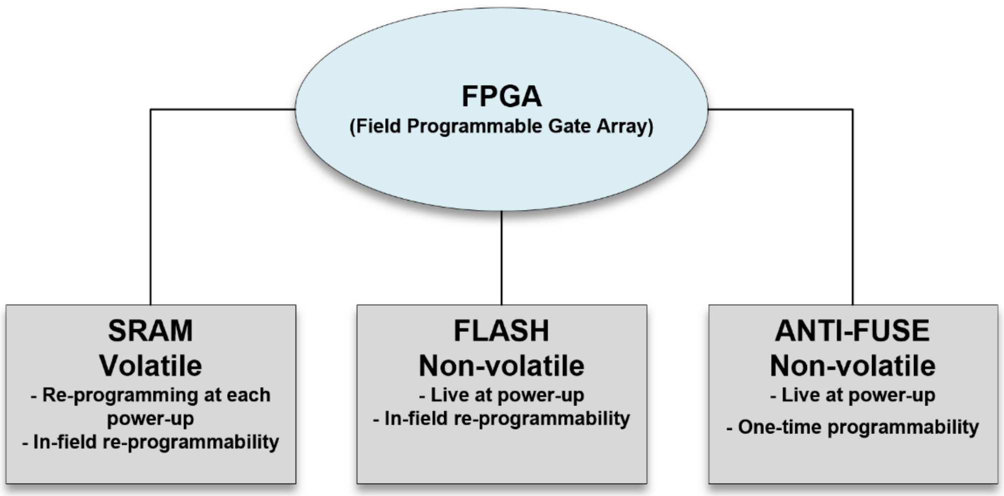

FPGA
(Field Programmable Gate Array)
SRAM
Volatile
- Re-programming at each power-up
- In-field re-programmability
FLASH
Non-volatile
- Live at power-up
- In-field re-programmability
ANTI-FUSE
Non-volatile
- Live at power-up
- One-time programmability

Figure 2-4 Three Types of FPGAs – SRAM, FLASH, and ANTI-FUSE.

technologies. On the other hand, the re-programmable structures are the Static RAM (SRAM) FPGAs. While a number of architectures make use of antifuse type, **Static RAM (SRAM)** architecture is the most preferred since it allows for reconfiguration – the essence of reconfigurable computing.

The SRAM programming bits are spread across the entire FPGA and stored locally with the LUTs and switchable interconnects. **This increases their vulnerability to probable performance degradations and reliability-downgrade due to undesired distributed hardware attacks.**

2.1.2.1 SRAM FPGA Specifics

Generally, a combination of two inverters and pass transistors is used to develop the Static RAM (SRAM) cells (see Figure 2-3). FPGAs based on SRAM cells are volatile in nature, meaning that as long as the SRAM cell is powered, the data remains stored and can be, therefore, read from the cell. However, as soon as there is no power, SRAM cell is unable to retain its value [5]. This feature offers some security and makes it difficult for the intruder to retrieve the configuration bitstream, thereby helps retain the data integrity.

In SRAM FPGAs, LUTs make optimum use of SRAM cells as programming bits. Healthy condition of LUTs is, therefore, vital to ensure unhindered operation and

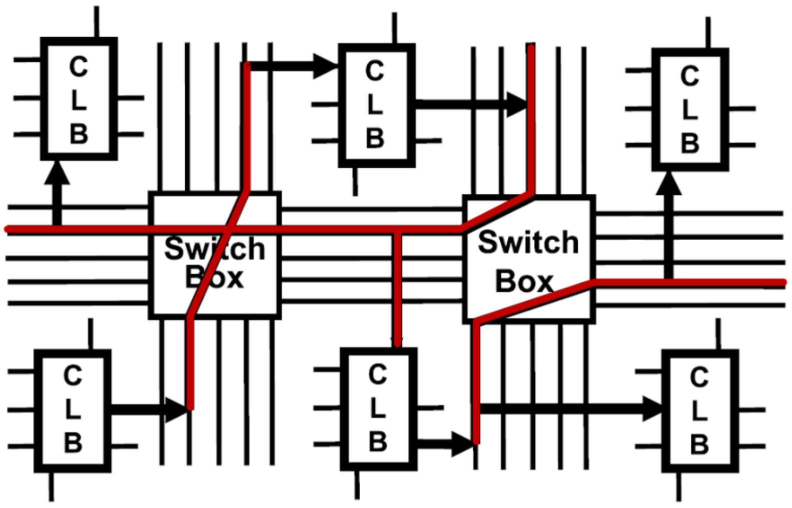


Figure 2-5 FPGA Interconnect Architecture. Programmable connections to and out from the CLBs [6].

optimum functioning of the implemented circuit patterns. Also, as seen in Figure 2-3, the LUTs in addition to flip flops, and multiplexers form an integral part of the larger regions, known as configurable logic blocks (CLBs). To connect the configuration and computational blocks together, there is a large routing channel, which is a set of pass transistors, providing programmable connections to and out from the CLBs (see Figure 2-5). The point-to-point connections between neighbouring routing channels containing longlines, on the other hand, are enabled through switchbox network. This complex routing architecture contributes to both the delays and area constraints in the FPGA, if not optimised. It is estimated that 80-90% of the typical FPGA area is occupied by the interconnect, facilitating both the physical wires and the configuration bits that link the wires together for any arbitrary interconnection network [6]. **Despite the criticality of the interconnect for configurability, it also poses complications in building a secure and reliable FPGA infrastructure.**

Keeping these FPGA architectural sensitivities in perspective, we delve the realms of reliability and security to build an integrated FPGA health management framework, in the ensuing sections.

2.2 Realm of Reliability in FPGAs

Reliability in FPGAs is their performance to specification over time in response to varied, but specified, environmental stress conditions. Hitherto, advanced manufacturing techniques have continued to maintain FPGA reliability at a level that is suitable for a large majority of applications. The semiconductor industry, based on ITRS, contemplates that the trade-off needs to remain at this level and continue to develop ICs with a failure rate of 50-2000 FITs (One Failure in Time equals to one failure in 10^9 hours) [7]. This means improving the reliability (in case of stochastic faults) of a transistor by a factor of five for technology nodes between 32 nm and 11 nm [8]. This may be a substantial challenge for a technology that is also faced with the manufacturing inadequacies in terms of process variation and yield complications.

The advanced techniques related to strained silicon and high- *k* gate dielectrics also pose considerable challenges for the suitable characterisation of device reliability. It, therefore, warrants implementation of innovative methods to realise continued scaling advantages and further enhance performance-efficiency. The deployment of multiple gates along with the new configurations of interconnect is one such example. However, the introduction of new materials and altering the structure of circuit components will result in substantial changes to the processes related to degradation. Ultimately, this may result in increased uncertainty about the reliability-performance relationship of devices.

FPGA reliability is envisaged to follow a downward trend, provided the existing thermal and power management mechanisms are optimised accordingly [9, 10, 11]. For instance, a considerable increase in the **current density** occurs with the decrease in the dimensions of the electron paths. This results in electromigration in interconnects – a particular concern. **Voltages**, though falling, are not matching the decrease in feature dimensions, thus generating **high electric field strength** - a dominant factor of acceleration in a number of degradation processes. Similarly, an increase in the **threshold voltage** is an indicator of FPGA ageing and the effect of this will enhance with shrunk supply voltage margins. Moreover, when it comes to the localised power dissipation, it is no different. The devices will experience increased levels of power dissipation due to higher circuit density. This may get compounded by a reduction in thermal conductivity, thereby causing higher junction temperatures (a key accelerating factor). In a nutshell, these wide-ranging challenges have the capacity to impact FPGA reliability in a number of ways. These could result in transient faults, such as the radiation-induced SEUs (Single Event Upsets), as well as performance degradation with transistor ageing [12].

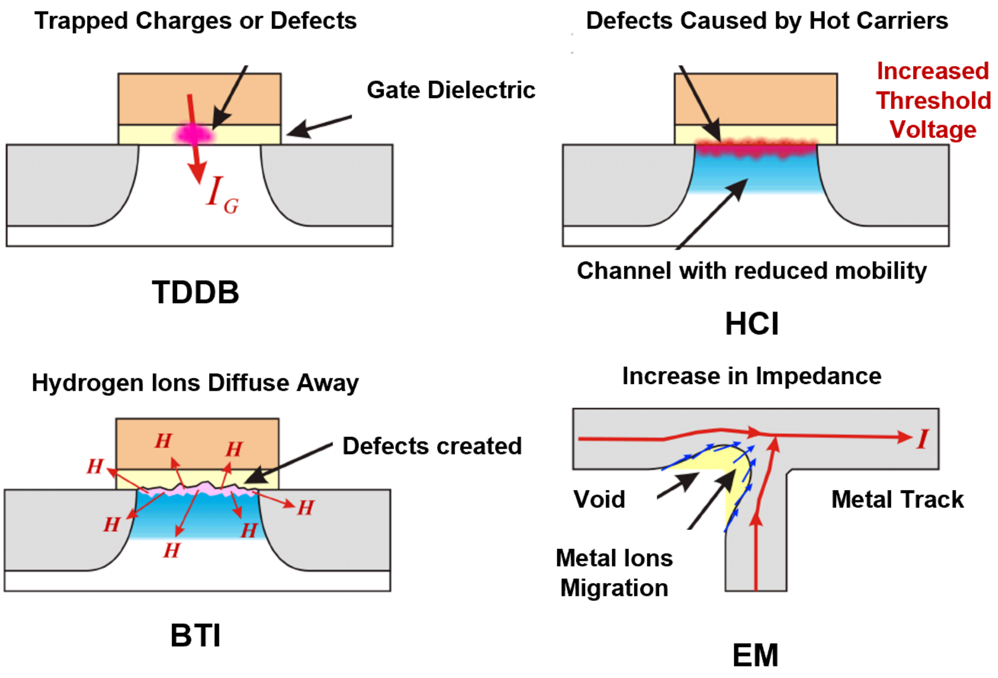
2.2.1 Degradation Mechanisms and Transistor Ageing

This subsection provides a concise overview of key degradation mechanisms that affect the reliability of FPGA in terms of CMOS transistors' ageing – a pressing reliability issue facing the VLSI devices at the nano-scale. The CMOS transistors are the underlying nano-architectures upon which the FPGA fabric is built. It is, therefore, prudent to investigate their performance, which ultimately helps adjudge FPGA health in terms of reliability.

Primarily, four degradation mechanisms are predominantly relevant to VLSI devices including FPGAs. These include: 1) Time-Dependent Dielectric Breakdown (TDDB), 2) Hot Carrier Injection (HCI), 3) Bias Temperature Instability (BTI), 4) and Electromigration (EM). In addition to these, the latch-up and soft-error generation are significant with respect to the environmental impact on the devices. These mechanisms are illustrated in Figure 2-6.

2.2.1.1 Time Dependent Dielectric Breakdown (TDDB)

TDDB mechanism is the formation of a conductive path via the gate dielectric due to the accumulation of trapped charges, or defects. These trapped charges result due to the strong gate-bias voltage. As these defects weaken the dielectric at any explicit



Trapped Charges or Defects
Gate Dielectric
I_G
TDDB
Defects Caused by Hot Carriers
Increased Threshold Voltage
Channel with reduced mobility
HCI
Hydrogen Ions Diffuse Away
Defects created
H
BTI
Increase in Impedance
I
Void
Metal Track
Metal Ions Migration
EM

Figure 2-6 The Degradation Mechanisms impacting the FPGA reliability [13].

location, a considerable amount of leakage current (I_G) starts flowing with rapid increase in magnitude due to the reinforcement of the path [13].

2.2.1.1.1 The Consequences

The consequential impact of TDDB is variable in nature due to the random variation of the breakdown path in conductivity and its physical location. When the breakdown condition is of mild nature, the increased leakage current results in increased power consumption and reduced switching speed. As the breakdown becomes severe in nature, higher power consumption is observed, which eventually prevents the transistor from switching, completely [14]. The main drivers of TDDB are the high temperature, thin oxide structures, and high electric fields and must be monitored for abnormalities.

2.2.1.2 Hot Carrier Injection

This mechanism also known as the hot-carrier or hot-electron effect, is also based on defect accumulation process. However, it occurs in the interface region between the channel and the gate dielectric. The hot carriers in the channel with energies high enough to escape the potential barrier of the gate dielectric induce defects at the interface. When accelerated by the gate field, a series of collisions take place with the ions present in the interface region. This generates defects, which in turn, lead to an increase in threshold voltage and a decrease in the drain current or carrier mobility, resultantly slowing down the transistor switching [15]. This mechanism is particularly dynamic in CMOS, meaning it occurs when the transistors switch. The main factors that aggravate HCI are the high carrier velocities and shorter channel length.

2.2.1.3 Bias Temperature Instability (BTI)

The BTI phenomenon manifests itself as a shift in the threshold voltage of MOSFETs with high temperature and negative/positive bias [16]. This causes switching delays in the transistor and consequently, as the delay of functional paths transcends the timing requirements, we can observe the signs of circuits' fatigue and hence, the ultimate failure. This can significantly reduce the operational lifetime of FPGA devices, thereby downgrading their reliability.

2.2.1.3.1 The Factor of High-k Dielectrics in 45nm and below FPGA technologies

This research work, as mentioned in Section 1.2.2.2, is focused on 28 nm process technology which is mainly made up of high-*k* dielectric metal gate transistors as opposed to low-*k* dielectric metal or polysilicon for > 48 nm process technologies [17]. In order to better understand the phenomena of BTI in a 28 nm process technology , it is important to first know as to what is high-*k* dielectric and how it helps achieve increased power efficiency and low leakage on one hand and why, on the other hand, BTI is becoming a critical reliability challenge in high-*k* dielectric metal gate transistors.

Essentially, dielectric is an insulating material that does not conduct electricity well, if at all. The measure of dielectric is given by dielectric constant '***k***', which is a parameter that defines the ability of a material to store energy or charges. Materials have different dielectric constants at room temperatures (e.g. Air = 1, Silicon dioxide = 3.9, Aluminium oxide = 10.1, and Hafnium oxide = 25 [15]). Any dielectric with a '***k***' value less than the conventionally used SiO_2 (***k*** = 3.9) is termed as a **low-k dielectric**. Whereas, dielectric with '***k'*** greater than that of silicon nitride (***k*** = 7) is regarded as a **high-*k* dielectric** [18]. A low dielectric constant of a material means that the material has a low ability to polarize and hold a charge. A high dielectric material is good at holding a charge and is therefore the preferred dielectric for CMOS construction.

Although, high-k dielectric leakage through the gate (gate oxide) is reduced by more than a factor of 10, the other significant leak, called source-to-drain or subthreshold leakage is becoming a source of concern [19]. It's a trickle of current seen even when the transistor is intended to be in the "off" state. Making transistors smaller has also meant steadily lowering the amount of voltage needed to turn them on, the threshold voltage. Unfortunately, steadily lowering the threshold voltage lets more current slip through. For many years, each new generation of transistor would increase drive current (and improve performance) by about 30 percent but would pay the price of about a threefold increase in subthreshold leakage [19]. Leakage currents have reached levels high enough to be a noticeable portion of IC's power consumption.

But a transistor can be designed to operate by adjusting the channel length or adjusting the threshold voltage [19]. A shorter channel leaks more but allows for a higher drive current. A higher threshold voltage pinches off the leak but also throttles the drive current. Adjusting the threshold voltage is where the high-*k* dielectric comes into play. A thicker dielectric reduces the gate's ability to open a conductive channel between the source and the drain, increasing the threshold voltage. A thinner dielectric layer has the opposite effect. Compared with the previous 65-nm transistors, 45-nm and below high-*k* plus metal gate transistors provide either a 25 percent increase in drive current at the same subthreshold leakage or more than a fivefold reduction in leakage at the same drive current, or anywhere between those values [19].

However, the problem lies in the low thermal conductivity of high-*k* dielectric materials that leads to self-induced thermal runaway and breakdown [20]. As a result, this influences BTI mechanism, which tends to be equally pronounced for 28 nm process technology as it is for devices greater than 45 nm.

2.2.1.3.2 NBTI and PBTI

BTI mechanism comprises two different degradation phenomena namely, negative bias temperature instability – NBTI and positive bias temperature instability – PBTI. NBTI impacts the PMOS transistors whereas PBTI affects the NMOS transistors. In technology nodes above 45nm, the impact of PBTI was insignificant as compared to NBTI. However, with the development and introduction of high-*k*/metal gates transistors in sub 45nm process nodes, the PBTI phenomenon has become equally

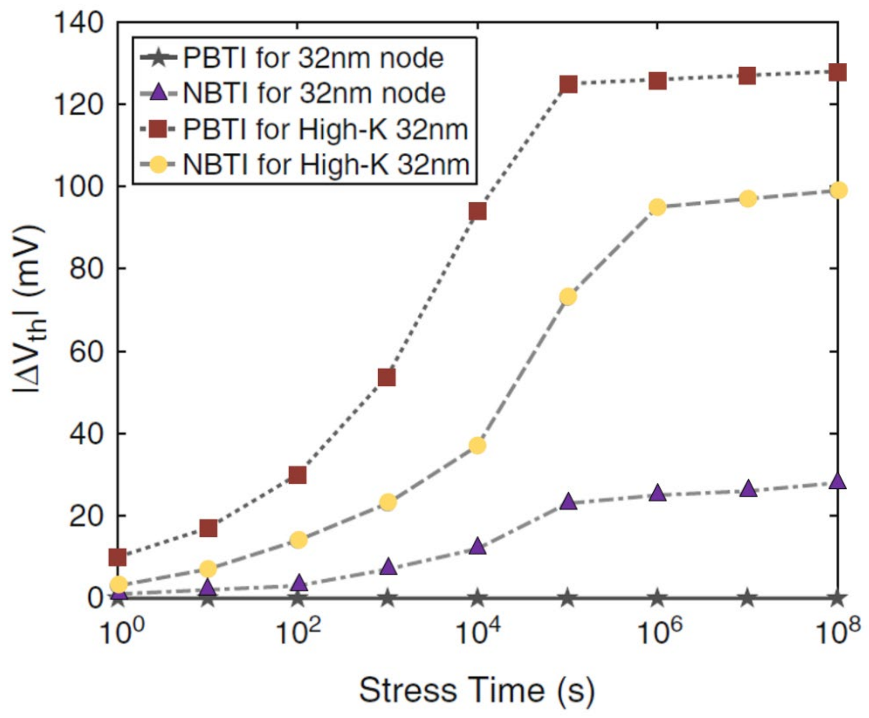


Figure 2-7 Shift in Threshold Voltage ΔV_{th} with high-*K*/Metal gates is becoming significant [19].

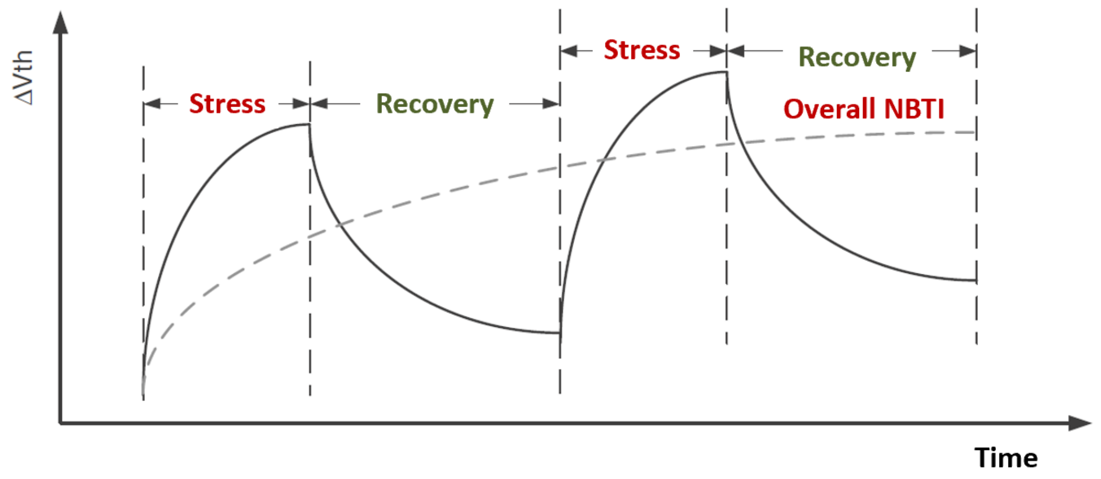


Figure 2-8 BTI-induced variations in V_{th} during the stress and recovery period.

significant for NMOS transistors as NBTI for PMOS transistors [21] (see Figure 2-7). It is, therefore, prudent to consider the combined effect of both the BTI components whilst evaluating the device reliability.

NBTI (PBTI) comprises two sequential phases – the Stress phase, during which the gate-source voltage is reversely (positively) biased i.e. $V_{gs} = -(+)V_{dd}$, and the Relaxation phase where $V_{gs} = 0$. At the initiation of the stress phase (i.e., when the transistor is ON, $V_{gs} = -V_{DD}$ for PMOS under NBTI and $V_{gs} = V_{DD}$ for NMOS under PBTI), few interface traps are generated at the interface of channel and gate oxide [22]. It is at this instance of time, the generated interface traps result in increasing the magnitude of threshold voltage (V_{th}). During the relaxation phase, when $V_{gs} = 0$, a small number of the interface traps are annealed. This leads to a decrease in the magnitude of transistor V_{th}. It is worth noting that this recovery cannot fully compensate the effect of stress phase. As a result, the overall impact of BTI is a rise in the magnitude of threshold voltage over the time [23] (see Figure 2-8).

2.2.2 FPGA Manufacturers' Perspective

It is important to be aware of the FPGA reliability practices employed by manufacturers to build an understanding of the various test methodologies adopted in their qualification and eventually identify room for improvement. This section, therefore, gives a concise account of different reliability program tests the FPGA manufacturers are practising.

2.2.2.1 Post-Manufacturing Reliability Test Regime

Generally, manufacturers have a product reliability goal for long term failure rate. This implies a failure rate of <200 FIT at 55ºC use condition and the product meeting lifetime goal of 100,000 hours of useful life. The reliability qualification and monitoring requirements, as given in Table-2-1, provide an overview of different types of post-FPGA manufacturing tests conducted.

It is noted that all these reliability tests are conducted less of security considerations, which if integrated with these requirements, would provide a more trustable product. This, however, does not imply that manufacturers do not incorporate security features in these state-of-the-art devices.

2.2.2.1.1 Life Test Methodology

The life-test methodology is implemented to accelerate failure mechanisms, including the wear-out degradation, as mentioned in Section 2.2.1. According to the manufacturers' reliability qualification and testing programs [24], [25], the FPGA life-test is carried out keeping the junction temperature at 125°C and the $\boldsymbol{V_{cc}}$ power supply is amplified between 10-20% (kept constant for a specific test duration). However, in

Table 2-1 Portfolio of post-manufacturing reliability tests conducted by FPGA Manufacturers [24-25].

TYPE OF TEST	STANDARD	METHOD / CONDITION
Life Test	JESD22-A108	1000 hours @ 1.1 - 1.2 x Vcc, Tj:110°C min - 140°C max 2000 hours for reference
High Temperature Retention Bake	JESD22-A103	1000 hours min. @ 150°C
Temperature Cycling	JESD22-A104	Preconditioning + 700 cycles. -55°C to +125°C
Biased Humidity/Temp	JESD-A101	Preconditioning + 85°C, 85% R.H.; 1000 hours @ Vcc nom;
H.A.S.T (Highly Accelerated Stress Test)	JESD-A110	130°C, 85% RH, 48 or 96 hours, @ Vcc nom.
Autoclave	JESD22-A102	121°C, 15 PSIG; 96 hours, 168 hours for reference
Unbiased H.A.S.T	JESD-A118	130°C, 85% RH, 96 hours
ESD	JESD22-A114 Mil Std 3015.7	100 pf, & 1500 Ω.
Latch-up	JESD 78	(Icc nom. + 100mA) or Icc nom. + 50% on I/O, Vcc + 50% on Power Supplies
Program/Erase Cycling	-	Program/Erase 100 cycles (EEPROM or FLASH)

certain cases where there is a risk of thermal runaway (a process, when accelerated by increased temperature, releases energy that causes the temperature to rise further to a point where it becomes self-sustaining) due to high junction temperature at 125°C, it is recommended practice to use a minimum junction temperature of 110°C. These tests are carried out using the life-test boards, installed with special high temperature sockets that help maintain lead integrity [24].

It is pertinent to mention that each device is prior-tested using production test equipment to data sheet specifications before the exposure to different stress conditions. All readouts are also performed on the same production test equipment in accordance with the parameters, defined in respective data sheets. A device is classified as a failure if it is unable to pass the specifications laid out in the data sheet.

2.2.2.1.2 Failure Rate Prediction

In order to attain an accurate and precise measurement and projection of an FPGA failure rate, some manufacturers assess each expected failure mechanism individually. The failure rate prediction process for each mechanism begins with the calculation of acceleration by employing an appropriate model and using the suitable constants. This is followed by determining the exponential distribution of time to failure (TTF) for each mechanism and then adding up individual failure rates to calculate the device total failure rate. Accordingly, the cumulative distribution function, $F_i(t)$**,** for each mechanism is determined by taking their product as $\Pi\ F_i(t)$ for i=1…n. As the distribution of time to failure for each mechanism is assumed to be exponential, the individual failure rates can be combined as $\Sigma\lambda_i$**,** which is basically the representation of the geometric mean.

We have also observed that some other manufacturers only consider an average activation energy (usually $E_a=0.7$) and then apply an Arrhenius model to the High-Temperature Operating Life (HTOL) test results. They do not take into consideration the individual failure mechanisms. Such practices do not guarantee the optimum performance of devices under challenging environments as they have not been tested for different degradation mechanisms. Although such tests cost less from both the manufacturing and time viewpoint, they may exhibit low MTBF, lower customer confidence and eventually cost higher during operational lifecycle.

Generally, the manufacturers determine the failure rates on the product family basis (for instance, Spartan-6, Virtex-7 etc.). So, it is not possible to define a standardised failure rate for all the FPGA devices collectively, which is quite realistic. However, it is possible to have an average range of failure rate values to assess their reliability. With respect to the device hours, it is a common practice to convert the hours accumulated during the stress conditions to normal use conditions by plugging in the acceleration factors. Whereas, the equivalent hours are determined under a typical use condition with nominal $\boldsymbol{V_{cc}}$ at 55°C still-air ambient or 70°C junction temperatures.

The failure rates are expressed as FIT or Failures In Time, where one FIT is equivalent to one failure in one billion or 10^9 device-hours [26]. Mathematically,

$$Failure\ Rate\ =\ \boldsymbol{X}^2\ /\ 2\ x\ \boldsymbol{A.F.} x\ Device\ Hours\ (failures/hr) \qquad (2\text{-}1)$$

where, ***X*** is the number of failures, ***A.F*** is the acceleration factor – product of thermal and voltage acceleration, whereas Device hours is given as:

$$Device\ hours\ =\ \boldsymbol{\Sigma}\ (hours\ in\ life - test)\ x\ (No. of\ devices) \qquad (2\text{-}2)$$

The JESD85 standard is most commonly employed for FIT rate calculation by most of the manufacturers. A Chi-squared distribution, for instance, is used to predict a 60% confidence level derived usually from the small number of failures and limited sample size of the tested devices.

2.2.3 FPGA Degradation/Reliability Modelling – Researchers' Perspective

This sub-section takes into account the various FPGA degradation and reliability models that have been developed and proposed to predict the failure rate in FPGAs due to the impact of different stochastic and systematic variations on their primitives' performance.

As a first example, researchers in [27] have characterised the delay degradation of LUTs using the duty cycle and frequency of stress signal as the main driving factors under the influence of Hot Carrier Injection (HCI). They have confirmed the dependency of HCI ageing mechanism on the frequency of input signals through transient simulations at the transistor level. However, their model lacks the measurements related to transistors' ageing parameters (delays and threshold voltage) and hence does not provide substantial evidence of reliability prediction or health assessment.

Edward *et al* [8], have studied the impact on FPGA reliability by analysing the changes that FPGAs experience as they age, and the varying factors that influence their performance. They concluded that the FPGA interconnects are less affected by degradation than the LUTs. But a lower-level approach is required when the design of basic resources – the N/PFETs – is to be considered to draw firm conclusions about degradation behaviour at the FPGA fabric level (LUTs, CLBs, Registers, Interconnects etc.) and an accurate assessment of reliability that can help estimate the overall health.

Similarly, the researchers in [28] have presented a detailed analysis of the impact of ageing on FPGA routing resources and data integrity of FPGA configuration cells. According to their study, the ageing of SRAM cells does not have any noticeable impact on the performance of FPGA. However, it does not consider the overall health assessment approach.

2.2.3.1 Fault-Tolerance and Self-healing – Sources of Reliability Enhancement

Interestingly, the researchers in [29] and [30] have proposed bio-inspired models that are not only aimed at building tolerance in reconfigurable devices against transient faults and soft errors, but also provide viable ideas to construct schemes like self-defence against hardware threats and self-repair strategies against hardware attacks. For instance, the optimal partitioning of cells to settle the optimal number of active and spare molecules gives a valuable reliability analysis. Also, it provides a design for dependability which considers the reliability as its essential attribute. According to [29], the reliability of a system is the consequence of the reliability figures of all its subsystems. This gives an inkling of the concept of an integrated approach towards FPGA health management.

Another effort relates to the self-healing electronic systems, inspired by eucaryotes and procaryotes [30]. The author in this study has talked about the equivalence of DNA fragments to memory cells. These fragments are representative of the characteristics and various functions of the cells. As a result, the faulty genes are extractable from the neighbouring cells and then based upon the correlation mechanisms, the damaged cells self-heal and re-establish their original states. The enormity of this self-healing system is traceable to its hierarchical nature. Right from the logic blocks corresponding to the biological molecules to an electronic array mirroring bacterial biofilms and a perfect likeness of a bus to cytoskeleton, the self-healing scheme presents a viable fault-tolerance and defence mechanism.

These two propositions, based on biological systems, are a good example of the efficient defence, repair, and heal mechanisms that can bolster FPGA reliability. However, they do not consider connecting the prognostics/reliability assessment approach with the security dimension.

2.2.4 Some Analysis

A critical analysis of the above discussion uncovers and reveals some key points. Firstly, a keen assessment of the degree of deviation or degradation from an anticipated normal operating condition provides data that can be carefully investigated to estimate instantaneous health condition of electronic/semiconductor devices. Secondly, it can be instrumental in providing advance warning of failures and hence curtailing unscheduled maintenance, stretching maintenance cycles, and maintaining effectiveness through timely repair and maintenance actions. Finally, adding a dimension of prognostics/health estimation would help cut the life cycle cost of equipment by reducing inspection costs, disruptions as well as the inventory.

2.2.5 Why Prognostics?

According to [31], prognostics is a process that helps determine the component/ system's remaining useful life by predicting the state of fault under the given extent of degradation, the load history, and the projected future operational and environmental circumstances to estimate the time at which the component/system is adjudged as unreliable. Whereas, health management is related to the decision-making process, which helps implement actions based on the estimate of the state of health derived from health monitoring and the probable future usage of the system.

Besides the reliability and degradation modelling and assessments (mentioned above), it is very vital to prognosticate FPGAs' health, especially with their progressive evolution into system-on-chip (SoC) and adaptive compute acceleration platforms (ACAPs). A simple reliability analysis and degradation modelling will not provide a holistic account of the FPGA health, which implies a direct or an indirect impact on the performance of the system and inappropriate health management. We have discussed the methods and techniques related to the degradation and reliability modelling in previous paragraphs. It clearly provides us with various useful methods that can be utilized in a composite manner to construct holistic FPGA health and security schemes into an integrated FPGA health management framework, as is described in Section 2.4.

2.3 Realm of Security in FPGAs

To assume that hardware implementations are secure in this area of FPGA security, is indeed a false belief. Since the majority of attacks come through software, it is then assumed that they will only come by this means. While this may be the case for the majority, it does not dismiss the fact that hardware attacks are just as exposed (exemplified by the recent Meltdown and Spectre cases [32], [33]). Abstraction levels in a complex design entail knowing the security of each step – thereof, in order to create higher abstraction levels, one must assume that the lower ones are secure in their operation. If the digital design of the system is compromised then it cannot completely be regarded as a secure system, disabling the overall formation of a secure implementation. Therefore, in order to assess the reliability of a device and its level of support towards a system, it is necessary to assess security levels in FPGAs (in particular, SRAM-based volatile FPGAs). These entail the transfer of configuration content in FPGAs, utilising FPGAs for security purposes, and using them as an adversarial tool.

2.3.1 FPGA Life Cycle and A Network of Hardware Threats

Both hardware and software attacks are included in the ways in which a computing device can be exploited. The exploitation ranges from being able to steal confidential information, to enabling systems to perform devious activities, leading to a complete destruction of the system [34]. The power and cost-effectiveness of reconfigurable hardware has had a conflicting effect, in that, although it has become more attractive to designers, it has in turn, made the system more vulnerable to attacks [35]. This section looks at the gamut of hardware security that has been ignored (until very recently). There could be potential attacks that aim at making changes to the hardware, spotting sensitive information using side channels, inserting unintended applications using the design tools, and stealing intellectual property. Here, we attempt

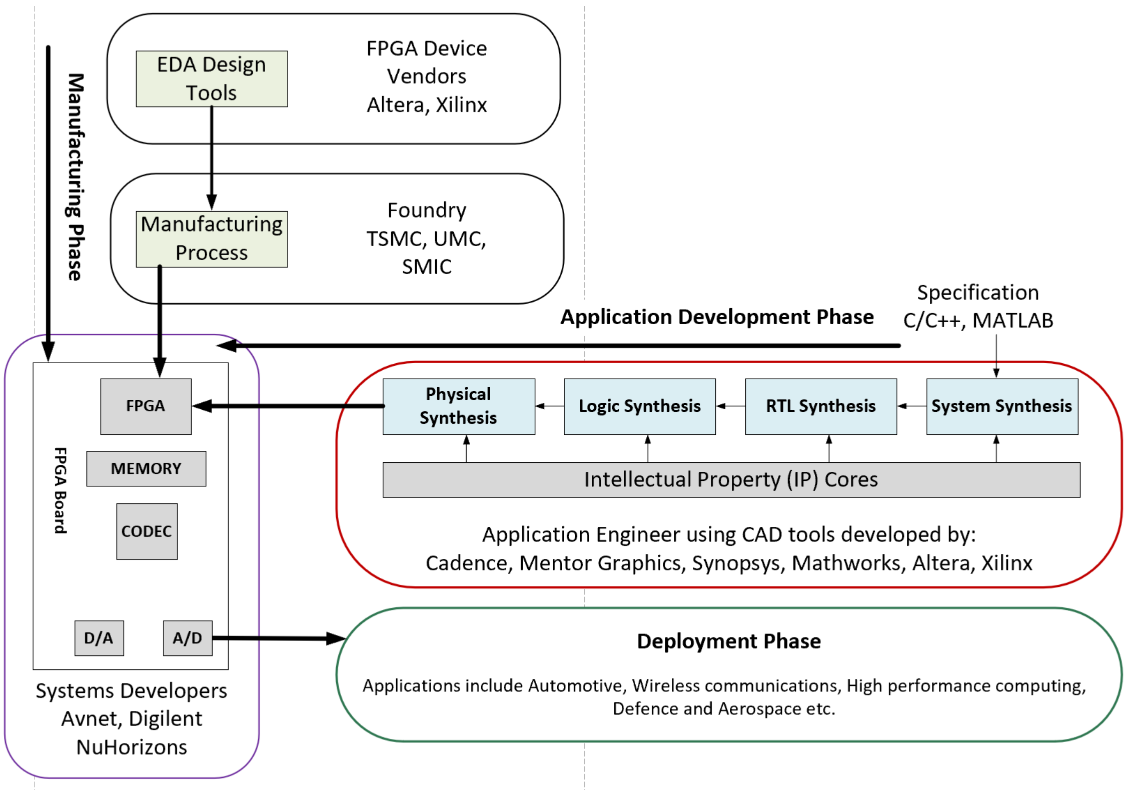

Manufacturing Phase
EDA Design Tools
FPGA Device Vendors Altera, Xilinx
Manufacturing Process
Foundry TSMC, UMC, SMIC
Application Development Phase
Specification C/C++, MATLAB
FPGA
MEMORY
CODEC
FPGA Board
D/A
A/D
Physical Synthesis
Logic Synthesis
RTL Synthesis
System Synthesis
Intellectual Property (IP) Cores
Application Engineer using CAD tools developed by: Cadence, Mentor Graphics, Synopsys, Mathworks, Altera, Xilinx
Systems Developers Avnet, Digilent NuHorizons
Deployment Phase
Applications include Automotive, Wireless communications, High performance computing, Defence and Aerospace etc.

Figure 2-9 FPGA Lifecycle – Manufacturing to Application Development to Final Deployment. Handled by several sources using a diverse range of design gadgets. Every phase of the FPGA lifecycle is prone to security threats that need to be collectively addressed to ensure and uphold the device and system reliability [36].

to highlight different aspects of security related to FPGA exposed to a plethora of hardware threats and vulnerabilities.

2.3.1.1 Vulnerabilities in FPGA Lifetime

FPGA lifetime can be divided into manufacturing, application development and deployment phases, as shown in Figure 2-9, to find vulnerabilities and gauge the intrusion of security issues at different levels .

2.3.1.1.1 Manufacturing Phase

Every 12 to 18 months, Altera and Xilinx, the leading reconfigurable hardware producers, have a new product. There is a lack of disclosure when it comes to the specifics of the product but reverse engineering that would create invasive physical attacks – although a complicated method – makes the possibility to gain specifics much more probable [36]. Once obtained from a third-party manufacturer, the FPGA companies sell the product to a system developer or partnering company to further the process to its final stages. A development board, consisting of an FPGA and its features of memory, audio, video, etc, is created to be sold further on. Thereof, the board is sold to specific industries – medicine and computing to name a few, in order that they may customise the product for themselves.

2.3.1.1.2 Application Development Phase

The application development phase allows for the FPGA to be integrated into the target system and programmed/reconfigured for the intended application. Development can take two routes - one of which includes the developer making their own platform for their own purposes using the FPGA chip [37] . The other route is that of using the development board provided by the third party.

CAD tools such as electronic system design (ESL) are utilised to translate high level language (C/C++, MATLAB, System C) to a register transfer level (RTL) hardware description language (HDL). Xilinx AccelDSP and Mathworks HDL Coder are both examples of such tools. A logical netlist is then created by means of combining the RTL by using tools from EDA companies. In order to program the FPGA, physical synthesis tools are necessary as they reconstruct the logical netlist into a bitstream.

Microcontrollers, signal processing cores and encryption cores are customised high-level functions that ESL design tools make use of. Each core requires a level of trust, with no one core (e.g. from tool vendors) having the same level as the other (e.g. from an online repository such as open-cores). It all depends on the source – where the core comes from. Intellectual Property (IP) cores can be distributed across the tools – distribution is a significant aspect because it makes reverse engineering more complicated, particularly when it is specified as a bitstream [38].

2.3.1.1.3 Deployment Phase

The point where a reconfigurable hardware is placed into the environment is when it reaches its last stage. This occurs when FPGAs are integrated into common devices and vehicles, all of which need this system. This system dictates how exposed a device is in terms of security – including both physical security (easy access and handling) as well as its practical use [39].

2.3.1.2 Securing the FPGA Life Cycle

Security concerns must be tackled from the initial point up to the final one. This is so that a greater understanding of the device is gained, in order to assess its flaws appropriately. Hence, enabling an effective response against the attacker. The way this is done is by considering the purpose of the device and external factors surrounding it – such as testing equipment and development tools.

Moreover, to ensure that the system is reliable for security purposes, a management plan outlining proper procedure is necessary. Assessments must test vulnerability and strength of the device in order to understand how exposed its security is [40]. In doing so, appropriate measures can be taken to ensure that the device is conducting optimal performance for its users.

Pertinent to mention is that the software's (EDA tools) role comes into play when the hardware is being made – this is because the hardware at that stage depends on that software, making it necessary for these assessments [41]. From this, the difference between an FPGA and a software is considered. The question is: by understanding the security problems and assessing how to resolve those problems, can the difference be made evident? In the following sections, we attempt to find answers to this.

2.3.2 Nature of Threats and Attacks on FPGAs

It is worth noting that security being compromised and then reconstructed is a cyclical process that remains ongoing. This means that as soon as there are measures put in place to strengthen security levels, they tend to be made redundant by the attacker. Thereof, prompting stronger, more effective measures to be inputted in the hope that another breach does not occur. To best exemplify this occurrence are smartcards. The level of security within these evolved overtime – a reaction to the exploitation of its initial, unsophisticated design.

A similar precautionary element can be noted in the development of FPGAs. They make up a pivotal part of the security factor in devices, making them more prone to attack. Those attacks may include that towards the device that uses the FPGA, the FPGA itself, physical attacks, and system-level attacks.

2.3.2.1 Counterfeiting

The function of FPGAs is that of a general use, in the sense that if they are made for one device, they can be used on others just the same. This essentially implies that bitstreams are easy to make copies of and use. The process of making those copies itself is quite general – it can be accomplished by a logic analyser and a skilled technician. This, however, also means that those copies would be of a lesser quality. It runs the risk of being marked as the original of its kind, negatively affecting the developer because the fake would be exposed as such easily. In this way, fraud has prevailed as an increasing issue where third parties may develop the hardware and sell any extra product without the responsibility of paying development costs [42]. That being said, this issue is handled by companies by labelling certain production facilities as reliable, through ensuring their close scrutiny and supervision.

Although mislabelling of FPGAs makes reverse engineering more complex, it tends to cause distrust amongst the buyer and the distributor because of the uncertainty surrounding the product. Smaller FPGA types would be easier to verify but when it comes to speed grades, it is more difficult [43]. It may be the case that slower ones be marked as faster and sold in that way. Commercial companies may find that vendors are more reliable than online buying because there would be no way for them to distinguish between a real and faulty package, unless they ran tests on it and obtained its results. Moreover, the reluctance of companies to publicise the number of frauds they have come across makes it harder to put an accurate figure on these occurrences.

2.3.2.2 Reverse Engineering

Bitstream reversal is the process of transforming an encoded bitstream into a functionally same description of the actual design – a reverse of the process from bitstream to HDL or netlist [44]. Furthermore, partial bitstream reversal is the mining of data from the bitstream (keys, BRAM/LUT content, or memory cell states) without replicating complete functionality. Although legal, reverse engineering is restricted for interoperability reasons or detection of breach of patents. If fully reversed, a bitstream's complete design and data could be exposed. This would lead to the data being used to make another bitstream, different from the initial one and violation of it would be more complicated. Hidden keys would be exposed as well. In the event that the attacker distinguishes the cryptographic algorithm, it would allow partial reversal to be of use.

Cryptographically, the bitstream’s code is not completely concealed but is still indeterminate [45]. FPGA vendors tend to keep this information as classified as they do for its design and layout.

The level of difficulty of reverse engineering is determined by the size, how obscure, and how complicated the bitstream is. As of yet, there are no reversals of modern FPGAs that have proven successful, nor any estimation of cost that is supported by data and analysis [43].

Bitstream encoding would no longer be depended on though, if reverse engineering became a problematic element, despite the repercussions of full reversal still being unknown. It is not the best strategy to hide keys in look-up tables and RAMs because all it takes is a very basic knowledge of bitstream construction and partial reversal to determine the information [46].

2.3.2.3 Readback

Readback is a snapshot of the FPGA’s current state while it is still operating. The FPGA sends the snapshot, after being requested to do so. Configuration, look-up tables, and memory contents to the host PC, by means of the configuration port are all included in that snapshot. This image is not the same as the original bitstream as the header, footer, initialization commands, and no-ops are not included. The dynamic data in LUTs and BRAMs is also not the same. Readback is efficient in verifying and testing FPGAs and allows the design to be corrected as it operates on the FPGA.

In the case that it is enabled though, an attacker can readback the design, add the missing static header and footer and use it in another device. They could then go on to re-program the FPGA with a modified version, or reverse engineer it. Active “readback difference attack” also occurs [47]. This allows the attacker to observe signal changes on an individual clock-cycle basis to evade defence mechanisms.

An example of a functional core waiting for an enable signal from an authentication process can be taken. The input clock’s control being in the hands of the adversary would allow him to take a snapshot before the signal is set, clock the design, and then proceed with another snapshot. Comparing the snapshots would let the attacker distinguish what needs to be changed to modify the signals which would in turn modify the original bitstreams to permanently assert the enable signal, overturning the defence [40]. Alternatively, readback as a defence would be able to identify any tampering such as when there may be an ionising radiation attack.

Xilinx provides a less effective bitstream for disabling readback but when bitstream encryption is used, multiple, majority-voted, disabling registers activate to prevent readback [48]. By using bitstream encryption, lattice devices can also disable readback. They can be located by means of invasive attacks, but there is no evidence that this has occurred.

2.3.2.4 Side Channel Attacks

Side-channel attacks depend on the signals related to internal processes that are prone to measurements external to the device and accordingly disclose secret data or modes of operation by manipulating the implementation rather than the algorithmic construction. Preventing SCA is challenging because of the isolation of internal operations of integrated circuits from their environment. The energy they consume and release, when interacting with other devices is that of electromagnetic and heat radiation types [49].

There are three types of side-channel attacks and their relevance to FPGAs are explained as follows:

2.3.2.4.1 Power Mapping and Analysis

By analysing the current consumption patterns of integrated circuits, information about specific data can be determined – the most sought after information being the key in a cryptographic operation.

The researchers in [50] introduced two types of power analysis - simple (SPA) and differential (DPA). The former allowing the attacker direct search power traces for patterns such as algorithmic sequences, conditional branches, multiplication, and exponentiation, that allow the inference of key material. While the latter compares acquired traces with a statistical power consumption model that targets device and specific implementation. Previously acquired knowledge or analysis of the device led to the development of this model which is then enhanced by many recorded samples of controlled operations, by processing known plaintexts with known keys. The attacker can work out key material even if the implementation details are not explicit – it controls single bit changes in the encryption process.

In an attacker-controlled environment, power analysis is essential to understand the vulnerabilities of FPGAs. With modern FPGAs working at over 500 MHz, the requisite measurement equipment is not insignificant [51]. There is a requirement of more advanced methods than the reliance on outdated small resistor architectures. It might not be possible to reduce the operating frequency because of detection circuits. In FPGAs, clock managers (such as a "Digital Clock Manager") are set to a particular frequency range. The attacker must separate the signal of the FPGA from the surrounding devices that contribute noise through the shared ground and power supply.

Detection circuits for clock and temperature tampering can be classed as countermeasures that disable attackers to interrupt the clock's frequency [50]. It would be fair to assume that attackers would have to face some challenges before proceeding to capture power traces and analysing them.

2.3.2.4.2 EM Emanation Analysis

It is when internal operations take effect that the movement of charge leads to the production of electromagnetic fields on circuits, upon which side-channel attacks rely. Tuned antennas are capable of picking up these fields outside of the device, without the need to remove its packaging. Proper setup of EMA attacks would make them more efficient and allow them to create better signal-to-noise ratios, making them superior over power analysis [52]. Moreover, the advantage of electromagnetic attacks over power analysis lies in the fact that they can be localised to a specific part of the chip where the wanted function would be occurring. This can then be implemented in the device's initial setting.

2.3.2.4.3 Timing Analysis

There is a possibility that leakage of information may occur if data-processing operations depend on secret materials such as a key. Conditional branching, memory access, and algorithmic operations tend to depend on cryptographic function implementations. A good amount of key bits can be attained when their timing signatures are analysed. When passwords are checked one character at a time, stopping on the first match, it can mean that a timing attack occurred [53]. The attacker can figure out the password with only a few tries once he has determined different processing times.

A way to prevent the leak of information through time processing could be to ensure that sensitive operations have the same amount of clock cycles, having randomised operations, or by utilising memory blocks to store data [53]. Certainly, what the device’s pin exposes should always be monitored for time-related leaks.

2.3.2.5 Radiation-Induced Threats – Single Event Upset (SEU)

Single event upsets (SEU) are the errors that are induced in integrated circuits through radiations. Basically, a stream of electron-hole pairs is created as the charged particles lose their energy whilst ionizing the medium they travel through [54]. SEUs in CMOS devices are created by atmospheric and ambient ionizing radiation that include neutrons, protons and heavy ions. Alpha particles are also emitted from materials used for integrated circuit packaging. A transient pulse known as “single transient effect” may be caused by SEU and in turn cause delay faults. Furthermore, memory bit may flip state, and with lowering probabilities it is possible that multi-bit upsets occur. SEU flips are known as “soft errors” as they can be fixed by being written over or power-cycling.

A change to the purpose of the device is caused by a flip in a used configuration cell within FPGAs. There are ways to detect and correct SEUs which include scanning the configuration cells and comparing their CRC or Hamming syndrome to the initial ones. Another solution can be triple modular redundancy (TMR), in which all logic is triplicated and radiation causes majority voters to establish logic faults [31]. The most common way this is utilised is in space applications because the mean time between function and failure is very low, accounting for the cost. The bitstream can be read back and used to reprogram if it were to be verified.

2.3.2.6 Hardware Trojans

A hardware Trojan is defined as “a malicious, intentional modification of a circuit design that results in undesired behaviour when the circuit is deployed” [55]. For example, adding logic that blocks resources such as memory, or even granting access to limited-access data. Testing is a way to fight against this. That being said, a special code could trigger the Trojan when the device is in use. By knowing the details of the device, a skilled attacker could target an input not normally targeted. This would mean that the chances of being detected during normal testing would lower. Many physical attacks could cause triggers. An example is getting direct access to the I/O pins. Electromagnetic radiation or thermal energy are less invasive but more complicated triggers. **Our research on 'Design for Prognostics and Security in FPGAs' revolves around this highly potential threat that could cripple the existing as well as the future computation systems when the era of autonomous, AI-based, IoT, and Industry 4.0 would be all-prevalent**. A concise account of hardware Trojan salient is presented in Chapter-3.

2.3.2.7 Kill Switch

The operability of hardware as well as the software sustainability are highly affected by the malicious exploitation of an entity called the 'kill switch'. It stops the chip from working at all. It does this through creating an open connection by thinning key wires in order to destroy a section of a wire by electromigration [56].

2.3.2.8 The Backdoor

Backdoor is another peculiar type of hardware Trojan. This includes functionality in a circuit where access to the system is granted to either stop or cause functionality. This can be exemplified when an encryption core is disabled without alerting the user, making detection more difficult [57]. A kill switch, on the other hand, disrupts the chip completely.

Although these attacks seem far-fetched, there have been many occurrences of hardware that has been modified in a malicious manner. Dating back to the Cold War, to be specific, Russians used these methods for surveillance purposes . They were sabotaged by the US who tinkered with oil pipeline control software and allowed the Russians to steal it. In retaliation, the Russians sabotaged typewriters on the way to the US, by adding keyloggers [57].

Nowadays, it is assumed that a kill switch was developed into a microprocessor and used to disable Syrian radars from picking up attacks from Israel. Although a theory at present, it can be a reality when considering the way hardware Trojans operate. Indeed, King et al. [58] have demonstrated a number of attacks can be possible by simply adding logic to Leon processor. Moreover, Agrawal et al. [59] have also suggested that keys can be leaked by adding 400 logic gates to a public key encryption circuit.

2.3.3 Analysis

The realm of security in FPGAs reveals the nature of threats and attacks that surround and affect the dependability and security attributes of the FPGAs [60] and the critical applications implemented therein. These attributes are shown in Figure 2-10. It can be seen that the attribute of reliability is also integral to FPGA security besides the dependability and is, in turn, impacted by the triad of confidentiality, integrity, and availability (CIA). This implies that whenever the security of an FPGA is compromised, it is basically the attribute of FPGA reliability that is getting affected - with repercussions ranging from harm to humans and machinery breakdown to systems' malfunction and the manufacturers' reputation.

The nature of hardware attacks mentioned above is likely to change, getting more complicated and complex, with downscaling of FPGA technologies. In that case, the fragmented approach towards solutions/countermeasures for hardware threats and attacks would not be effective. We have already seen recent surge in hardware vulnerabilities with 'Spectre and Meltdown' [61], [62] and Amazon's FPGA cloud sharing platforms [63], [64]. Under such circumstances, it is indispensable to build and adopt an integrated approach so as to enable security and prognostics/health estimation schemes for an effective and enduring defence against known and unknown vulnerabilities.

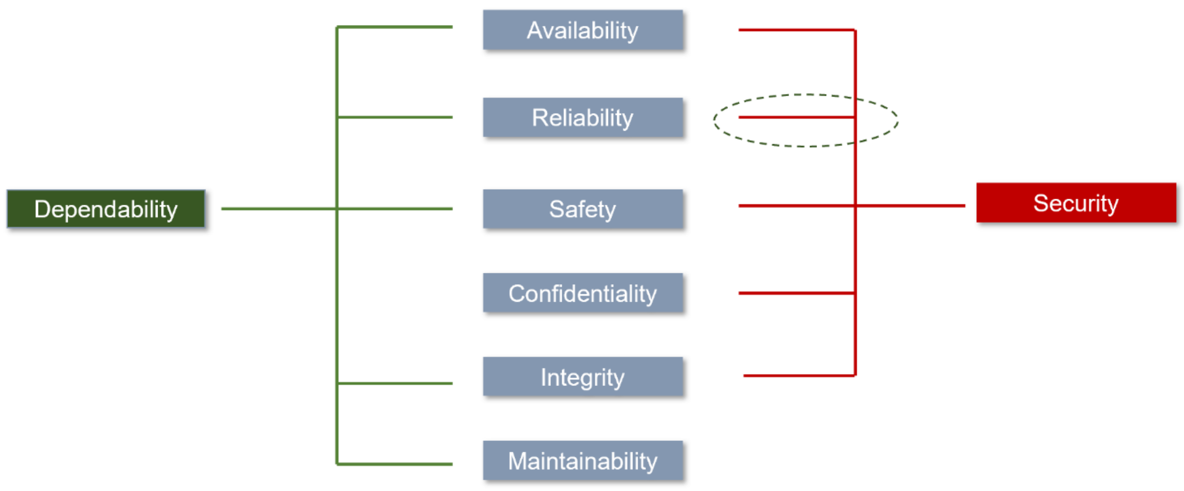


Figure 2-10 Dependability and Security Attributes of an FPGA [60].

It is not viable to prognosticate FPGA health with FPGA degradation models that are deficient not only in optimal reliability assessments but also completely devoid of the security concerns. As a result, we firmly believe that some previous and more recent cyberattacks had their roots in the hardware abstraction levels (and they took place because the FPGA designs - both pre and post manufacturing- lacked the integrated prognostics/reliability and security approach). Based on the above discussions and the outlining of different efforts, we believe, that one of the most important reasons for not being able to control and defy cyberattacks on FPGA based applications, in particular, is the fragmented approach towards FPGA health management. This implies that cybersecurity solution lies in the hardware security. It also highlights that the design for testability approaches reliability evaluation not only without prognostics element but also misses the security facet.

2.4 Integrated FPGA Health Management (IFHM) Framework

In consonance with the analysis of the realms of reliability and security in Sections 2.2 and 2.3, we have deduced that:

1. The FPGA reliability and security dimensions are inter-dependent - not two separate entities. When evaluated separately, the appreciation of hardware threats, vulnerabilities, and the impact of attacks on FPGA reliability may not be viable.

2. The existing reliability/degradation modelling and security design solutions may provide an optimum operational and functional FPGA evaluation in isolation, however, they are not attuned with an integrated approach for the management of FPGA health.

3. The individualistic approach toward FPGA assessment (reliability and security) needs to be enhanced to a composite and holistic FPGA health regimen to fight and guard against the upcoming technological challenges in a robust manner.

4. There is a need for a high-level framework that provides a roadmap and guidelines to achieve integrated FPGA health management, that encompasses the facets of reliability, security, and the prognostics.

2.4.1 The Framework

Above in perspective, it is evident that the existing individualistic approach toward FPGA health management does not consider the essential elements of reliability, prognostics and security collectively. This has resulted in fragmented solutions that do not reflect the true state of the operational condition of an FPGA. A high-level framework, called **'Integrated FPGA Health Management – (IFHM)'** framework has, therefore, been devised, as shown in Figure 2-11. This framework provides a guidance for the FPGA researchers, design and manufacturing engineers, and expert end-users in establishing the relationship between 'degradation/failure mechanism' and 'hardware threat/attack', determining 'failure precursor', constructing and optimizing the experimental set-up, defining test conditions and estimating the health of an FPGA in a composite manner.

As can be seen in Figure 2-11, there are three main functional levels namely, ***Resourcer***, ***Conjoiner*** and ***Unifier***. At the ***Resourcer*** functional level, the elements

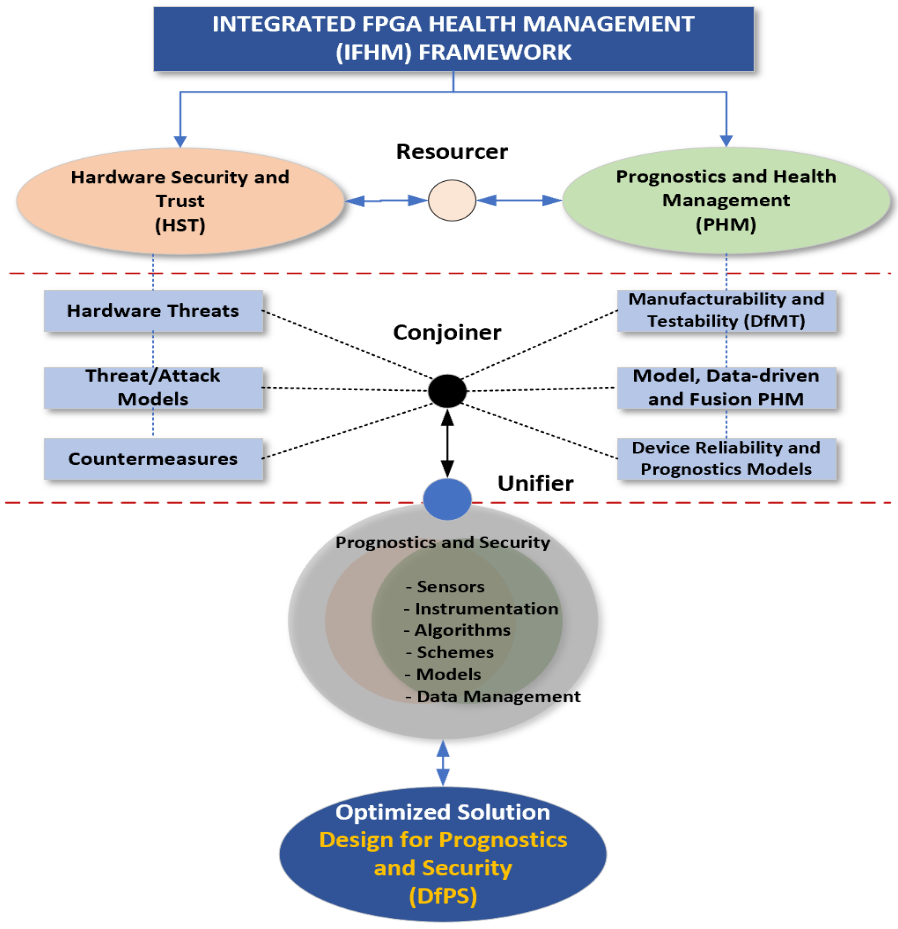

INTEGRATED FPGA HEALTH MANAGEMENT (IFHM) FRAMEWORK
Resourcer
Hardware Security and Trust (HST)
Prognostics and Health Management (PHM)
Hardware Threats
Threat/Attack Models
Countermeasures
Conjoiner
Manufacturability and Testability (DfMT)
Model, Data-driven and Fusion PHM
Device Reliability and Prognostics Models
Unifier
Prognostics and Security
- Sensors
- Instrumentation
- Algorithms
- Schemes
- Models
- Data Management
Optimized Solution
Design for Prognostics and Security (DfPS)

Figure 2-11 An integrated approach towards Reliability, Prognostics, and Security in FPGAs to bolster FPGA health for high-end Computational Systems.

under each of the two cardinals of Hardware Security and Trust (HST) and Prognostics and Health Management (PHM) are explored to develop a comprehensive set of desired information, which is then analysed and combined at the ***Conjoiner*** functional level into a focused set/library of interdependencies between FPGA reliability/prognostics and security. These interdependencies are then unified at the ***Unifier*** functional level to determine an optimized solution for the '***Design for Prognostics and Security***' by designing sensors, devising algorithms, developing models and undertaking implementation tests, followed by verification and validation.

2.4.1.1 The Resourcer

The Resourcer is a repository function comprising two areas of knowledge and information i.e., Hardware Security and Trust (HST) and Prognostics and Health Management (PHM). They are termed as 'Cardinals' because of being the principal sources of the requisite knowledge. It builds a comprehensive list/database of hardware threats/attacks, FPGA vulnerabilities, threat/attack models, and countermeasures (including detection, mitigation, and prevention methods/techniques) for the HST cardinal. The PHM cardinal, on the other hand, is built on reliability and degradation modelling, the various schemes of design for testability (DfT) and manufacturability, PHM methods, and prognostics/health estimation techniques for electronic devices. This wide range of two knowledge domains is then compiled into a Resourcer database, which at present, is constructed on MS Excel. The information contained in the realms of reliability and security provides a segment of the Resourcer repository.

2.4.1.2 The Conjoiner

The Conjoiner is an analysis function that builds upon the Resourcer database, which is classified into three elements (scalable) for the two cardinals each, as shown in Figure 2-11. This function forms the critical data analysis level where, based upon the specific threat assessment/evaluation requirement and the corresponding degradation mechanism, relevant data is scrutinised, and quantitatively compared with host of related models. The analysed data results in an optimised set/library of interdependencies between the reliability and security in the form of graphical data, mapping diagrams, and data flows.

2.4.1.3 The Unifier

The Unifier is a decision function that fetches the required set/library of analysed data, and accordingly helps design, implement, test, verify and validate the hard and soft components of the final optimised 'Design for Prognostics and Security'. It may comprise an on-chip parametric sensor for the detection of parametric variations due to the presence of malicious circuitry (hardware Trojan), a comprehensive FPGA security scheme, and a prognostics/health estimation process. The main outputs of this function may include new FPGA reliability/degradation models, optimal anomaly detection and mitigation algorithms, an experimental test-rig (comprising hard and soft components like test equipment, electronic design kits etc.), accurate measurements, and optimised verification and validation of the developed schemes.

2.4.2 IFHM Framework Workflow

In this sub-section, we present an overview of how the IFHM framework is used to build FPGA Health scheme consisting of the security and prognostics elements. It expands on the Cardinals and their corresponding elements that our IFHM framework covers. The high-level selection of an '**On-Chip Digital Sensor**' is used to explain the workflow of IFHM framework.

The researcher will first give the specific FPGA degradation mechanism information, for example, '**BTI**' degradation mechanism, as an input to the Resourcer function of the framework. The Resourcer function will use this information to scan and track the PHM cardinal through its elements for the relevant BTI data and then correlates it with the HST cardinal elements to fetch the relevant list of hardware threats, attacks, and models for the researcher. The function will also highlight the most optimal relevance/outcome of all the HST elements. It will be, however, up to the researcher to select the relevant HST and PHM elements before stepping onto the next functional level i.e. The Conjoiner. One possible selection could be the **CMOS parametric variation-based hardware Trojan**.

The Resourcer data, as mentioned above, is fed into the Conjoiner function, which will then analyse it against the FPGA vulnerabilities (within different technology nodes such as, high junction temperatures, design for testability and de-bug etc.), the faults and defects associated with it (intermittent delays, transient faults etc.), the reliability and performance impact of the selected degradation mechanism under the influence of the selected category of hardware Trojan/malicious anomaly (frequency and delay degradation, device ageing, functional failures, exponential increase in junction temperature, power consumption etc.), and the statistical data on various relevant FPGA degradation models. The researcher, with this analytical information available, chooses the most effective set of solutions. For example, for the BTI degradation mechanism and parametric variation-based hardware Trojan category, the researcher may opt to construct the FPGA security and prognostics/health estimation scheme around the variation of threshold voltage in P/NFETs with N/PBTI.

With the above set of information inputted to the Unifier function, all the relevant and recommended solutions with respect to monitoring, detection, mitigation, and prevention algorithms, different on-chip sensor options (with performance metrics), security schemes, and prognostics/health estimation techniques (ML-based) are made available to the researcher to select and optimise the most viable FPGA Health scheme that provides a composite solution. For example, in our case 'the Design for Prognostics and Security in FPGAs' is an integrated FPGA health management approach which begins with the design and implementation of a novel sensor (Chapter-4), solidifies it to an FPGA security scheme (Chapter-5), and then employs a Kernel-based Machine Learning technique to estimate FPGA health (Chapter-6).

While this high-level IFHM framework may be a logical and necessary step in assisting and guiding the researcher, manufacturer, and the expert end user in designing secure and reliable FPGAs, it would not eliminate the need for the FPGA health (security and reliability) assessment expert. This IFHM framework is intended to be an automated framework, integrated with the traditional FPGA/ASIC design flow so that the FPGA health evaluation can be made as an inherent segment of the design and implementation process.

2.5 Summary

The evolutionary transformation of FPGAs into Systems-on-Chip (SoC) and more advanced platforms like Advanced Compute Acceleration Platform (ACAP) has led to their widespread applications across all industries. These include several sensitive applications, like national infrastructures consisting of power grids, network routers and data-centres, medical equipment, transportation comprising airplanes, spacecrafts, and autonomous vehicles, defence systems, Industry 4.0, etc. An incisive look into FPGA architecture, its associated structural and functional strengths as well as its vulnerabilities to ever-growing reliability and security issues have been the focus of this chapter. Most importantly, based on the detailed account of the realms of FPGA reliability and security, a high-level Integrated FPGA Health Management (IFHM) framework has been presented as a guideline for the VLSI design and manufacturing community (including researchers and expert end-users) to develop highly optimised FPGA security and prognostics schemes by adopting integrated approach. The subsequent automation of this framework and integration with electronic design automation (EDA) tools would be highly useful.

The next chapter (Chapter-3) provides a brief account of hardware Trojans as a potent hardware threat for modern FPGAs, outlines their taxonomy, and presents several countermeasures.

REFERENCES

[1] Grand View Research, "*Field-Programmable Gate Array Market – Cruising Ahead,"* [Online]. Available at: https://www.grandviewresearch.com/blog/field-programmable-gate-array-fpga-market-size-share. [Accessed : July 2020].

[2] T. Huffmire, C. Irvine, T. D. Nguyen, T. Levin, R. Kastner, and T. Sherwood, *Handbook of FPGA Design Security*. Springer, 2010.

[3] K. Compton and S. Hauck, Reconfigurable computing: a survey of systems and software. *ACM Comput. Surv*. 34, 2, pp 171–210, 2002.

[4] Andre DeHon, "Comparing computing machines," Proc. SPIE 3526, Configurable Computing: Technology and Applications,1998.

[5] J. Singh and B. Raj, "SRAM Cells for Embedded Systems," in *Embedded Systems - Theory and Design Methodolgy*, no. June 2014, 2012.

[6] D. Kissler, F. Hannig, A. Kupriyanov and J. Teich, "Hardware Cost Analysis for Weakly Programmable Processor Arrays," *2006 International Symposium on System-on-Chip*, Tampere, pp. 1-4, 2006.

[7] An Chen, "*Beyond-CMOS technology roadmap"*, Emerging Research Devices (ERD), ITRS, 2015.

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

[9] Bhunia, Swarup, Tehranipoor, Mark M. (Eds.), *The Hardware Trojan War-Attacks, Myths, and Defenses*, Springer International Publishing, DOI 10.1007/978-3-319-68511-3, 2018.

[10] Bhunia, Swarup, Tehranipoor, Mark M. (Eds.), *Hardware Security – A Hands-on Learning Approach*, Elsevier Inc. ISBN: 978-0-12-812477-2, 2019.

[11] Y. Shiyanovskii, F. Wolff, A. Rajendran, C. Papachristou, D. Weyer and W. Clay, "Process reliability based trojans through NBTI and HCI effects," *2010 NASA/ESA Conference on Adaptive Hardware and Systems*, Anaheim, CA, pp. 215-222, 2010.

[12] A. Amouri and M. Tahoori, “A Low-Cost Sensor for Aging and Late Transitions Detection In Modern FPGAs,” *2011 21st Int. Conf. F. Program. Log. Appl.*, pp. 329–335, 2011.

[13] D. Patra *et al.*, “Adaptive accelerated aging for 28 nm HKMG technology,” *Microelectron. Reliab.*, vol. 80, no. December 2017, pp. 149–154, 2018.

[14] P. Mangalagiri, S. Bae, R. Krishnan, Y. Xie, and V. Narayanan, “Thermal-aware reliability analysis for platform FPGAs,” *IEEE/ACM Int. Conf. Comput. Des. Dig. Tech. Pap. ICCAD*, pp. 722–727, 2008.

[15] A. Kerber, S. Cimino, F. Guarin, and T. Nigam, “Assessing device reliability margin in scaled CMOS technologies using ring oscillator circuits,” *2017 IEEE Electron Devices Technol. Manuf. Conf. EDTM 2017 - Proc.*, vol. 1, pp. 28–30, 2017.

[16] Campos-Cruz, A.; Espinosa-Flores-Verdad, G.; Torres-Jacome, A.; Tlelo-Cuautle, E. On the Prediction of the Threshold Voltage Degradation in CMOS Technology Due to Bias-Temperature Instability. *Electronics 7*, 427, 2018.

[17] Xilinx, 7 Series FPGAs Data Sheet: Overview, Accessed on: July 10, 2020.

[Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.

[18] Mohanty, Saraju P. Srivastava, Ashok. *Nano-CMOS and Post-CMOS Electronics - Devices and Modelling, Substrate and Gate Dielectrics.* Institution of Engineering and Technology, Volume 1, 2016.

[19] Mark T. Bohr, Robert S. Chau, Tahir Ghani and Kaizad Mistry, "The High-K Solution", Accessed on: July 10, 2020. [Online]. Available: https://spectrum.ieee.org/semiconductors/design/the-highk-solution.

[20] Ethan A. Scott, A John T. Gaskins, A Sean W. King, A Patrick E. Hopkins, "Thermal conductivity and thermal boundary resistance of atomic layer deposited high-k dielectric aluminum oxide, hafnium oxide, and titanium oxide thin films on silicon." APL Materials. 6, 058302. 2018.

[21] I. Agbo *et al.*, “Integral impact of BTI and voltage temperature variation on SRAM sense amplifier,” *Proc. IEEE VLSI Test Symp.*, vol. 2015-Janua, 2015.

[22] B. P. Linder, J. J. Kim, R. Rao, K. Jenkins, and A. Bansal, “Separating NBTI and PBTI effects on the degradation of ring oscillator frequency,” *IEEE Int. Integr. Reliab. Work. Final Rep.*, vol. 3, pp. 1–6, 2011.

[23] S. Zafar *et al.*, “A Comparative Study of NBTI and PBTI (Charge Trapping) in SiO 2 / HfO 2 Stacks with FUSI , TiN , Re Gates,” *2006 Symp. VLSI Technol. 2006. Dig. Tech. Pap.*, vol. 9298, no. 2005, pp. 23–25, 2006.

[24] Intel. Corporation, “Reliability report 1H 2017.” Accessed on: June, 2019. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/rr/rr.pdf.

[25] Xilinx Corporation, “Device Reliability Report,” *UG116 (v10.1)*, vol. 116, pp. 1–104, 2014.

[26] A. Bensoussan, “Microelectronic reliability models for more than moore nanotechnology products,” *Facta Univ. - Ser. Electron. Energ.*, vol. 30, no. 1, pp. 1–25, 2017.

[27] M. Naouss and F. Marc, “FPGA LUT delay degradation due to HCI: Experiment and simulation results,” *Microelectron. Reliab.*, vol. 64, pp. 31–35, 2016.

[28] B. Khaleghi, B. Omidi, H. Amrouch, J. Henkel and H. Asadi, "Estimating and Mitigating Aging Effects in Routing Network of FPGAs," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 3, pp. 651-664, March 2019.

[29] L. Prodan, M. Udrescu, and O. Boncalo, “Design for Dependability in Emerging Technologies,” *J. Emerg. Technol. Comput. Syst*, vol. 3, no. 2, pp. 1–24, 2007.

[30] M. Samie, G. Dragffy, A. Popescu, T. Pipe, and J. Kiely, “Prokaryotic Bio-Inspired System,” *2009 NASA/ESA Conf. Adapt. Hardw. Syst.*, pp. 171–178, 2009.

[31] IEEE Standards Association, I*EEE 1856-2017-IEEE Standard Framework for Prognostics and Health Management of Electronic Systems*, RS/SC -IEEE Reliability, 2017.

[32] M. Lipp *et al.*, "Meltdown: Reading Kernel Memory from User Space," Communications of the ACM, Vol. 63 No. 6, pp. 46-56, June 2020.

[33] P. Kocher *et al.*, "Spectre Attacks: Exploiting Speculative Execution," *IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, pp. 1-19, 2019.

[34] S. P. Skorobogatov, “Semi-invasive attacks-a new approach to hardware security analysis,” *Tech. report, Univ. Cambridge, Comput. Lab.*, no. 630, p. 144, 2005.

[35] M. Tehranipoor, Wang**,** Cliff (Eds.), *Introduction to Hardware Security and Trust.* Springer-Verlag New York, 2012.

[36] R. Kastner and T. Huffmire, “Threats and Challenges in Reconfigurable Hardware Security.” Proceedings of the 2008 International Conference on Engineering of Reconfigurable Systems & Algorithms, ERSA 2008, Las Vegas, Nevada, USA, July 14-17, 2008.

[37] S. M. Trimberger and J. J. Moore, “FPGA security: Motivations, features, and

applications," *Proc. IEEE*, vol. 102, no. 8, pp. 1248–1265, 2014.

[38] Xilinx Inc., "WP365(v1.2): Solving Today's Design Security Concerns," *Xilinx, Inc.*, vol. 365, pp. 1–14, 2012.

[39] S. M. Trimberger, "Three Ages of FPGAs : A Retrospective on the First Thirty Years of FPGA Technology," *Proc. IEEE*, vol. 103, no. 3, pp. 318–331, 2015.

[40] S. Drimer, "Volatile FPGA design security – a survey," *Univ. Cambridge*, pp. 1–51, 2008.

[41] PYK Cheung, "Digital Design with FPGA and Verilog," Imperial College London V4.3, 7 Nov 2017.

[42] U. Guin, K. Huang, D. Dimase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. IEEE*, vol. 102, no. 8, 2014.

[43] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, "Hardware security: Threat models and metrics," *IEEE/ACM Int. Conf. Comput. Des. Dig. Tech. Pap. ICCAD*, pp. 819–823, 2013.

[44] P. Swierczynski, M. Fyrbiak, P. Koppe, A. Moradi, and C. Paar, "Interdiction in practice—Hardware Trojan against a high-security USB flash drive," *J. Cryptogr. Eng.*, vol. 7, no. 3, pp. 199–211, 2017.

[45] P. Swierczynski, M. Fyrbiak, P. Koppe, and C. Paar, "FPGA Trojans Through Detecting and Weakening of Cryptographic Primitives," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 34, no. 8, pp. 1236–1249, 2015.

[46] S. Skorobogatov and C. Woods, "In the blink of an eye : There goes your AES key," *IACR Cryptol. ePrint Arch.*, vol. 3, no. May, pp. 1–7, 2012.

[47] S. Drimer, "Security for volatile FPGAs," *Rapp. Tech. UCAM-CLTR-763, Univ. …*, no. 763, 2009.

[48] P. Mishra, S. Bhunia, and M. Tehranipoor, *Hardware IP security and trust*, Springer International Publishing, 2017.

[49] S. Bhunia and M. M. Tehranipoor, *The Hardware Trojan War*. Cham: Springer

International Publishing, 2018.

[50] C. Rooney, A. Seeam, and X. Bellekens, “Creation and Detection of HardwareTrojans Using Non-Invasive Off-The-Shelf Technologies,” *Electronics*, vol. 7, no. 7, p. 124, 2018.

[51] Z. Lu, D. Li, H. Liu, M. Gong, and Z. Liu, “An anti-electromagnetic attack PUF based on a configurable ring oscillator for wireless sensor networks,” *Sensors (Switzerland)*, vol. 17, no. 9, 2017.

[52] H. Xue and S. Ren, “Hardware Trojan detection by timing measurement: Theory and implementation,” *Microelectronics J.*, vol. 77, no. May, pp. 16–25, 2018.

[53] L. Sterpone and L. Boragno, “A probe-based SEU detection method for SRAM-based FPGAs,” *Microelectron. Reliab.*, vol. 76–77, pp. 154–158, 2017.

[54] M. Tehranipoor and F. Koushanfar, “A survey of hardware trojan taxonomy and detection,” *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 10–25, 2010.

[55] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, “Hardware Trojans: Lessons learned after one decade of research,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 1, pp. 1–23, 2016.

[56] X. Guo, R. G. Dutta, Y. Jin, F. Farahmandi, and P. Mishra, “Pre-Silicon Security Verification and Validation : A Formal Perspective,” in *Proceedings of the 52nd Annual Design Automation Conference*, 2015.

[57] R. Kastner and T. Huffmire, “Threats and Challenges in Reconfigurable Hardware Security.” Proceedings of the 2008 International Conference on Engineering of Reconfigurable Systems & Algorithms, ERSA 2008, Las Vegas, Nevada, USA, July 14-17, 2008.

[58] Samuel T. King, Joseph Tucek, Anthony Cozzie, Chris Grier, Weihang Jiang, and Yuanyuan Zhou, Designing and implementing malicious hardware. In Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET'08). USENIX Association, USA, Article 5, 1–8, 2008.

[59] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, “Trojan detection using IC fingerprinting,” *Proc. - IEEE Symp. Secur. Priv.*, pp. 296–310,

2007.

[60] N. Tuptuk and S. Hailes, “Security of smart manufacturing systems,” *J. Manuf. Syst.*, vol. 47, no. April, pp. 93–106, 2018.

[61] J. Sanders, "Spectre and Meltdown Explained: A Comprehensive Guide for Professionals". Accessed on: January 15, 2020. [Online]. Available: https://www.techrepublic.com/article/spectre-and-meltdown-explained-a-comprehensive-guide-for-professionals/.

[62] J. Fruhlinger, "Spectre and Meltdown explained: What they are, how they work, what's at risk". Accessed on: January 15, 2020. [Online]. Available: https://www.csoonline.com/article/3247868/spectre-and-meltdown-explained-what-they-are-how-they-work-whats-at-risk.html.

[63] Jin, Chenglu, et al. "Security of Cloud FPGAs: A Survey." *arXiv preprint arXiv:*2005.04867, 2020.

[64] Gnad, D. R. E., Krautter, J., & Tahoori, M. B. Leaky Noise: New Side-Channel Attack Vectors in Mixed-Signal IoT Devices. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 305-339, 2019.

3 Understanding the Hardware Trojans in FPGAs

3.1 Introduction

Hardware Trojans are malicious modifications to the intended functionality of a hardware circuit [1-4]. These modifications (or tampering) are undesired and unknown to the hardware designer and can have devastating effects on the electronic system. Trojans have three key characteristics: malicious intention, evasion of detection, and rarity of activation [5]. The intent of a Trojan is always the same: perform an unintended action to compromise the confidentiality, integrity, or authentication of the underlying hardware.

This compromise may be in the form of a shortened operational lifetime of the hardware (e.g., 5 years instead of 20 years) or complete failure of the system upon the Trojan's activation. It may allow an attacker to gain unauthorized access into the hardware (i.e., remote access through a backdoor) or lead to leakage of information (e.g., cryptographic keys for secure data communication). Hardware Trojans may manifest from software Trojans inside of pirated software tool suites during the

Chapter 3
3.1 Introduction
3.2 Threat Model and Taxonomy
3.3 Trojans in FPGA Fabric
3.4 Trojans in FPGA Tool Chain
3.5 FPGA Attacks
3.6 Trojan Countermeasures
3.7 Summary

Figure 3-1 The Disposition of Chapter-3.

synthesis portion of the design flow or be inserted as a result of collusion between multiple parties at different stages of the hardware’s life cycle [6]. Trojans can also be designed with the sole intention to damage or destroy the brand reputation of a company, which may result in bankruptcy of the company and a competitive advantage for the adversary.

As mentioned earlier, the growing demand for power-efficient and high-performance integrated circuits (ICs) has created a surge in usage of FPGAs in the recent years. FPGAs are also available as cloud services [7], where one can create and run custom hardware designs on a remote FPGA in a server farm. Hence, exploring security issues associated with FPGA designs is critical.

This chapter explores the insertion of hardware Trojans into genuine designs targeted for FPGAs so as to understand their impact and evaluate different countermeasures. It goes without saying that such compromised designs may result in subpar performance, leakage of confidential information, and unauthorized and pernicious operations by an attacker. The use of compromised designs in critical infrastructures such as smart grids, nuclear power plants, medical prosthetic devices, and military equipment can be catastrophic. To explain the different classes of Trojans, this chapter uses Xilinx FPGA design flow. However, the same methodology can be extended to any FPGA and CAD tool vendors. The chapter is arranged as per Figure 3-1.

We first present the threat model and a taxonomy of FPGA Trojans in Section 3.2. Next, we focus on two broad categories of FPGA Trojans: Trojans in FPGA fabric in Section 3.3 and Trojans in FPGA tool chain in Section 3.4. An example of FPGA attack and a case study is presented in Section 3.5. Section 3.6 discusses the countermeasures that specifically target Trojans in FPGA bitstreams. Finally, Section 3.7 summarises the chapter.

3.2 Threat Model and Taxonomy

3.2.1 FPGA Design Flow

Figure 3-2 shows the high-level FPGA design flow. An FPGA designer designs the FPGA fabric. Fabless FPGA design houses send the layout of the FPGA fabric to a foundry for manufacturing. Many of these foundries are located typically offshore and are untrusted. Post-fabrication, the FPGAs are tested for defects and faults. FPGAs

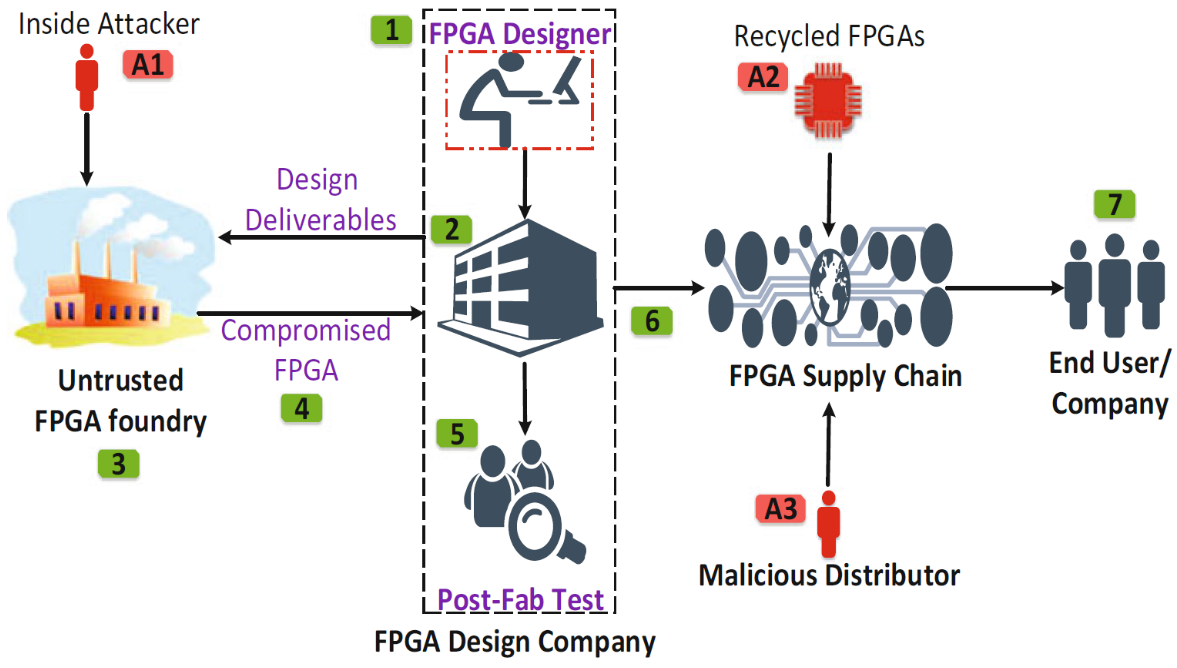


Figure 3-2 FPGA threat model: The attacker can insert hardware Trojans at the untrusted foundry (A1). A malicious distributor can reduce the reliability of an FPGA in the supply chain (A3), and even recycled FPGAs can be inserted into the FPGA supply chain (A2). Design Trojans can also enter through FPGA CAD tool flow.

are sold on the market. The end-user implements the target design on the FPGA. Converting a design described in a modelling language (VHDL or Verilog) into an FPGA-specific programming file involves multiple steps, as explained below:

- Synthesis involves the conversion of HDL into a logical netlist (similar to logic diagram or circuit).
- Implementation consists of translate and map processes, where the logical netlist gets converted and mapped to target device's physical primitives.
- Place and route (PAR) takes a mapped native circuit description (NCD) file, places and routes the design, and produces an NCD file to be used by the programmable file generator.
- In bit-file generation, the routed NCD is used to create a bit-file that can be programmed onto an FPGA.

3.2.2 Threat Model

In the quest to reduce the development cost of hardware/system, the silicon industry has inadvertently created a complex and extremely vulnerable supply chain shown in

Figure 3-2. An attacker can be present anywhere in the supply chain. The threat model, shown in Figure 3-2, involves:

3.2.2.1 Overproduction

An untrusted foundry that has access to the FPGA layout mask fabricates more number of FPGAs than requested or authorized by the design company. It can insert these FPGAs into the supply chain without the knowledge of FPGA design company. These FPGAs may not be properly tested and can introduce reliability issues. This results in either loss of revenue or reputation for the design company.

3.2.2.2 Recycling and remarking

FPGAs can be extracted from electronic waste, used FPGAs can be removed, and their package can be repainted and/or remarked. The die can also be removed from the packaging, repackaged, and remarked. These FPGAs are then reinserted into the

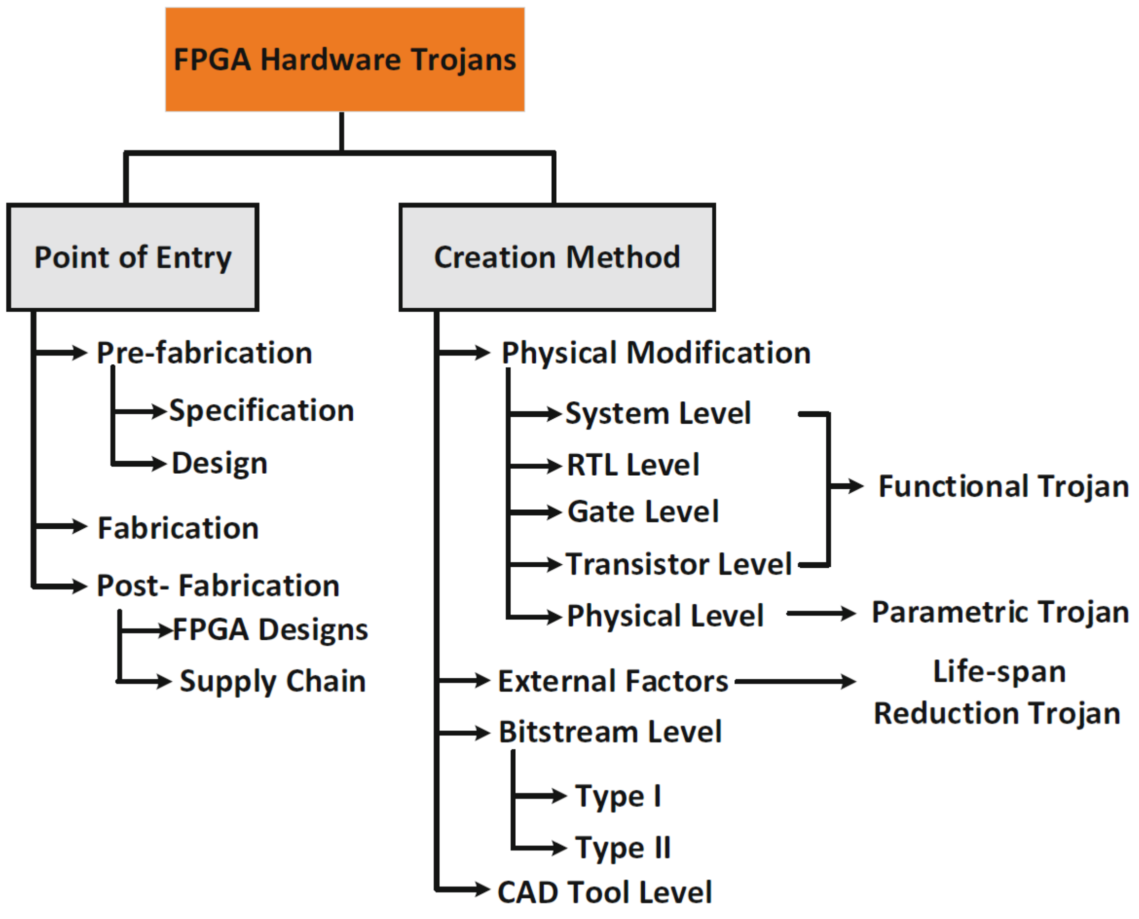


Figure 3-3 FPGA hardware Trojan taxonomy based on two primary attributes.

supply chain as genuine and new FPGAs. These FPGAs can be highly unreliable, are prone to defects, and typically lead to subpar performance.

3.2.2.3 Cloning and Piracy

It is an unauthorized reproduction of an FPGA by reverse engineering without the legal intellectual property (IP) rights to manufacture the FPGA. These FPGAs can also have malicious modifications.

Apart from these threats, FPGAs are also susceptible to insertion of Trojans, as shown in Figure 3-3.

3.2.3 FPGA Hardware Trojans Taxonomy

Malicious changes can be made at any phase of the FPGA design such as design, fabrication, packaging, and in the supply chain as shown in Figure 3-2. A taxonomy based on hardware Trojans' physical, activation, and functional characteristics have already been proposed [2, 3]. We classify Trojans based on the method of creation, activation, and point of entry into the FPGA fabric as shown in Figure 3-3. The definitions of most of the FPGA Trojans are similar to the IC Trojan taxonomy in [2, 3].

3.2.3.1 Point of Entry

Based on the point of entry of Trojans in FPGA, they can be classified as:

3.2.3.1.1 Prefabrication

It is the phase where the specification of systems such as functionality, size, power, delay, etc., is finalized. Trojan insertion in this step will result in alteration of design or constraints. For example, it could alter the timing of circuit or increase switching frequency of the circuit. A rogue employee can insert a malicious circuit, e.g., a backdoor, to take control of the chip at a later point in time when the FPGAs are deployed in the field. These manifest as FPGA fabric Trojans.

3.2.3.1.2 Fabrication

Here, a set of masks are designed to fabricate the digital circuit on a silicon wafer. Trojans can be added by a malicious attacker inside an untrusted foundry. These Trojans can be either functional or parametric. These are called as FPGA fabric Trojans.

3.2.3.1.3 Post Fabrication

In this phase, RTL/HDL designs are used to program an FPGA to achieve desired functionality. Trojans can be either inserted in RTL/HDL designs by a rogue employee or can also enter RTL/HDL designs from IPs from third-party IP providers. These are FPGA design Trojans. Additionally, even the reliability of an FPGA can be reduced by such type of Trojans.

3.2.3.2 Creation Method

Based on the creation method, Trojans can be classified as follows:

3.2.3.2.1 Functional Trojan

They are created by modifying the FPGA fabric. This includes addition/deletion of gates/transistors, modifying the RTL or layout without affecting the primary functionality of the FPGA fabric. It can enter during prefabrication phase by a rogue employee in the FPGA design company or during fabrication phase by a malicious insider at an untrusted foundry.

3.2.3.2.2 Parametric Trojan

They are created by modifying physical device parameters, such as thinning of wires, gate channel length variation, dopant level variation [8], transistor size variation, etc. It is always on and primarily created to reduce the reliability and lifespan of an FPGA.

3.2.3.2.3 Life-span Reduction Trojan (LRT)

It is the only class of Trojans that are not inserted in the hardware during or before fabrication. It is created by subjecting the FPGA with external factors, such as extreme temperatures, focused ion beams [9], etc. LRT accelerates aging of complete or part of FPGA fabric. It is typically created by a malicious distributor in the FPGA supply chain to reduce the reliability and, hence, reduce the lifespan of FPGAs.

3.2.3.2.4 Bitstream Trojan

It is inserted by modifying the FPGA bit-file itself. Bitstreams can be reverse engineered to identify the areas of FPGA occupied by the programmed logic, and malicious circuits can be inserted into it. If the malicious circuit does not disturb the original circuit, it is called Type-I bitstream Trojan. Type-II Trojans typically modify the original circuitry with respect to CLBs or other FPGA resources to perform malicious operations.

3.2.3.2.5 CAD Tool Trojan

They are FPGA design Trojans that exploit the CAD tool flow to insert the Trojans at various intermediate netlist formats. These Trojans can be inserted in a synthesized netlist and even in mapped or placed and routed netlists. Due to the lack of resources to understand the intermediate and typically proprietary formats, these Trojans can easily evade detection.

3.3 Trojans in FPGA Fabric

FPGA fabric Trojans are inserted into the FPGA silicon fabric. They can be inserted either during fabrication by a untrusted foundry or during the design phase of FPGA by a rogue employee in the FPGA design company. Functional fabric Trojans are characterized by addition/deletion of gates by the attacker to carry out malicious activities, whereas parametric fabric Trojans are created by changing device parameters/specification such as thinning of wires and weakening of transistors or flip-flops to reduce the reliability of the FPGA [10, 11]. In this section, we describe three Trojans that can be inserted into the FPGA fabric: Trojans that increase delay, create voltage fluctuations, and reduce lifetime.

3.3.1 Trojans That Increase Delay

The delay-based fabric Trojan is created by modifying interconnect connecting lookup tables (LUTs) across two configurable logic blocks (CLBs). The delay-based fabric Trojans correspond to change or perturbation in the physical layout of FPGA due to the addition of malicious elements. The assumption is that the silicon fabric of the FGPA is dense and highly utilized. An attacker needs to alter the FPGA silicon in order to add a Trojan. For example, when a Trojan is inserted in a CLB or routing switch matrix (RSM), it will perturb the physical layout of original fabric, thereby increasing the delay.

3.3.2 Trojans That Induce Voltage Fluctuations

This set of Trojans is implemented by adding simultaneous switching signals that utilize dense interconnect resources around a CLB. This corresponds to the addition of malicious elements without disturbing the genuine layout of FPGA fabric. This switching signal is connected to unused wires and programmable interconnect points (PIPs) in the tile where the target CLB is configured.

This Trojan increases switching activity that will increase dynamic power and therefore impacts the oscillation frequency. In our case, it is observed that voltage drop due to the Trojan switching activity impacts the sensor frequency.

3.3.3 Life-Span Reduction Trojan (LRT)

Life-span reduction Trojan (LRT) can be induced into an FPGA by artificially creating conditions that accelerate ageing of FPGA fabric. Key contributors for an FPGA ageing (or any IC) among several physical factors are negative-bias temperature instability (NBTI) and hot carrier injection (HCI). Both the factors lead to a shift of threshold voltage of the affected transistors, which manifest as increase in switching and path delays. This will subsequently lead to timing violations and wears out an FPGA faster. The threshold-voltage triggered hardware Trojan described in Chapter-5 is a novel example of this type of Trojan.

3.4 Trojans in FPGA Tool Chain

3.4.1 Trojan Insertion in FPGA Designs

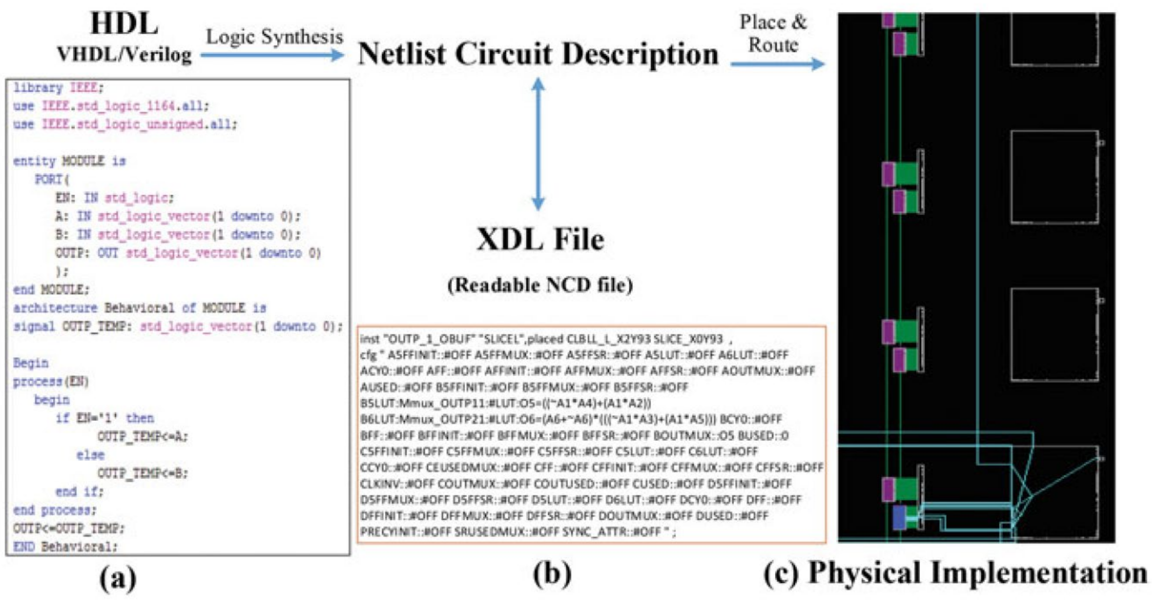
The goal of the attack tool is to decide where to place the Trojan for a given design. The placement of Trojan can be achieved with or without disturbing the original design mapping and routing. The latter requires considerable effort and access to multiple files from the FPGA design cycle.

To insert hardware Trojans in FPGA designs, an attacker may need to have knowledge of internal wires or logic and preferably where the design is physically placed on the FPGA. If the Trojan is conditionally activated based on the input or internal states, the attacker needs to tap into the required wires of the design and connect the Trojan activator circuit. The Trojan payload can be connected to the target elements by disconnecting the wires connecting to these elements and reconnecting with the output of the payload circuit. The original payload and Trojan payload can be connected using multiplexers, with the select line controlled by the activator circuit.

After the logic synthesis process, FPGA CAD tools typically rename and merge (after logic optimization) the internal wires connecting logic elements. An attacker needs to track the name changes in the design, to connect them with the Trojans. This can be achieved by converting the synthesized binary netlist (called NGC by Xilinx) to a readable Electronic Design Interchange Format (EDIF) file and Xilinx Design Language (XDL) file. Figure 3-4a shows the HDL code, and Figure 3-4b shows the corresponding XDL file obtained from routed netlist (NCD) after PAR, which describes how the HDL is mapped into LUTs. Additional information on the location of configurable logic block (CLB) and LUT is also present in the XDL file. Figure 3-4c shows how the CLB blocks are connected with each other to implement the functionality described in HDL. We can extract the locations and interconnections used by the original design from the NCD or XDL files.

3.4.2 Trojans in HDL

An attacker can insert the Trojan in the HDL design. In the HDL, the attacker can easily track the logic elements or states to be used as an activator and deliver the Trojan payload to the target. Inserting Trojan at this level is significantly easier for an attacker, as the wires and logic elements can be found from the behavioural or structural code.



```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity MODULE is
   PORT(
      EN: IN std_logic;
      A: IN std_logic_vector(1 downto 0);
      B: IN std_logic_vector(1 downto 0);
      OUTP: OUT std_logic_vector(1 downto 0)
      );
end MODULE;
architecture Behavioral of MODULE is
signal OUTP_TEMP: std_logic_vector(1 downto 0);

Begin
process(EN)
   begin
      if EN='1' then
            OUTP_TEMP<=A;
         else
            OUTP_TEMP<=B;
      end if;
end process;
OUTP<=OUTP_TEMP;
END Behavioral;
```

```
inst "OUTP_1_OBUF" "SLICEL",placed CLBLL_L_X2Y93 SLICE_X0Y93 ,
cfg " A5FFINIT::#OFF A5FFMUX::#OFF A5FFSR::#OFF A5LUT::#OFF A6LUT::#OFF
ACY0::#OFF AFF::#OFF AFFINIT::#OFF AFFMUX::#OFF AFFSR::#OFF AOUTMUX::#OFF
AUSED::#OFF B5FFINIT::#OFF B5FFMUX::#OFF B5FFSR::#OFF
B5LUT:Mmux_OUTP11:#LUT:O5=((~A1*A4)+(A1*A2))
B6LUT:Mmux_OUTP21:#LUT:O6=(A6+~A6)*(((~A1*A3)+(A1*A5))) BCY0::#OFF
BFF::#OFF BFFINIT::#OFF BFFMUX::#OFF BFFSR::#OFF BOUTMUX::O5 BUSED::0
C5FFINIT::#OFF C5FFMUX::#OFF C5FFSR::#OFF C5LUT::#OFF C6LUT::#OFF
CCY0::#OFF CEUSEDMUX::#OFF CFF::#OFF CFFINIT::#OFF CFFMUX::#OFF CFFSR::#OFF
CLKINV::#OFF COUTMUX::#OFF COUTUSED::#OFF CUSED::#OFF D5FFINIT::#OFF
D5FFMUX::#OFF D5FFSR::#OFF D5LUT::#OFF D6LUT::#OFF DCY0::#OFF DFF::#OFF
DFFINIT::#OFF DFFMUX::#OFF DFFSR::#OFF DOUTMUX::#OFF DUSED::#OFF
PRECYINIT::#OFF SRUSEDMUX::#OFF SYNC_ATTR::#OFF " ;
```

Figure 3-4 HDL to FPGA physical implementation. (a) Description of the design in HDL. (b) The configuration of a routed CLB described in XDL format. Only LUT B is used in this CLB. (c) The physical implementation of the design.

3.5 FPGA Attacks and a Case Study

In order to elucidate the significance of the growing hardware Trojan threat, we present a threat scenario related to the increased usage of FPGAs in a cloud platform. Consider a typical architecture of a cloud platform with FPGAs as shown in Fig. 3-5. FPGA boards are connected with the servers using PCIe (Peripheral Component Interconnect express) wires. PCIe wires are the de facto standard for establishing and maintaining communication between a server and the FPGA in commercial FPGA clouds [12]. The cloud service providers divide the programmable resources on an FPGA into two areas such that one is dedicated for implementing the shell, and the other for users to implement customized logic.

The shell includes PCIe modules, DRAM controllers, and control modules, to enable the communication with the servers and DRAM. Typically, the cloud provider's logic (shell) interacts with user logic via Advanced eXtensible Interface (AXI) protocols [13]. On the CPU side, the software development kit provides the application programming interfaces (APIs), so the users with little FPGA experiences can still interact with FPGAs easily [14]. In the modern commercial clouds like Amazon EC2 F1, an FPGA

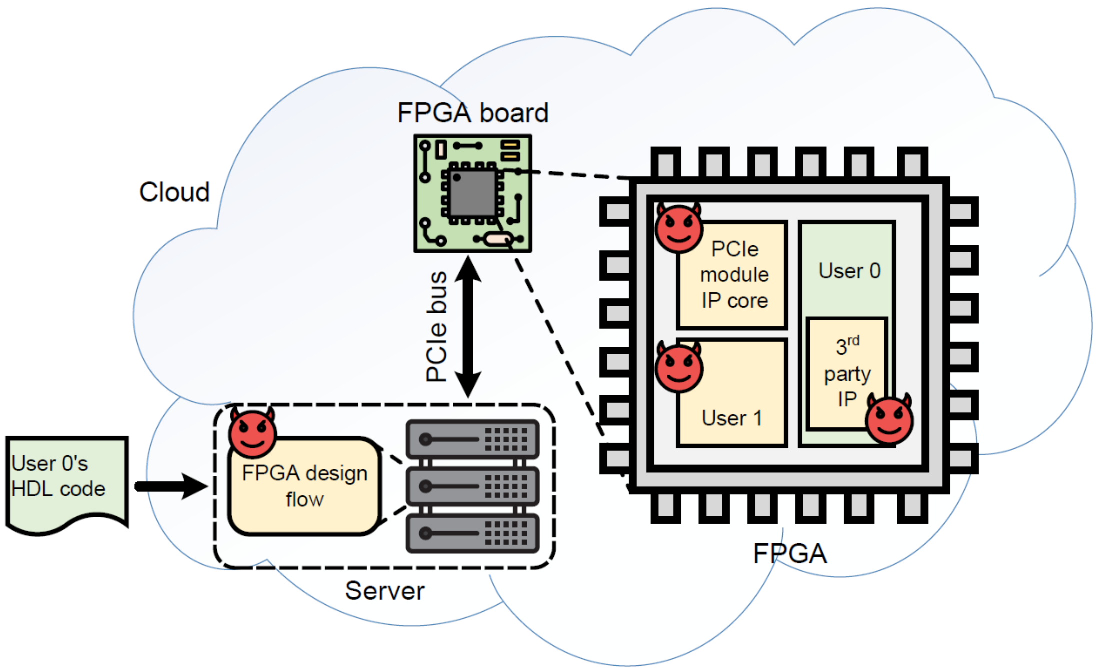


Figure 3-5 Architecture of an FPGA in the cloud. The four different threat models considered in this paper are (1) malicious cloud providers, (2) malicious co-tenants, (3) malicious IP providers, and (4) malicious FGPA toolchain. These are indicated in the figure by devil icons in the shell (PCIe module and IP core), user 1's logic, 3rd party IP core, and the FPGA design flow, respectively.

is not allowed to be shared by multiple users due to security concerns [15]. However, we envision that multi-tenant cloud FPGAs will be realized soon, as it is more cost-effective for both the cloud providers and the users to share resources.

3.5.1 Threat Scenario/Model and Associated Attacks

To understand the possible threats posed to the cloud FPGA users, we categorize the threat model into four types: (1) malicious cloud providers, (2) malicious cloud users/co-tenants, (3) malicious IP providers, and (4) malicious toolchains. Figure 3-5 illustrates where the threats reside in the architecture of an FPGA cloud.

3.5.1.1 Malicious Cloud Providers

In traditional threat models of cloud security, the cloud service providers are generally assumed to be untrustworthy, so a user needs to implement his/her security measures to protect him/herself in the clouds. Additionally, the users on the same cloud platform can be a threat to other users, too. However, a malicious cloud model is stricter than the malicious user model because a cloud provider has all the privileges to the platform, including physical access and full control of the computation resources.

3.5.1.1.1 Direct Sensitive Data Leakage

In a cloud without programmable hardware, all the computation and the data are contained in one container (virtual machine). Each container is isolated from another in the hypervisor layer. In the case of a cloud with programmable hardware attached, an attacker with system privilege can tamper with the logic or tap the communication between the FPGA fabric and the processor. This can enable him/her to steal the secret data. In current commercial FPGA-enabled clouds, the FPGA boards connect to the processors via the PCIe protocol. Thus, the cloud provider can intercept the communication between the FPGA boards and the processors with ease.

3.5.1.1.2 Intellectual Property Theft

The most common use of cloud FPGAs is to implement hardware accelerators for specific computation tasks. The IP of such an accelerator developed and owned by a developer should be protected. Since the developer hands over the bitstream files of the IP cores to the cloud providers, a malicious cloud provider can access the RTL design of the IP core. Bitstream reverse engineering techniques can enable this [16, 17]. Thus, a malicious provider can steal the design IP and replicate the accelerator on another FPGA.

3.5.1.1.3 Tampering with User Logic

A malicious cloud provider can access the user’s RTL design. So, during the integration of the user’s design with the shell in the cloud FPGA, the providers can introduce malicious modifications in the design. This security threat is also known as hardware Trojans that have been studied for decades [18]. On cloud FPGAs, the Trojans can leak sensitive information, which has been protected by other schemes in traditional cloud computing platforms. Also, the Trojans can sometimes be inserted automatically [19]. One of the future challenges is to provide a remote attestation feature which allows a remote user to verify the integrity and authenticity of his/her designs in a cloud FPGA. This feature might be similar to the remote attestation provided by Intel SGX [20].

3.5.1.2 Malicious Co-tenants

Besides the security threats from a malicious cloud provider, threats from malicious users/co-tenants need to be considered. The basic principle of cloud computing is that all the users can dynamically have a share of the large computation resource pool. Due to this, a victim user can be allocated close to a malicious user. Moreover, the victim and the malicious user might even share some computation resources. Although, in general, the computation resources used by different users are logically isolated, the computation resources are likely to be physically connected due to the shared hardware platform. Attackers can leverage such a shared hardware platform to perform a variety of attacks such as side-channel attacks, fault-injection attacks, and establishment of covert channels.

3.5.1.2.1 Side Channel Attacks

The attack methods that exfiltrate information, not leak able through standard digital output channels are called side-channel attacks. Power side-channel [21], timing side-channel [22], electromagnetic side-channel [23], and photonic-emission side channel [24] are a few examples of side-channels. An attacker must collect the side-channel information of victim devices in these attacks. Hence, researchers have believed for a long time that the side-channel attacks can be launched only by the attackers with physical access to the devices. However, the ability to program the hardware deployed in the cloud is similar to having physical access to the device. This allows the attackers to monitor the side-channel information remotely in the physical environment, as shown in Figure 3-6. The power consumption of a victim logic disturbs the power distribution network on the FPGA, and measuring this disturbance allows the attacker to estimate the power consumption of the victim. Remote power-based side-channel attacks have been demonstrated in the literature [25]. Moreover, crosstalk between FPGA long wires (a specific type of routing resource on FPGAs) can also serve as a method to leak information [26].

3.5.1.2.2 Fault-injection Attacks

In fault-injection attacks, an attacker injects faults in the execution process of a computation task. Thus, the device produces wrong outputs at the output ports. This problem can have severe implications in a cryptographic system. In such a system, faulty outputs can lead to a successful recovery of the secret key in the system [27]. Traditionally, an attacker injects faults by manipulating power or clock signals, or by electromagnetic pulses. These methods require physical access to the target device. However, using FPGAs shared with a victim, an attacker can build an on-chip fault injector and tamper with the computation of the victim.

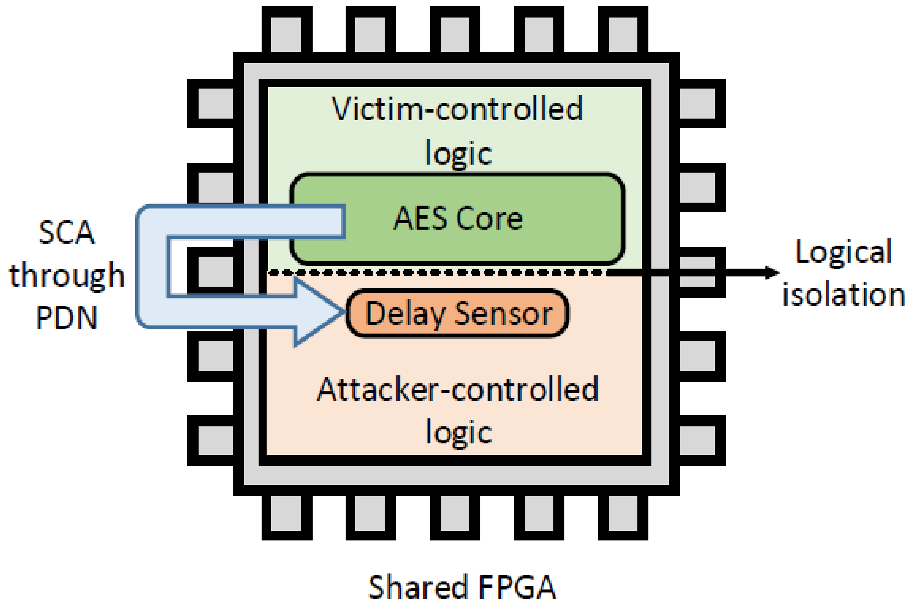


Figure 3-6 Remote power analysis attack for a multi-tenant FPGA. The side-channel analysis (SCA) is performed through the power distribution network (PDN) in spite of the logical isolation between the victim logic and the sensor [25].

3.5.1.2.3 Denial-of-Service Attacks

One property of concern for both the cloud providers and the users is the availability of the cloud platform. Denial-of-service (DoS) attackers target the availability of this platform. On an FPGA+CPU heterogeneous cloud, an attacker can launch a remote DoS attack on the FPGA [28]. By programming a malicious circuit that switches on and off frequently, a significant voltage drop is created on the FPGA, and the FPGA shuts down to protect itself. An FPGA shut down by voltage emergency requires manual power-cycling of the device.

3.5.1.2.4 Row-Hammer Attacks

Interestingly, in an FPGA+CPU heterogeneous system, the FPGA has a unique privilege to access the DRAM without being detected by any monitoring mechanism in the CPU. Also, the FPGA can bypass the cache in the processor and launch a row-hammer attack (i.e., flipping the bits in DRAM by repeated accesses) twice as fast as the traditional row-hammer attack launched by a CPU [29]. Consequently, the row-hammer from an FPGA to a DRAM can trigger four times as many bit-flips as the CPU initiated attacks. By exploiting this vulnerability, one can tamper with the data and possibly the control flow of the program in the system.

3.5.1.3 Malicious IP Providers

The modern hardware design process is very complicated and time consuming. Practitioners need to integrate 3rd-party intellectual property (3PIP) cores to speed up the development process. This gives attackers a leeway to introduce malicious IPs, and the IPs can be exploited later to leak information, e.g., via covert channels [30, 31]. This threat requires the attacker or the attacker's logic to be present in the proximity of the target FPGA fabric. Thus, the attacker can collect leaked information. So, either the cloud provider or a cloud co-tenant has to be malicious as well.

3.5.1.3.1 Power Covert Channels

The idea of voltage manipulations used in power side-channel attacks can be extended to establish covert channels on multi-tenant FPGAs. An example of this is the work done by Gnad et al. in [32]. They have demonstrated high-speed covert-channel (8MBit/s) communication. The transmitter of the covert channel uses ROs to generate measurable voltage spikes according to the secret data to be transmitted. The receiver, which is another tenant on the same FPGA chip, uses another set of

ROs to measure the voltage spikes. The attacker designs both the transmitter and the receiver. This enables the attacker to modulate the transmitted signal leading to robust communication, which can work in the presence of environmental noise introduced by other tenants on the same FPGA fabric.

Establishing such power covert channels can be challenging if the receiver and the transmitter are on separate dies. However, Giechaskiel et al. demonstrated such an attack on cloud FPGAs in [31]. They established a power covert channel on cloud FPGAs that are on separate dies. They use Xilinx UltraScale+ FPGAs for this. UltraScale+ FPGAs used by cloud providers like Amazon and Huawei have three distinct dies that are connected and powered through a silicon interposer. Thus, even though the receiver and the transmitter are on separate dies, they still share the same power supply through the silicon interposer. A successful covert channel, operating at more than 4.6Mbps with an accuracy of over 97.6%, is established in such a setup. Moreover, they showed that the channel is present for all combinations of the three dies as receiver and transmitter.

3.5.1.3.2 Cross talk in Long Wires

Crosstalk phenomenon in long wires can be exploited to launch covert channel communication as well [33]. The attacker is assumed to have a malicious IP core as a part of the victim logic. It is also assumed that the attacker’s logic is on the same FPGA fabric and is placed close to the victim’s logic. Since the adversary is the designer of the IP core, he/she can define the internal placement and routing of his/her blocks. Thus, the attacker can force his/her cores to use specific routing resources, in particular long wires. The attack exploits the phenomenon that the delay of FPGA long wires depends on the logical state of nearby wires. In particular, when the transmitter wire (the long wire in the victim design) carries a logic 1, the delay of the nearby receiving wire (the long wire in the attacker’s design) is lower than it would be if the transmitter wire carried a logic 0. An RO involving the receiver long wire can measure the delay of the receiver wire. This reveals the logic state of the nearby transmitter long wire. Thus, a covert-channel is created for attackers to leak sensitive information from a victim hardware design. This covert channel can work effectively, even in the presence of power and temperature fluctuations.

3.5.1.3.3 Thermal Covert Channel

Most of the covert channels in the literature require the designs of attackers and victims to be present on the same FPGA chip, i.e., a multi-tenant FPGA setup. However, cloud providers have not adopted the multi-tenant FPGA model yet. There exists a covert channel on the cloud FPGAs which does not require a multi-tenant setup. The covert channel described by Tian et al. in [34] is an example. It exploits the temporal sharing of a single FPGA. This channel can transmit data stealthily on a single-tenant cloud FPGA. The transmitter heats an FPGA by operating many ROs. Then, the transmitter turns off the ROs, leaves the cloud, and the receiver uses the same FPGA. The receiver can measure the temperature of that FPGA with ROs. This is possible because the frequency of an RO depends on the temperature of the FPGA. The bandwidth of such a thermal covert-channel depends on the number of FPGAs used simultaneously. A binary string can be transmitted and received by the temporal sharing of four cloud FPGAs simultaneously. This covert channel was demonstrated on the cloud FPGAs in Texas Advanced Computing Centre in [34].

3.5.1.4 Malicious FPGA Tools

Adversaries can reverse-engineer commercial FPGA design tools and embed malicious functionalities in the toolchain. This way, malicious tools can alter the compiled hardware design. Under this threat model, the adversary can inject Trojans in a design. This maliciously-altered design behaves functionally and formally equivalent to the original design throughout the design flow until the tool writes the design as a bitstream configuration file [35].

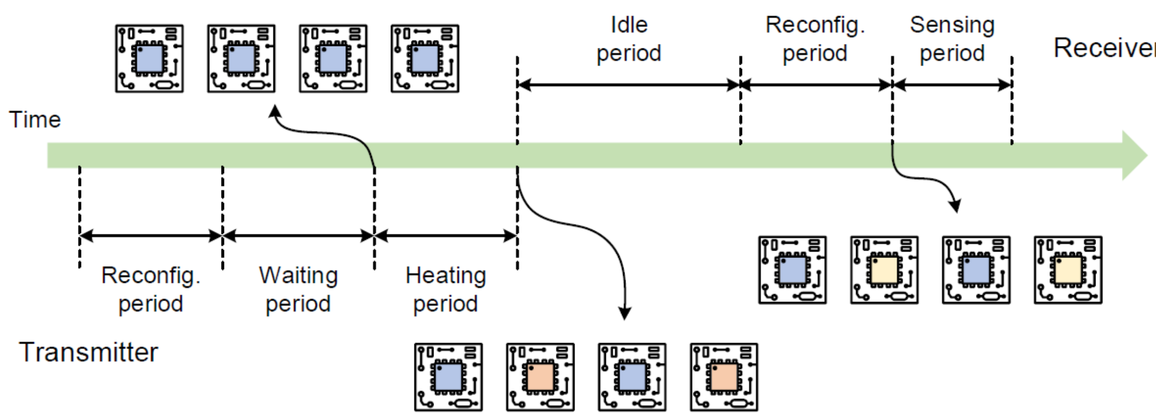


Figure 3-7 Establishment of thermal covert channel on cloud FPGA [34]. The transmitter uses 4 FPGAs simultaneously and sends the binary string 0101 in this example. The orange colour of the FPGAs after the heating period represents high temperature. The yellow colour of the FPGAs after the reconfiguration period on the receiver side represents a temperature higher than the un-heated FPGAs, but lower than the heated FPGAs.

3.5.2 Case Study – Remote Power Side-Channel Attacks

Security researchers have studied power side-channel attacks extensively in the past decade [21,36]. An attacker can exploit the fact that the data that the system processes affects the dynamic power consumption of the system [21]. So, by observing the power consumption of the circuit, the attacker can infer the secret key in the cryptographic hardware. This attack requires side-channel information to be collected from the hardware. Consequently, it was believed that such attacks could be carried out only if the attacker had physical proximity to the target system. However, in the context of cloud FPGAs, a malicious user does not have physical access to the target FPGA. Hence, all previous techniques would not work.

3.5.2.1 Threat Model

In general, remote power analysis attacks assume that the adversary's logic and the victim's logic are on the same remote FPGA fabric [25, 37]. So, the adversary has access to some of the LUTs in the remote FPGA. In other words, the attacker can implement his/her logic on some part of the shared multi-tenant remote FPGA. Although currently, the cloud FPGA providers do not allow sharing of an FPGA by multiple users, it is envisioned that multi-tenant FPGAs will be realized soon for better efficiency in terms of cost and utilization.

3.5.2.2 Key Idea

To launch a remote power analysis attack, an attacker has to implement a power monitor on the FPGA fabric shared with the victim. For example, the attacker can monitor the power consumption of a victim process by using time-to-digital converter (TDC) sensors. Using the power traces collected by the on-chip power monitors, the attacker can perform a power side-channel attack.

3.5.2.3 Attack Method

A key component in the attack is the power distribution network (PDN) on FPGA chips. The PDN handles the distribution of power to all the components on the FPGA [38]. The PDN spans across different abstraction levels, from printed circuit board level to individual transistors on the FPGA. The PDN consists of resistive, capacitive, and inductive elements in the form of a power mesh. The power consumption of an FPGA chip at any instant depends on the logic that is being operated at that time. The changes in logic values affect the voltage and current drawn by the transistors in FPGA. These voltage fluctuations affect the delays of the other logic circuits implemented on the same FPGA due to the shared PDN. Hence, measuring delays in one part of the FPGA reveals information about power consumption in a different part of the FPGA. In particular, the higher the fluctuations in the voltage, the higher is the change in the delays. So, the attacker can monitor the power fluctuations on the FPGA by implementing appropriate delay sensors. To this end, the attacker can implement a TDC on the shared FPGA as a delay sensor [37]. As the delays of the buffers in the TDC depend on the supply voltage, the change in delays can be monitored as a proxy for voltage fluctuations. When a victim process becomes active in a different region of a multi-tenant FPGA, it disturbs the PDN. This results in a change in the delay values of the TDC sensor. Thus, the attacker can create a mapping between the power traces and the delay values. This mapping can then be used to perform a standard Correlation Power Analysis (CPA) attack. Such an attack was demonstrated in [37]. The proof of concept for this attack was demonstrated on a victim AES core operating at 24MHz on a Xilinx Spartan-6 FPGA. Two scenarios were considered: (1) when the sensor is placed close to the victim AES logic, with a gap of just 4 FPGA slices, and (2) when the sensor is placed far from the AES core. In both cases, the attacker can recover the AES key.

3.5.2.4 Attacking the Processor System

In an FPGA+CPU heterogeneous chip, like a Xilinx Zynq system, an ARM processor system (PS) shares the PDN with the FPGA fabric (programmable logic or PL). Zhao et al. demonstrated an attack that uses the PL to monitor the power consumption of the PS [25]. By doing so, they recovered the control flow of the program in the PS. This vulnerability made a simple power analysis on RSA possible. Similarly, an FPGA-to-processor correlation power analysis has been demonstrated in [39]. The authors used a TDC on the FPGA to measure the power traces of the processor. Using that, they attacked an AES core running on the processor with 111k to 127k power traces.

3.5.2.5 Experiments on Amazon Clouds

In DATE’20, Glamocanin et al. published their results on launching remote power side-channel attacks on AWS EC2 F1 instances [40]. They chose to use TDC sensors for measuring power consumption on a cloud FPGA, and the results showed that they could successfully break the secret keys of all 16 bytes of an open-source AES-128 core with 5 × 105 traces. This result validated the feasibility of remote power side-channel attacks on a commercial cloud platform, so this research area raises serious concerns.

3.6 Trojan Countermeasures

Hardware Trojan detection techniques in the literature can be classified into two categories: invasive and non-invasive. Invasive techniques require some modification to the original design to aid in fingerprinting the IC and to verify the authenticity after fabrication. A few examples include ring oscillator-based design-for-trust, IC camouflaging, and logic encryption [41, 42].

FPGAs consist of a massive amount of programmable components, and invasive techniques would require additional gates or hardware for each of these programmable components. Thus, invasive techniques are not feasible for FPGAs as this would lead to exponential silicon area overhead. Moreover, physical reverse engineering techniques can be used to test a small number of ICs, but do not guarantee that the remaining ICs are free from malicious modifications.

Non-invasive techniques, on the other hand, do not modify the original design. Instead, a fingerprint of power, timing delay, and/or other side channels of a golden design, in combination with functional testing, are used. Most of the techniques in this category use statistical analysis to distinguish a malicious chip from a genuine one.

There are only a handful of works that detect hardware Trojans in the FPGA fabric [43, 44, 45].

3.6.1 Hardware Trojan Tolerance Using Modular Redundancy

A triple modular redundancy (TMR)-based technique is used to create a Trojan tolerant design methodology in [44]. TMR is a renowned fault mitigation methodology used to mask circuit faults wherein three redundant copies of the original system perform a process, and the result is processed by a majority voting system to produce a single output. Any single fault in one of the redundant modules will not lead to an error at the output as the majority voter selects the result from the two faultless modules. TMR, however, leads to 3 area and power overhead. To reduce the overhead, adapted TMR (ATMR) is proposed where only two modules are used at a time, and the third module is employed only when there is a mismatch in the results of two active modules. An arbiter is used to identify the erroneous module. The experimental results show a 1.5 x reduction in power by using ATMR with negligible performance and hardware overhead when compared to TMR.

3.6.2 FPGA TrustFuzion

The FPGA TrustFuzion (FTZ) security mechanism is a non- destructive methodology to detect and isolate anomalies such as Trojans in the FPGA fabric. The authors use the term anomalies to indicate the presence of Trojans and reliability problems in FPGA fabric. Anomalies are detected based on the violation to the spatial correlation of intra-die PV in the FPGA fabric. These anomalies are then isolated such that any designs can run reliably even on a Trojan-infected FPGA chip. FTZ is based on the observation that physical characteristics of FPGA fabric's intra-die process variation (PV) display a huge amount of spatial correlation [46, 47, 48].

Anomalies are detected based on the spatial correlation violations. The device locations with anomalies are isolated and excluded from being used in the designs targeting the FPGA device with anomalies. The FPGA is then divided into different zones accounting for the exclusion of locations with anomalies. These FPGA areas are called TrustFuzion zones.

3.7 Summary

In the last decade, the use of FPGAs has increased significantly and has been employed in various applications including mission critical systems and cloud services. Studying the threat landscape for FPGA is critical not only from security perspective alone but also to evaluate it alongside FPGA's overall health. This chapter has identified different FPGA Trojans that can be inserted at various phases of FPGA life cycle. Most research on FPGA security emulates Trojan on FPGA while trusting the FPGA fabric. Trojans can be inserted in even FPGA fabric, similar to any other type of ICs/ASICs. FPGA fabric Trojans that can be inserted by an untrusted foundry and a malicious actor in the supply chain in literature are identified, and the taxonomy is introduced. The bitstream files and the designs in HDL format can also be corrupted with Trojans. A comprehensive FPGA attack scenario has been presented using cloud

FPGAs case study. It provides an in-depth knowledge of the current and future nature of threats and challenges to FPGA security, which in turn, impacts the reliability and hence, FPGA’s health.

This chapter also discussed the countermeasures proposed in the literature. Only a couple of techniques exist to verify the physical fabric of FPGA for hardware Trojan. Trojans present in physical FPGA fabric could be detected by accounting for spatially correlated intra-die process variations. The intra-die process variation based approach can identify anomalies contributing to delay or voltage change by locating inconsistent physical characteristics from the ones in close-by regions. Comprehensive work on malicious modification effects can be done by designing and simulating layout without and with anomalies. This would give a better insight into anomaly characteristics and would potentially aid in indicating the precise type of anomaly inserted into the FPGA fabric. However, many of the design details remain proprietary information and are not available to researchers, thus impeding the research.

REFERENCES

[1] R.S. Chakraborty, S. Narasimhan, S. Bhunia, Hardware Trojan: threats and emerging solutions, in *IEEE International High Level Design Validation and Test Workshop* pp. 166–171, 2009.

[2] R. Karri, J. Rajendran, K. Rosenfeld, M. Tehranipoor, Trustworthy hardware: identifying and classifying hardware Trojans. IEEE Comput. 43(10), pp 39–46, 2010.

[3] M. Tehranipoor, F. Koushanfar, A survey of hardware Trojan taxonomy and detections. IEEE Des. Test Comput. 27(1), pp 10–25, 2010.

[4] M. Tehranipoor, C. Wang, "*Introduction to Hardware Security and Trust,"* Springer, New York, 2012.

[5] S. Bhunia, M.S. Hsiao, M. Banga, S. Narasimhan, Hardware Trojan attacks: threat analysis and countermeasures. Proc. IEEE 102(8), pp 1229–1247, 2014.

[6] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, M. Tehranipoor, Hardware Trojans: lessons learned after one decade of research. ACM Trans. Des. Autom. Electron. Syst. 22(1), pp 6:1–6:23, 2016.

[7] Amazon, Amazon EC2 F1 instances – run custom FPGAs in the AWS cloud, [Online]. Available at: https://aws.amazon.com/ec2/instance-types/f1/. Accessed on: 15 Dec 2019.

[8] G.T. Becker, F. Regazzoni, C. Paar, W.P. Burleson, Stealthy dopant-level hardware trojans, in *International Workshop on Cryptographic Hardware and Embedded Systems* pp. 197–214, 2013.

[9] A.N. Campbell, K.A. Peterson, D.M. Fleetwood, J.M. Soden, Effects of focused ion beam irradiation on MOS transistors, in *IEEE International Reliability Physics Symposium* pp. 72–81, 1997.

[10] V. Jyothi, M. Thoonoli, R. Stern, R. Karri, FPGA trust zone: incorporating trust and reliability into FPGA designs, in *IEEE International Conference on Computer Design* pp. 600–605, 2016.

[11] Y. Pino, V. Jyothi, M. French, Intra-die process variation aware anomaly detection in FPGAs, in *IEEE International Test Conference* pp. 1–6, 2014.

[12] D. Pellerin, "*Announcing Amazon EC2 F1 Instances with Custom FPGAs*," [Online]. Available at: https://www.slideshare.net/AmazonWebServices/announcing-amazon-ec2-f1-instances-with-custom-fpgas. Accessed on: 10 Jul 2020.

[13] AWS, "*AWS Shell Interface Specification*". [Online]. Available at: https://github.com/aws/aws-fpga/blob/ master/hdk/docs/AWS_Shell_Interface_Specification.md. Accessed on: 10 Jul 2020.

[14] AWS. 2018. AWS EC2 FPGA Software Development Kit. [Online]. Available at: https://github.com/aws/awsfpga/ blob/master/sdk/README.md. Accessed on: 10 Jul 2020.

[15] Azure. 2020. Azure Machine Learning pricing. [Online]. Available at: https://azure.microsoft.com/en-us/pricing/ details/machine-learning/. Accessed on: 10 Jul 2020.

[16] Florian Benz, André Seffrin, and Sorin A. Huss. , Bil: A tool-chain for bitstream reverse-engineering. In International Conference on Field Programmable Logic and Applications. IEEE, New York, NY, 735–738, 2012.

[17] J. Baptiste Note and É. Rannaud. 2008, From the bitstream to the netlist. In Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays. ACM, New York, NY, 264–264, 2008.

[18] Kan Xiao, Domenic Forte, Yier Jin, Ramesh Karri, Swarup Bhunia, and Mohammad Tehranipoor. 2016. Hardware Trojans: Lessons Learned after One Decade of Research. ACM Transactions on Design Automation of Electronic Systems 22, 1, 1–23, 2016.

[19] Vinayaka Jyothi, Prashanth Krishnamurthy, Farshad Khorrami, and Ramesh Karri, TAINT: Tool for Automated INsertion of Trojans. In IEEE International Conference on Computer Design. IEEE Computer Society, New York, NY, pp 545–548, 2017.

[20] V. Costan and S. Devadas, Intel SGX Explained. IACR Cryptology ePrint Archive 2016, 086 pp 1–118, 2016.

[21] Paul C. Kocher, Joshua Jaffe, and Benjamin, Differential Power Analysis. In Annual International Cryptology Conference (Lecture Notes in Computer Science), Vol. 1666. Springer, Berlin, Heidelberg, pp 388–397, 1999.

[22] Paul C. Kocher, Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Annual International Cryptology Conference (Lecture Notes in Computer Science), Vol. 1109. Springer, Berlin, Heidelberg, pp 104–113, 1996.

[23] Vincent Carlier, Hervé Chabanne, Emmanuelle Dottax, and Hervé Pelletier, Electromagnetic Side Channels of an FPGA Implementation of AES. IACR Cryptology ePrint Archive 2004, pp 145, 2004.

[24] J. Krämer, D. Nedospasov, A. Schlösser, and J.P. Seifert, Differential Photonic Emission Analysis. In International Workshop on Constructive Side-

Channel Analysis and Secure Design (Lecture Notes in Computer Science), Vol. 7864. Springer, Berlin, Heidelberg, pp 1–16, 2013.

[25] M. Zhao and G. Edward Suh, FPGA-Based Remote Power Side-Channel Attacks. In IEEE Symposium on Security and Privacy. IEEE Computer Society, New York, NY, pp 229–244, 2018.

[26] C. Ramesh *et al.*, FPGA Side Channel Attacks without Physical Access. In IEEE Annual International Symposium on Field-Programmable Custom Computing Machines. IEEE Computer Society, New York, NY, pp 45–52, 2018.

[27] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, Security analysis of concurrent error detection against differential fault analysis. Journal of Cryptographic Engineering 5, 3, pp 153–169, 2015.

[28] Dennis R. E. Gnad, F. Oboril, and M. Baradaran Tahoori, Voltage Drop-based Fault Attacks on FPGAs using Valid Bitstreams. In International Conference on Field Programmable Logic and Applications. IEEE, New York, NY, pp 1–7, 2017.

[29] Z. Weissman et al., JackHammer: Efficient Rowhammer on Heterogeneous FPGA-CPU Platforms. In IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES), Volume 2020, Issue 3, 2020.

[30] I. Giechaskiel, K. Eguro, and K. B Rasmussen, Leakier wires: Exploiting FPGA Long Wires for Covert-and Side-channel Attacks. ACM Transactions on Reconfigurable Technology and Systems 12, 3, pp 1–29. 2019.

[31] I. Giechaskiel, K. Rasmussen, and J. Szefer, Reading Between the Dies: Cross-SLR Covert Channels on Multi-Tenant Cloud FPGAs. In IEEE International Conference on Computer Design. IEEE, New York, NY, pp 1–10, 2019.

[32] D. Gnad, C. D. K. Nguyen, S. H. Gillani, and M. B. Tahoori, Voltage-based Covert Channels in Multi-Tenant FPGAs, Cryptology ePrint Archive, Report 2019/1394 (2019).

[33] I. Giechaskiel, K. Bonne Rasmussen, and K. Eguro, Leaky Wires: Information Leakage and Covert Communication Between FPGA Long Wires. In Proceedings of the Asia Conference on Computer and Communications Security. ACM, New York, NY, pp 15–27, 2018.

[34] S. Tian and J. Szefer, Temporal Thermal Covert Channels in Cloud FPGAs. In Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, New York, NY, pp 298–303, 2019.

[35] C. Krieg, C. Wolf, and A. Jantsch, Malicious LUT: A Stealthy FPGA Trojan Injected and Triggered by the Design Flow. In Proceedings of the International Conference on Computer-Aided Design. ACM, New York, NY, 43, 2016.

[36] S. Mangard, E. Oswald, and T. Popp, Power analysis attacks: Revealing the secrets of smart cards. Springer, Berlin, Heidelberg, 2007.

[37] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, An Inside Job: Remote Power Analysis Attacks on FPGAs. In Design, Automation & Test in Europe Conference & Exhibition. IEEE, New York, NY, pp 1111–1116, 2018.

[38] K. Arabi, R. Saleh, and X. Meng, Power Supply Noise in SoCs: Metrics, Management, and Measurement. IEEE Design & Test of Computers 24, 3, pp 236–244, 2007.

[39] J. Gravellier, J. Dutertre, Y. Teglia, P. Loubet-Moundi, and F. Olivier, Remote Side- Channel Attacks on Heterogeneous SoC. In International Conference on Smart Card Research and Advanced Applications (Lecture Notes in Computer Science), Vol. 11833. Springer, Berlin, Heidelberg, pp 109–125, 2019.

[40] O. Glamocanin, L. Coulon, F. Regazzoni, and M. Stojilovic, Are Cloud FPGAs Really Vulnerable to Power Analysis Attacks?. In Design, Automation & Test in Europe Conference & Exhibition. IEEE, New York, NY, 2020.

[41] J. Rajendran, V. Jyothi, O. Sinanoglu, R. Karri, Design and analysis of ring oscillator based design-for-trust technique, in *IEEE VLSI Test Symposium* pp. 105–110, 2011.

[42] J. Rajendran, Y. Pino, O. Sinanoglu, R. Karri, Logic encryption: a fault analysis perspective, in *Design, Automation Test in Europe Conference Exhibition*, pp. 953–958, 2012.

[43] V. Jyothi, M. Thoonoli, R. Stern, R. Karri, FPGA trust zone: incorporating trust and reliability into FPGA designs, in *IEEE International Conference on Computer Design* pp. 600– 605, 2016.

[44] S. Mal-Sarkar, A. Krishna, A. Ghosh, S. Bhunia, Hardware trojan attacks in FPGA devices: threat analysis and effective counter measures, in *ACM Great Lakes Symposium on VLSI Design* pp. 287–292, 2014.

[45] Y. Pino, V. Jyothi, M. French, Intra-die process variation aware anomaly detection in FPGAs, in *IEEE International Test Conference* pp. 1–6, 2014.

[46] A. Agarwal, D. Blaauw, V. Zolotov, Statistical timing analysis for intra-die process variations with spatial correlations, in *IEEE International Conference on Computer Design* pp. 900–907, 2003.

[47] H. Chang, S.S. Sapatnekar, Statistical timing analysis under spatial correlations, in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System* pp. 1467–1482, 2005.

[48] B. Cline, K. Chopra, D. Blaauw, Y. Cao, Analysis and modeling of CD variation for statistical static timing, in *IEEE International Conference on Computer Design* pp. 60–66, 2006.

4 FREQUENCY DEGRADATION (FRED) DETECTION AND MEASUREMENT SENSOR

This chapter is built on the requirements emanating from the realms of FPGA reliability and security, as discussed in Chapter 2. Divided into six sections, as shown in Figure 4-1, the Section-1 gives an overview of the various factors that essentialise and justify the requirement of highly efficient and accurate on-chip sensors to monitor its health and performance. Section-2 provides a deep insight into the current research on capturing frequency/delay variability in VLSI circuits. In Section-3, the design and implementation of the FRED sensor in a 28 nm process technology is elaborated. Section-4 provides simulation and real-time experimentation alongwith results whereas Section-5 gives performance evaluation of the proposed sensor with the recommendation for the most optimal sensor configuration. The chapter is summarized and concluded in Section-6.

4.1 Introduction

The requirement for higher reliability, optimal timing performance, and lower power consumption have remained the focal point of the manufacturers of VLSI devices such as the FPGAs. Due to the continual miniaturisation of transistor technology [1], as

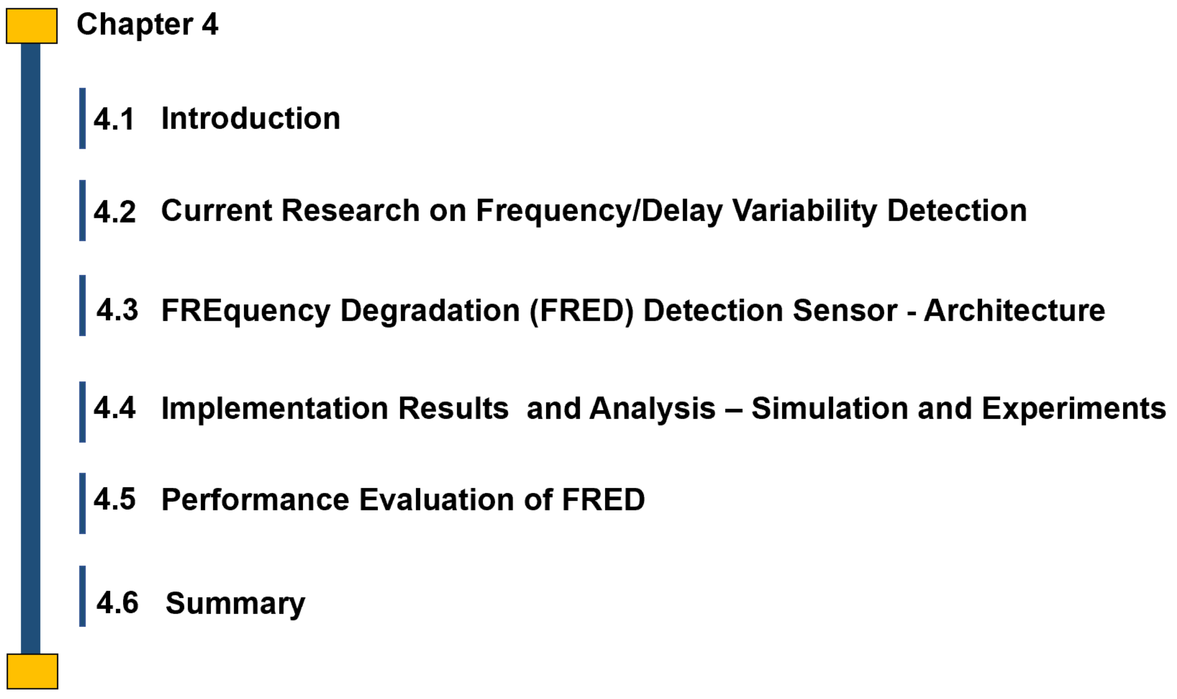

Chapter 4
4.1 Introduction
4.2 Current Research on Frequency/Delay Variability Detection
4.3 FREquency Degradation (FRED) Detection Sensor - Architecture
4.4 Implementation Results and Analysis – Simulation and Experiments
4.5 Performance Evaluation of FRED
4.6 Summary

Figure 4-1 The Disposition of Chapter 4.

shown in Figure 4-2, there is an increased probability of aggravation in degradation mechanisms (NBTI/PBTI, HCI, and EM) [2], and ease of accessibility for side-channel analysis of electrical and acoustic parameters. Where it increasingly impacts FPGA reliability, it also makes it vulnerable to hardware attacks, particularly in the wake of growing cybersecurity threats and enhanced knowledge as well as skills of individuals working toward infiltrating the hardware base of computational systems. This may lead to a gradual build-up of various types of malicious and stealthy electronic circuitries and codes (hardware Trojans) inserted into FPGAs' fabric during their pre and post-fabrication periods. Leveraging the degradation mechanisms of ageing, observable with delay degradation due to abrupt changes in CMOS circuits' electrical behaviour under high-stress environment, the reliable and secure functioning of emerging industrial IoT, cyber-physical, and autonomous systems are highly likely to be jeopardized and severely challenged. It is, therefore, essential to identify, monitor, and precisely measure electrical parameters of the CMOS base for an unbiased characterisation of the ageing phenomenon in FPGAs, thereby ensuring an effective regime of reliability and security.

Especially, the key FPGA performance parameters of frequency and delay must be observed to ensure unhindered functioning of applications built using FPGA primitives. It is well known that the frequency and the corresponding delay degradation in transistors are the function of ageing induced by the degradation mechanisms of

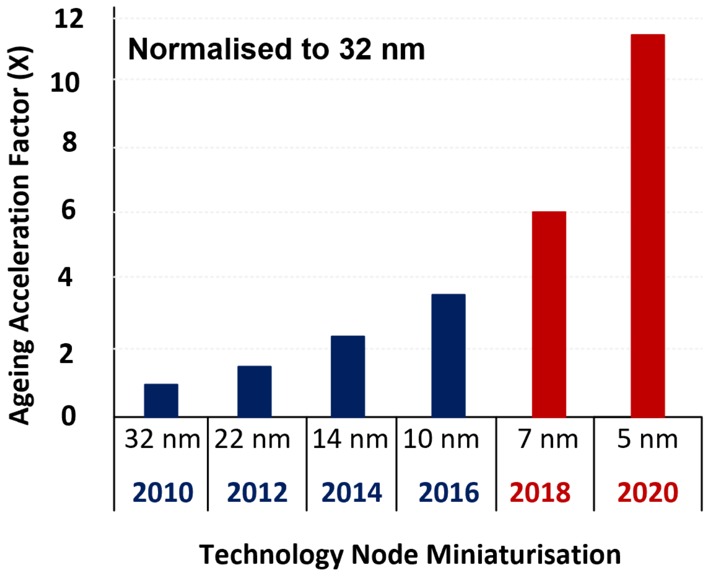


Figure 4-2 VLSI Ageing Vs Technology miniaturisation [2].

negative bias temperature instability (NBTI) and positive bias temperature instability (PBTI) among other significant phenomena such as hot carrier injection (HCI) and electromigration (EM).

Several questions warrant attention to understand and interpret the detection and measurement of ageing fully at the outset. For instance, how the degradation mechanisms of N/PBTI impact the CMOS devices with high-*K* dielectric? Is it possible to detect minor shifts in N/PFETs parameters such as threshold voltage, and how do they relate to frequency degradation/delay variability in an FPGA? Are the ring oscillator-based on-chip sensors a viable option to improve the detectability of minute parametric shifts in frequency, delays, and hence the device ageing to impair hardware Trojans? We have attempted to seek answers to these questions in the following sections.

4.2 Current Research on Frequency/Delay Variability Detection

Despite the best efforts of the manufacturers, no two FPGAs exhibit the same parametric behaviour. Delay variations, in particular, impact them at the time of manufacturing and throughout their operating lifetime. Three primary sources of delay variations identified as physical, environmental, and temporal variabilities act as catalysts in different FPGA applications. The physical variability relates to the process variation, which is primarily the parametric variation of components in an FPGA due to the variability (inconsistencies) in fabrication processes. It accounts for as much as ± 15% of delay variation at the instance of manufacturing [3]. Environmental variability implies fluctuations in supply voltage, thermal coupling, clock jitter [4], and crosstalk that causes uncertain or unexpected delays. Whereas, temporal variability includes the degradation mechanisms of BTI, HCI, and TDDB. BTI, in particular, causes shifts in the threshold voltage of N/PFETs that result in a gradual deterioration in switching performance with more than 20% reduction in the circuit operating speed of 65 nm technology nodes [3].

Various delay quantification methods and circuits have been put forth. Ring Oscillators (ROs), for instance, besides being employed for characterising the impact of the intra-die process, temperature, and voltage variations, are commonly used to infer delays [5] and [6]. Nevertheless, they are prone to self-heating due to their free-running states and dynamic power dissipation. As a result, it becomes onerous to distinguish the difference in propagation delay between the rising and falling edges of the signal.

Time to Digital Converters (TDCs) have been used for precise delay measurements of the components they are made up of, such as registers and buffers (as delay elements), followed by digitization. Different types of delay elements such as Tapped Delay Lines [7] and Vernier Delay Lines [8] provide improved delay measurement resolution for TDCs. They, however, struggle to maintain identical skew for the start and end signals, which is essentially required to cancel out the asymmetric effect.

Degradation in logic applications has also been inferred by manipulating their repeatability through Tunable Replica Circuits (TRCs). These circuits are excited with either worst-case stress data for degradation, or with vectors from the application logic. The overall performance is then monitored by employing TDC (Time to Digital Converters) or timing error detection circuitry [9]. The dominant source of degradation, such as N/PBTI, is usually tapped for constructing the worst-case data [10]. The accuracy of such a degradation inference is not well ascertained and explored [11]. However, combined with measurement technologies, TRCs can infer degradation more effectively, in addition to the quantification of intra-die process variation, junction temperature, and voltage variability [12]. This is because the circuit is designed to behave in a similar fashion to the critical paths in the application circuit so that its timing can be used to infer timing performance. Inference is enhanced by combining TRCs with error recovery [10] that allows response to fast-changing variations.

In addition, unlike canary circuits [13], for instance, TRC does not need to fail with a prescribed margin before the critical path fails. Instead, when the TRC reports an error, it is assumed that the critical path contains erroneous data as well, and error recovery is initiated. However, the main drawbacks, as compared to error-detection circuits [14-16] are the lack of ability to respond to within-die variations (thus requiring a small within-die margin), and the necessity of tuning the TRC at test time.

An extensive application of frequency sweep based techniques that introduce timing failure in the circuit to measure delay variations is also proven suitable for signature analysis, failure rate detection, and transition probability. Timing measurement employing signature analysis implies that a functional application configured to remain in the same state, when pulsed with the same input signals, will yield the same output

[17]. Subsequently, the output of the circuit under test is analysed either by generating a signature or by storing the vectors in a designated memory. An example is the Multiple-Input Signature Register (MISR) [18], which is used to generate signatures, internally or externally to the device. It improves the time consumed in measurement and simplifies the comparison for each output value. Similarly, a Linear-Feedback-Shift-Register (LFSR) is also shown to be suitable owing to its deterministic response, which helps produce the same input, if initialized with the same signal. On the other hand, in the case of transition probability, the circuit under test is stimulated with different input vectors while the clock frequency is gradually increased. For each frequency, the output transition probability is logged in the accumulator from which the circuit delay is determined by exploiting the clock jitter and the asymmetry between the rising and falling delays. The circuit delay measurement based on this method offers high accuracy with lower overheads [19].

In addition to the above, shadow registers have been employed to monitor circuits for timing failure, however, not strictly the timing measurement. For instance, failure prediction [20] has been used to provide a warning in case of the violation of a pre-defined guard-band, thereby indicating an impending timing failure. Primarily, the shadow registers are made timing-critical as compared to the register they shadow, to initiate timing failure much earlier with the increase in circuit delay. The shadow register shares both the clock signal and the D-input from the main register but with an additional delay in the path of the D-input. It results in the shadow register being more timing-sensitive than the main, resultantly triggering timing failure much earlier. Researchers [21], [22] have also proposed to enhance the performance of such monitoring circuits by advancing the clock signals to the shadow registers for pre-emptive latching and gradually increasing guard-bands to retrieve the higher amount of delay information. Presumably, such an arrangement may be less susceptible to process variations and helps enhance the probability of detection.

In [5], a flip flop named 'Razor' is proposed for the detection and response to the occurrence of timing errors. It is the main pipeline register that is configured to trigger at the rising edge and is augmented with a shadow register that samples on the falling edge of the clock. Such configuration provides additional time for the shadow register to capture the correct state of the signal. The Razor flags an error as the two registers latch different data, indicating that a timing violation has occurred in the main register.

This raises issues like metastability, which makes it hard to find a correct latched value. However, in improved designs, a metastability detector is introduced to mitigate such risks of metastability. The detector is implemented and used in the comparator circuitry to detect metastability and effect correction further. Researchers have observed that the timing faults due to all kinds of delay variability are detectable. It is, however, opined that the introduction and detection of timing errors make this circuit design non-deterministic and hence not viable for tightly coupled and hard real-time constraints-based systems.

The evaluation of the abovementioned methods and circuits for the detection and measurement of delay variability and degradation reveals several significant points. The sensors based on ring oscillators, for example, provide simple circuits that are manufactured on test chips, however, their vulnerability to self-heating and averaged rise and fall times, make TDCs a more viable choice. Though slightly more complicated, TDCs such as Tapped Delay Lines are not affected by these issues. At the same time, optimal thermal and power management of RO structures can help overcome the problems mentioned above.

Establishing the timing performance of a VLSI device, under different operating conditions and at the design corners, is key to the differentiation between a healthy and faulty (hardware Trojan-infected) chip. Signature analysis and Transition Probability (TP) are suitable as non-invasive methods for this purpose, with TP being more capable of achieving it with lower overhead and quick measurement time.

This points to the requirement of sensing circuits and methods which can directly measure the impact of frequency/delay variability on an application circuit itself, accounting for the process, environmental and temporal variabilities, with a precise inference of how the behaviour of a circuit elsewhere on the die applies to the application circuit. It should be accomplished without affecting the normal functioning of the circuit and observing gradual changes in the delay of the circuit for the signs of timing failure which has occurred or is impending. It is achievable by combining the strengths of frequency sweep and shadow register-based timing measurement schemes into a novel ring oscillator architecture to detect delay variability in an FPGA.

4.3 FREquency Degradation (FRED) Detection Sensor – Architecture

A lightweight sensor based on ring oscillators, called **FRED** sensor, is presented in this section. It is designed to help detect minute frequency/delay variations under nominal and accelerated negative and positive bias temperature instability (N/PBTI) degradation/ageing mechanisms in high-K dielectric material technology nodes, such as the target 28 nm FPGAs. Before delving deep into the sensor design and implementation, a brief analysis and understanding of ageing/degradation mechanisms are essential, as they form the basis of the design for this novel sensor.

4.3.1 Degradation Mechanisms and Device Ageing

During the functional mode of an FPGA, the transistors tend to age primarily due to N/PBTI and HCI degradation mechanisms. This work, however, is focused on N/PBTI. Essentially, N/PBTI acts as a catalyst to ageing accompanied by frequency shifts, timing issues, and propagation delays. There are primarily two main models that explain the BTI mechanisms, namely the Reaction-Diffusion (RD) and Trapping-De-trapping (TD) models. According to the Reaction-Diffusion (RD) model, BTI sets in with the generation of interface traps due to the prolonged duration of the negative/positive bias and high-temperature stresses on PMOS/NMOS transistors. In the case of negative bias temperature instability (NBTI), the 'Reaction' process occurs with the ON state of the PMOS transistor during which the covalent bond of 'Si-H'

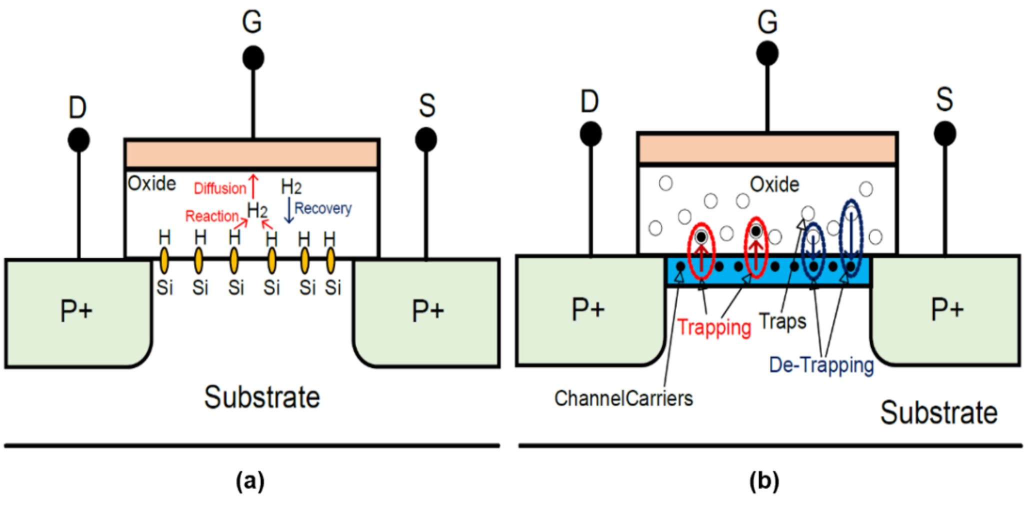


Figure 4-3 Depiction of BTI Mechanisms. (a) Reaction-Diffusion (RD) Mechanism. (b) Trapping-Detrapping (TD) Mechanism.

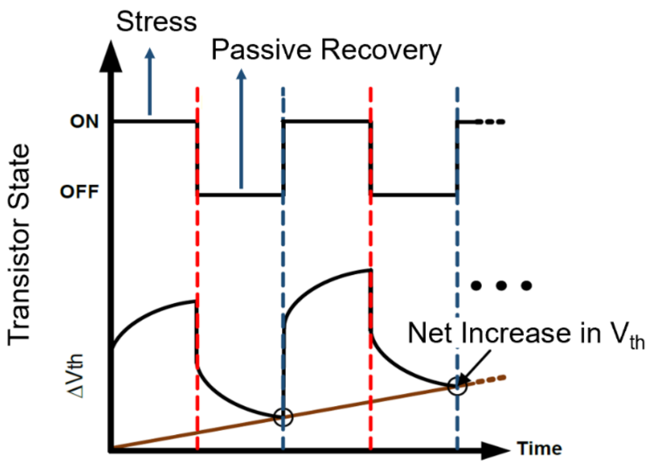


Figure 4-4 Dynamic BTI behaviour modelling based on TD mechanism. Net increase in threshold voltage V_{th} is observed despite passive recovery.

disintegrates at the interface of Si-oxide /Si-crystal lattice boundary. The disbanded hydrogen atoms combine to create H_2, thereby resulting in diffusion at the gate of the transistor, as shown in Figure 4-3(a). These disintegrated 'Si-H' bonds generate positively charged traps (called holes) that cause a negative shift in threshold voltage V_{th}. This, in turn, results in reduced transistor current and increased gate voltage. As soon as the PMOS transistor switches to an OFF state *(V_{gs} = 0)*, with stress removed, the recovery process is initiated, and H diffuses back to anneal the disintegrated 'Si-H' bonds that leads to the reduced number of interface traps and a passive recovery from NBTI degradation.

In Trapping – DeTrapping (TD) model [23] when the PMOS transistor is in the ON state, modulation of trap energy, relative to Fermi energy level, is initiated (Figure 4-3(b)). As a result, the trap generated may gain energy high enough to capture a charge carrier. It reduces the number of carriers in the channel and modulates the threshold voltage, ultimately causing scattering and reduced mobility. This whole process is termed as 'Trapping.' Whereas during the OFF state of the PMOS transistor, some of the traps undergo the annealing process, similar to the RD model, and build up an equilibrium, thereby helping the transistor achieve partial recovery.

Although the impact of PBTI has been insignificant in previous technologies above 40 nm, it has now become the source of a significant shift in threshold voltage alongside NBTI in NMOS transistors, primarily due to the introduction of high k-dielectrics [24].

It is, therefore, a common practise to model the impact of the PBTI effect similar to the NBTI degradation mechanism.

Based on the TD model, it is inferred that the V_{th} of P/NFETs increases logarithmically. The overall dynamic BTI behaviour can, therefore, be shown as in Figure 4-4. Mathematically, if at time $t = 0$, the transistor is switched ON with no application of voltage stress, the threshold voltage shift $\boldsymbol{\Delta V_{th}}$ until the application of stress at time $\boldsymbol{t_{st}}$ can be expressed as:

$$\Delta V_{th}(t_{st}) = \varphi_{st}(X+log\,(1+Yt_{st})) \quad (4\text{-}1)$$

$$\Delta V_{th}(t_{st}+t_{rec}) = \varphi_{rec}(X+log(1+Yt_{rec})) + \Delta V_{th}(t_{st})\left(1-\frac{(X+log(1+Ytrec\,)}{(X+log(1+Y\,(tst\,+\,trec))}\right) \quad (4\text{-}2)$$

$$\varphi \sim Kexp(\frac{-E}{kT})exp(\frac{ZV_{dds/r}}{kTt_{ox}}) \quad (4\text{-}3)$$

where, X, Y, and Z are constants (with negligible variation) across the same technology node, $\boldsymbol{\varphi}$ is the main variation parameter (function of T and V_{dds} / V_{ddr}), K represents the fitting parameter, E is the activation energy, t_{ox} is the oxide thickness, k is the Boltzmann's constant, T is the temperature, and V_{dds} and V_{ddr} represent the supply voltages under the stress and recovery conditions, respectively. This VLSI device level BTI model reflects the strong dependence of the V_{th} shift on the voltage and temperature during the stress as well as recovery modes with an eventual impact on the device ageing in terms of increased delays. This forms the basis of our sensor design and subsequent highly accelerated stress and life tests to validate its performance.

4.3.2 The Sensor Architecture

A detailed architecture of a novel ring oscillator-based FRED sensor is shown in Figure 4-5. As mentioned earlier, the main purpose of developing this sensor is to enable high-sensitivity as well as specificity to swiftly and correctly detect changes in frequency and corresponding delays that originate and increase with variations in the electrical parameters of the CMOS transistors under the influence of N/PBTI degradation mechanisms exacerbated by the presence and impact of the hardware Trojan attack.

4.3.2.1 Design Considerations and Objectives

In order to achieve the stated purpose, few essential design considerations cum objectives have been devised. These include 1) the ageing rate of the sensor must be very high to pick up a hardware Trojan based on threshold voltage shifts, occurring with the initiation of N/PBTI mechanism, 2) the sensor must not undergo any ageing during the fabrication/manufacturing testing phases (this requires operation management -when to switch on and off), 3) the impact of process variations on the

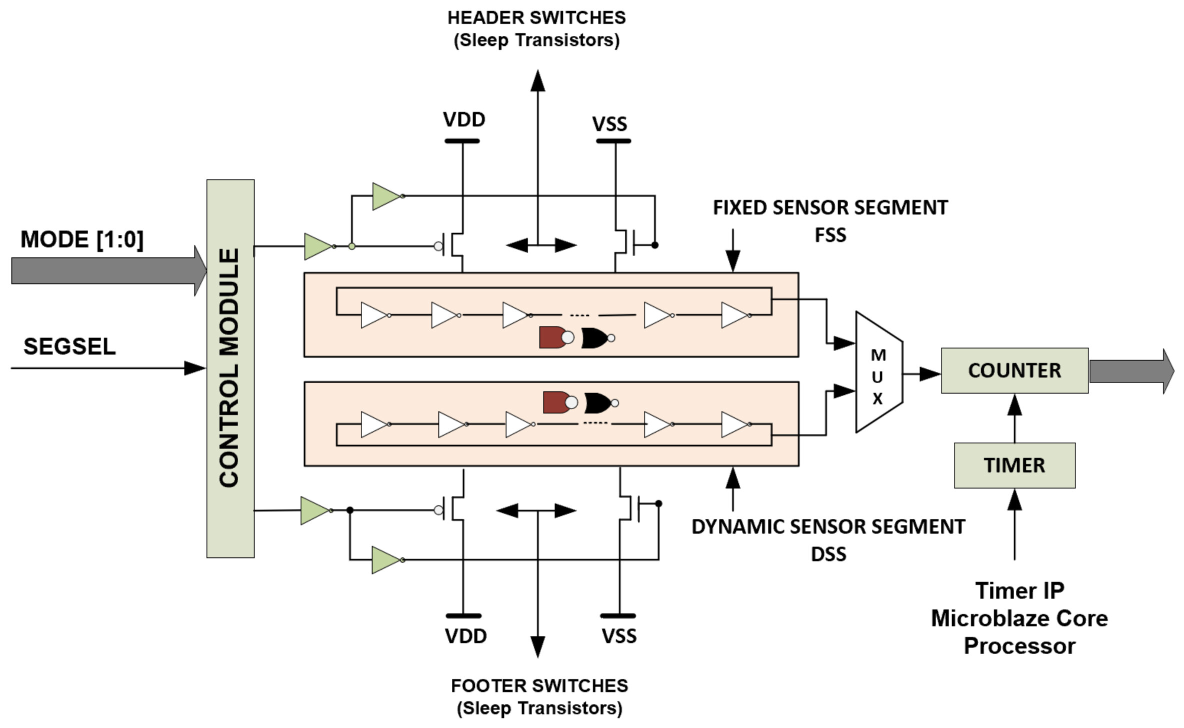

HEADER SWITCHES (Sleep Transistors)
VDD
VSS
FIXED SENSOR SEGMENT FSS
MODE [1:0]
SEGSEL
CONTROL MODULE
M U X
COUNTER
TIMER
DYNAMIC SENSOR SEGMENT DSS
Timer IP Microblaze Core Processor
VDD
VSS
FOOTER SWITCHES (Sleep Transistors)

Figure 4-5 The Architecture of FREquency Degradation (FRED) Detection and Measurement Sensor.

sensor must be negligible; and 4) finally, the sensor, itself, must also be resilient to malicious attacks.

Here, it is important to note that the incremental shifts in threshold voltages of N/PFETs, due to prolonged positive/negative bias and high junction temperatures in FPGAs or sudden bursts of hotspots and high bias, inflict the frequency degradation/lowering in ring oscillators, which in turn, causes path delays and hence point at the onset of ageing. Accordingly, the FRED sensor skeleton, shown in Figure 4-5, is architectured to have dual-sensor segments – the **Fixed Sensor Segment (FSS)** and the **Dynamic Sensor Segment (DSS)**.

The **Dynamic Sensor Segment (DSS)** is designed to experience bias and thermal stresses at accelerated rates for rapid ageing. This is made possible by employing gates having high threshold voltage, with the overall purpose of enabling faster and more accurate identification of any malicious anomaly. The Fixed Sensor Segment (FSS), on the other hand, is a minimum-stressed array of gates and is used to provide frequency reference for comparison with DSS to determine delays. FSS is, therefore, gated off from the power line during the time the FPGA is in functional mode to experience minimum stress. The frequency difference between these two segments is then used to detect the presence of a hardware Trojan that sheds its payload due to shifts in the threshold voltages of CMOS transistors. **The larger the frequency difference is, the higher the probability of the presence of a malicious anomaly, manifested as a path/propagation delay.**

However, the presence of intra-die process variations could affect the accuracy of anomaly detection and identification. We have made an effort to address this concern by placing the proposed sensor segments close to each other. It is further strengthened with an effective statistical analysis that differentiates the frequency degradation due to threshold voltage shifts from other intra-die process variations. The simulation and experimental results presented in the ensuing paragraphs support this analytical process.

4.3.2.2 The Sensor Operation

Having discussed the main structural components of DSS and FSS, the complete operational circuitry of the FRED sensor is presented herein. As shown in Figure 4-5, in addition to the RO-based segments, the sensor consists of the auxiliary elements, namely, the control module, a multiplexer, and a counter. The counter is designed to measure the cycle counts of both RO-based sensor segments at predefined timings, given and controlled by the timer. However, because of the possible measurement period variability due to the circuit ageing, the system clock is employed in the timer to counter and minimize the extent of variation. The multiplexer selects between the two segments to enable their frequency measurement, which is, in turn, controlled by the **SEGSEL (Segment Select)** signal. The FSS (Fixed Sensor Segment) and DSS (Dynamic Sensor Segment) have the same configuration of ring oscillators (number and type of gates), with the underlined CMOS transistors having high threshold voltage (**HV_{th}**). Both sensor segments are, therefore, developed with 21-stage (smaller configuration) NAND gates, keeping in view the counter's measurement speed limitation for the given 28 nm FPGA technology. In this case, a 16-bit counter is optimized to operate at the frequency of up to 1GHz. Hence, the NAND-based RO segments are implemented in a 21-stage configuration. The rationale behind this type and stage configuration is explained in Section 4.3.2.2.2.

In order to connect the FSS and DSS segments to the power supply, sleep transistors (shown in Figure 4-5) are used such that PMOS sleep transistors act as the header switches and control the supply connection between the V_{DD} and the sensor segments. NMOS sleep transistors, on the other hand, act as the footer switches and control the supply connection between the sensor segments and the V_{SS}.

4.3.2.2.1 Modes of Operation

Two modes of operation, namely '**the functional mode'** and '**the detection mode**,' have been devised for the FRED sensor. These modes are executed through the control module by different mode signals as delineated below:

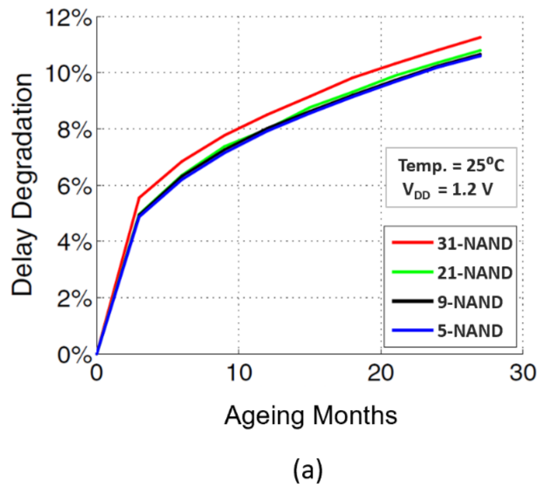
- When the FPGA is in functional mode, the fixed sensor segment (FSS) is kept disconnected from V_{DD} and V_{SS}. The dynamic sensor segment (DSS) is gated ON and exposed to stresses. Consequently, the frequency of DSS will degrade gradually with shifts in threshold voltages of N/PFETs. The delays are eventually detected as traces of ageing.
- When the FPGA is in detection mode, both the segments are gated ON by connecting them to the power supply. DSS will experience a gradual as well as

sudden frequency degradation (for instance, with the activation of hardware Trojan) at a much higher rate as compared with FSS.

The timer and counter are enabled to measure the cycle counts of sensor segments (FSS and DSS). Accordingly, the SEGSEL (Segment Select) signal is generated to select the segment whose frequency needs to be measured. It is pertinent to note that during the detection mode, the applications running on FPGA are turned off by the mode signals for a very short period. The modes of operation are designed to ensure that the frequency difference between the FSS and DSS sensor segments is detected precisely with each increment over time as the FSS (Fixed Sensor Segment) cannot be gated ON in isolation. Also, in the normal functional mode, as the FSS is kept isolated from the power supply, it is extremely challenging for an adversary or a rogue element to observe its presence or for that matter, change its mode to delays cum ageing detection. The only method that could enable the modification or disabling of the FRED sensor would be the bitstream reverse engineering process [25], which again is quite cumbersome and unreliable. **It is, therefore, very difficult for the rogue element to detect, modify, or damage the sensor.**

As is evident from Figure 4-5, the NAND gates (when replaced with inverters) of the FSS and DSS segments are placed physically next to each other as a small composite module. This helps in reducing the process and environmental variations between them to a considerable extent. It is, therefore, assumed that the frequency difference between the FSS and DSS segments would be negligible for the malicious anomaly-free FPGA. However, this may not be true in the case of the normal functional mode of the FPGA when it is kept in operation for a longer duration. The dynamic sensor segment (DSS) may have suffered delay degradation and hence age from its own oscillations, whereas the fixed sensor segment (FSS) remains unaffected as it is gated OFF during the normal functional mode.

The FRED sensor presents a small area overhead (explained in Section-4.5.2) with no constraint on the circuit layout and is designed to be robust against any removal and tampering attempts by rogue elements. Moreover, we have devised two working modes of the sensor to ensure that the Fixed Sensor Segment is not gated on its own and eventually prevent the integrity of the measured frequency difference between the two sensor segments (FSS and DSS) from being modified to mask detection.



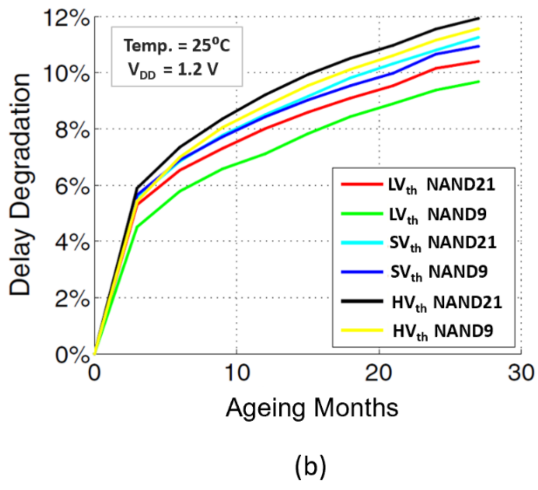


Figure 4-6 Degradation of NAND gate (as Sensor Segment) with (a) different number of stages and (b) different threshold voltage levels.

4.3.2.2.2 Rationale for 21-NAND Gate based RO Configuration

The selection of NAND gate with HV_{th} (0.46 – 0.85 Volts) is based on the simulation results, shown in Figure 4-6(a) and (b). The chains comprise a configuration of 5, 9, 21, and 31 NAND gates with SV_{th} (Standard threshold voltage – 0.46V), HV_{th} (High threshold voltage – 0.85V), and LV_{th} (Low threshold voltage < 0.46V) ranges. The NAND chains are exposed to high clock stress for an uninterrupted period of 27 months. The response is captured in Figure 4-6(a). It is observed that there is a considerable effect of the clock stress on the degradation of NAND chains. This implies that the number of NAND gates does influence the level of degradation, despite the fact that they experience the same amount of stress. It, ultimately, results in a modest rate of delay degradation.

Taking into account the threshold voltage parameter, the NAND chains comprising different NAND sizes are simulated using all the three ranges, i.e., SV_{th} , HV_{th}, and LV_{th}. It is observed that the chain of NAND gates with HV_{th} undergoes a higher level of degradation as compared to the chain with LV_{th} or SV_{th}. For instance, as shown in Figure 4-6(b), the NAND-21 with HV_{th} has a more pronounced degradation than the NAND-21 chain with SV_{th}.

In the search for the most effective configuration for FRED sensor segments, NAND and NOR gate chains with HV_{th} are also simulated at 25 degrees Celsius and V_{DD} at

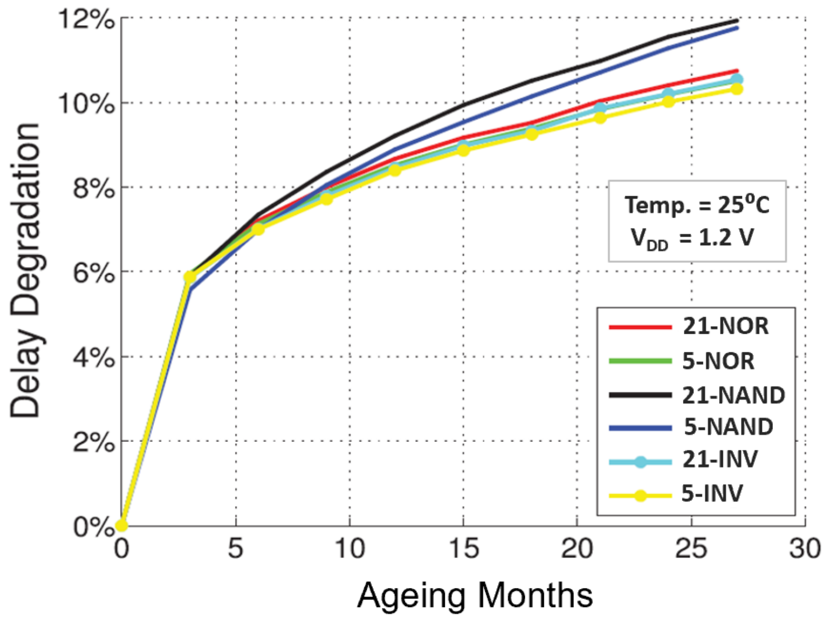


Figure 4-7 Degradation pattern of different gate configurations.

1.2V, keeping the clock stress at 500 MHz. The results shown in Figure 4-7 confirm that the gate type (same as the number of gates) does present a modest impact on the ageing speed. It is obvious that the NAND chain ages relatively faster than the others whereas the NOR and the Inverter chains exhibit approximately the same percentage of delay degradation. Therefore, in order to develop FRED sensor segments, the NAND gates' chain with HV_{th} is considered viable to conduct further simulation tests and verify the sensor's detectability.

4.4 Implementation Results and Analysis – Simulation and Experiments

4.4.1 The Sensor Simulation

Before hardware implementation of the FRED sensor architecture in a 28 nm FPGA, a comprehensive simulation test regime is designed to verify its architectural concept. For this purpose, the **AgeMOS-II** model of Cadence Virtuoso RelXpert Reliability Simulator is leveraged to investigate the individual as well as the combined effect of N/PBTI on the ageing and delay degradation of P/NMOS transistors at the gate level. Initially, different numerical sets of NAND gates (9 and 21-stage – an arbitrary selection for minimum area and power consumption) with the same capacitive load and clock stress at 500 MHz are simulated at 25, 80, and 110 degrees Celsius at a nominal voltage of 1.2 V. The DSS is gated ON and subjected to N/PBTI stressing

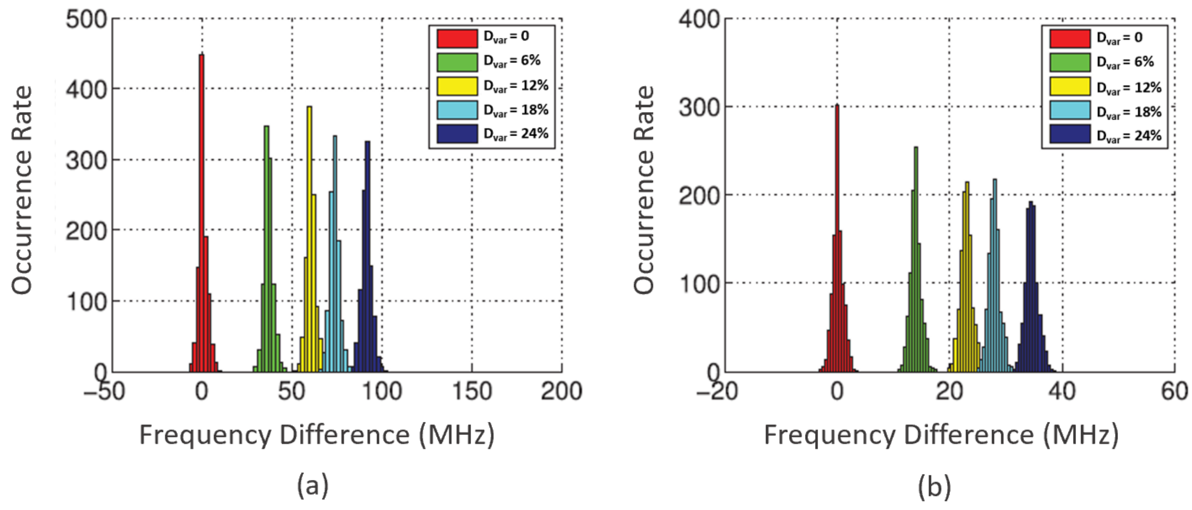


Figure 4-8 Frequency Difference distribution of the FRED sensor with PV_a employing (a) 9-Stage Sensor Segments and (b) 21-Stage Sensor Segments.

whereas the FSS is kept gated OFF. In the detection phase however, as mentioned earlier, both the segments are gated ON and their frequency differences are measured as dictated by the SEGSEL signal. The timer is set for the measurement time of 100 µs during the simulation process. The frequencies of the FRED sensor segments are retrieved from the counter (cycle count/measurement time) as soon as it is clocked by the signal from the segment.

4.4.1.1 Some Analytics – Stage Analysis

The FRED sensors with 9-stage and 21-stage segments are simulated at 25ºC, 80ºC, and 110ºC with the nominal-case (PV_a) intra-die process variations, given in Table 4-1. Accordingly, 1000 chips are generated using Monte Carlo simulation by RelXpert Simulator with the maximum delay variability set at 24% and a 6% step progression. As can be seen in Figure 4-8(a), there is a gradual increase in the frequency difference $\boldsymbol{f_{diff}}$ between the 9-stage fixed and dynamic sensor segments (FSS and DSS). A healthy FPGA has a $\boldsymbol{D_{var}}$ (Delay Variability) near to 0% with ±1%

Table 4-1 Process variations profile for inter and intra process variations.

	Inter-die			Intra-die		
	V_{th} (%)	L (%)	T_{ox} (%)	V_{th} (%)	L (%)	T_{ox} (%)
PV_a	3	3	2	4	4	1
PV_b	6	6	3	5	5	2
PV_c	15	15	5	10	10	4

difference depending upon the impact of process variations. It is, therefore, prudent to refer to the range of healthy FPGA frequency differences $f_{diff(h)}$ to compare and determine the subsequent Delay variabilities. For instance, at D_{var} = 6% the f_{diff} increases to the ranges between 5 and 20 MHz as a result of high bias and thermal stresses on the dynamic sensor segment (DSS). It can be seen that even the minimum or lowest f_{diff} at D_{var} = 6% is larger than the largest $f_{diff(h)}$ observable in the healthy FPGA range of frequency differences. This implies that as the Delay variability approaches 6%, it is readily detected by the sensor. In other words, the detection rate is 100% in this case. Similarly, at D_{var} = 12%, 18%, and 24%, the f_{diff} increases with gradual increase in stresses experienced by DSS.

The frequency difference f_{diff} profile of 21-stage sensor segments, shown in Figure 4-8(b), although much smaller as compared to the 9-stage sensor segments, it still enables the FRED sensor to detect D_{var} with a 100% detection rate. This implies that the absolute value of f_{diff} between the FSS and DSS sensor segments may be affected with larger number of RO-segment stages. Nevertheless, the D_{var} detection rate is not affected significantly.

4.4.1.2 Process Variations and Temperature Analysis

The accurate detection of Delay variability by the sensor is partially dependent on the intra-die process variations that may exist between its fixed and dynamic sensor segments. For an effective and a reliable performance by the sensor, it is important to keep as minimum and small the impact of intra-die variations as possible. Table 4-1

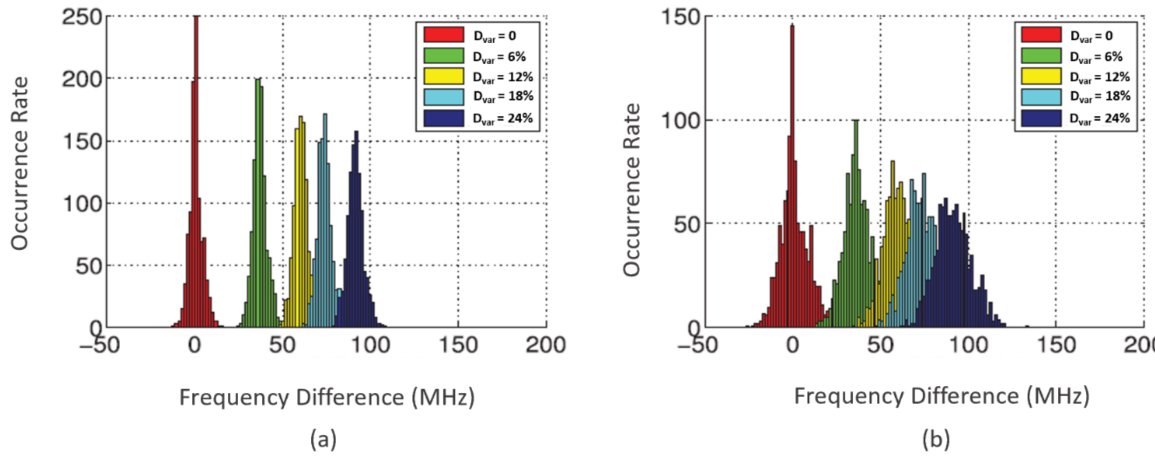


Figure 4-9 Frequency Difference (f_{diff}) distribution of a 21-Stage FSS and DSS segment sensor with: (a) PV_b and the temperature of 80ºC and (b) PV_c , and a temperature of 110ºC.

shows the different intra-die and inter-die process variations with different impact levels starting from PV_a to PV_c – Best, Typical, and Worst-case. One of the ways to minimize the impact of process variations is designing the FRED sensor as a small module and then using the hard macro, place the FSS and DSS sensor segments physically close to each other. Looking at the simulation results of 1000 chips with process variations PV_b and PV_c in Figures 4-8 (a) and (b) respectively, the $\boldsymbol{f_{diff}}$ between the sensor segments becomes larger with higher levels of process variations. The $\boldsymbol{D_{var}}$ detection rate, in this case, with PV_b is found to be around 96% for $\boldsymbol{D_{var}}$ = 6%. However, it goes up to 100% as the $\boldsymbol{D_{var}}$ = 12%, 18%, and 24%. Figure 4-9(a) represents the frequency difference occurrence rate between the 21-stage FSS and DSS segments with PV_b and the temperature of 80ºC. Similarly, Figure 4-9(b) presents the simulation results with PV_c , and a temperature of 110ºC. It is observed that $\boldsymbol{f_{diff}}$ variations in Figure 4-9(a) are comparatively more than those in Figure 4-8(a). In addition to this, it is observed that the detection rate of $\boldsymbol{D_{var}}$ = 6% is around 94.5%. However, it goes up to 100% as the Delay variability rate increases, thereby, demonstrating that the FRED sensor is effective and sensitive even in the presence of worst-case process variations and high temperatures.

In a nutshell, the minor Delay variability that can be 100% detected using the FRED sensor, could be slightly different for other technologies. The performance of the sensor is liable to be affected by the process variations and high temperature regimes. It is deduced that the FRED sensor comprising HV_{th} cells can detect FPGAs suffering from minor Delay variability with a 100% detection rate as compared to the sensor with LV_{th} cells.

4.4.2 The Sensor Implementation and Validation

A simple frequency measurement process flow is shown in Figure 4-10. for the detection of delay variability due to the impact of N/PBTI degradation mechanisms on the target 28 nm FPGA technology. 45

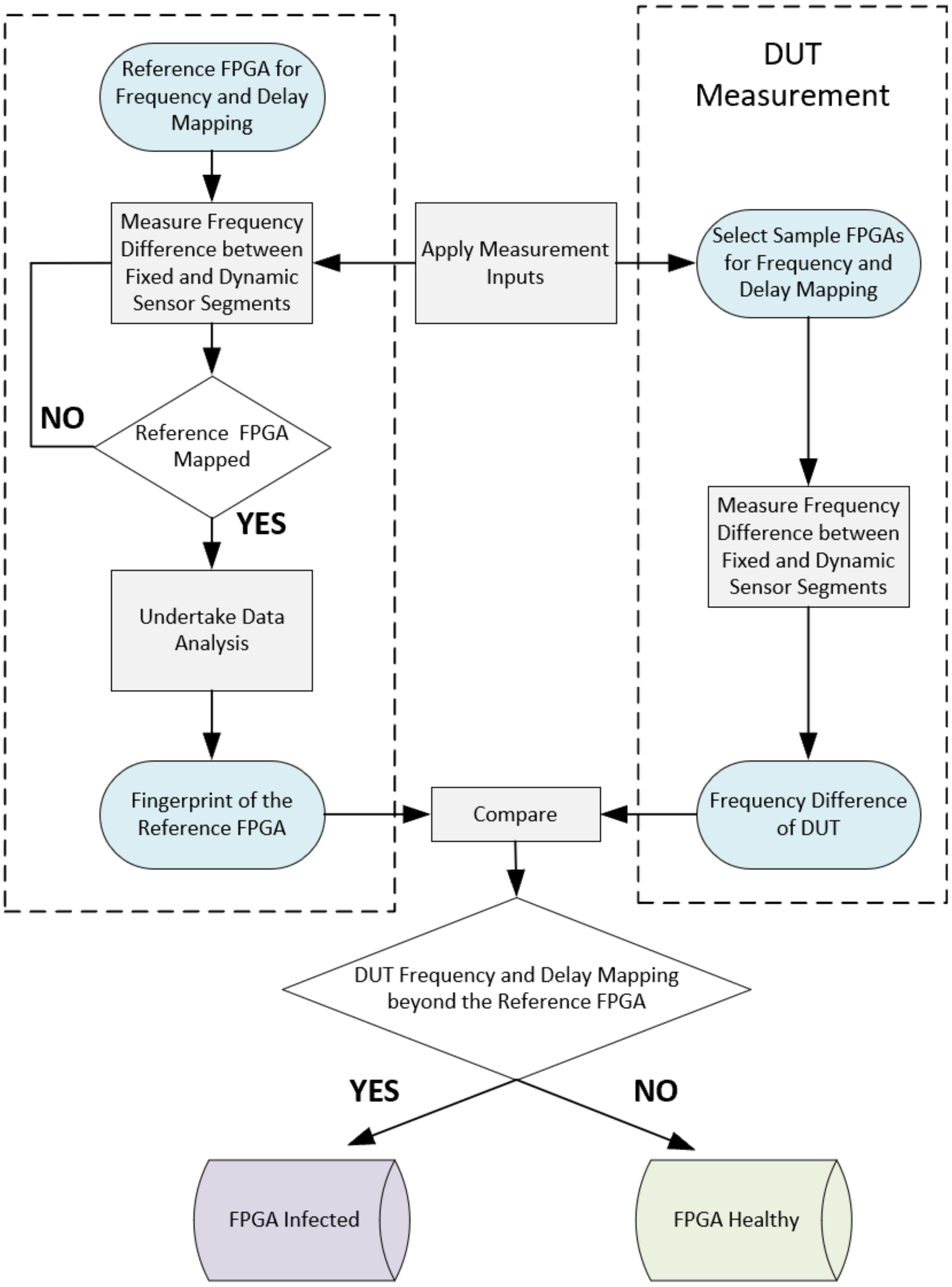

DUT
Measurement
Reference FPGA for Frequency and Delay Mapping
Measure Frequency Difference between Fixed and Dynamic Sensor Segments
Apply Measurement Inputs
Select Sample FPGAs for Frequency and Delay Mapping
NO
Reference FPGA Mapped
YES
Measure Frequency Difference between Fixed and Dynamic Sensor Segments
Undertake Data Analysis
Fingerprint of the Reference FPGA
Compare
Frequency Difference of DUT
DUT Frequency and Delay Mapping beyond the Reference FPGA
YES
NO
FPGA Infected
FPGA Healthy

Figure 4-10 A Simplistic Frequency and Delay Measurement Process Flow .

Table 4-2 Thermal and Voltage Stress Test Conditions

Stress Test Condition-1				Stress Test Condition-2			
Temp (ºC)	Bias (V)	PV (%)	Duration (hrs)	Temp (ºC)	Bias (V)	PV (%)	Duration (hrs)
60	AC	PV$_a$	2	25	1.2 DC	PV$_a$	27
110					1.4 DC	PV$_b$	
					1.6 DC	PV$_c$	
Stress Test Condition-3				**Stress Test Condition-4**			
Temp (ºC)	Bias (V)	PV (%)	Duration (hrs)	Temp (ºC)	Bias (V)	PV (%)	Duration (hrs)
80	AC	PV$_b$	27	25	1.2 AC	PV$_a$	27
110					1.4 AC	PV$_b$	
					1.6 AC	PV$_c$	

In order to establish the validation and effectiveness of the proposed FRED sensor, frequency and delay degradation under N/PBTI experiment is conducted under high thermal and voltage stress conditions (given Table-4-2) on Xilinx Spartan-7 (28 nm process) FPGA having 52,160 logic cells, 32,600 look-up-tables (LUTs), and 65,200 flip flops (FFs). Using Vivado Design Suite and Verilog, the FRED sensor as described in Section 4.3.2 is implemented in such two FPGAs using 8-LUTs (Look Up Tables) and 6-FFs (Flip Flops) fabric. This accounts for just **0.018%** and **0.009%** of the LUT and FF resources respectively, available in the target FPGAs. Similarly, the power consumption measured using the Vivado Power Estimator and Analyser comes out to be **1.5%**. For the full spectrum frequency (***f****max*)mapping of the target FPGAs, the

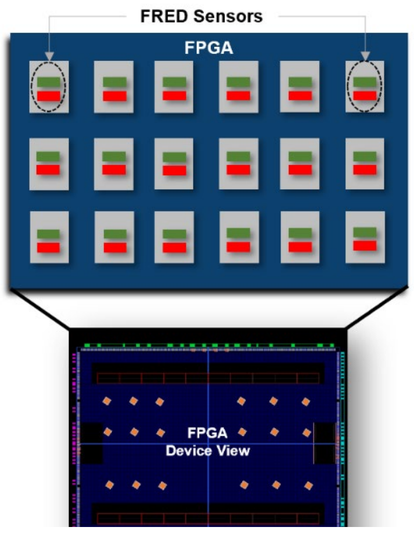


Figure 4-11 28nm FPGA Floorplan showing the spread of 18 FRED sensors implemented to capture Frequency/Delay shifts under varying Thermal and Voltage Stress Conditions.

FRED sensors are spread across the FPGA surface as per the floorplan shown in Figure 4-11. Two sets of experiments are conducted with one experiment on each of the test FPGAs. First experiment comprises the accelerated ageing of the test FPGA-1 under a highly stressed temperature of **110ºC** and an elevated voltage of 1.6V for the duration of 72 hours. This is conducted on the test FPGA-1 with the FRED sensor implemented in 21-stage NAND configuration. The second experiment comprises the same test configuration with specific stress test conditions to observe N/PBTI behaviour and capture the frequency response in the test FPGA-2.

4.4.2.1 Experimental Setup

The auto-test flow has been developed to capture frequency and delay degradations. It is shown in Figure 4-12. The test FPGAs are interfaced with the computation system – PC through the application development and evaluation boards, namely Xilinx Zedboard and Arty-S7 that contain the 7-series Spartan-7 FPGAs/SoC. The Application Processor Unit (APU) controls the required peripherals using the ARM

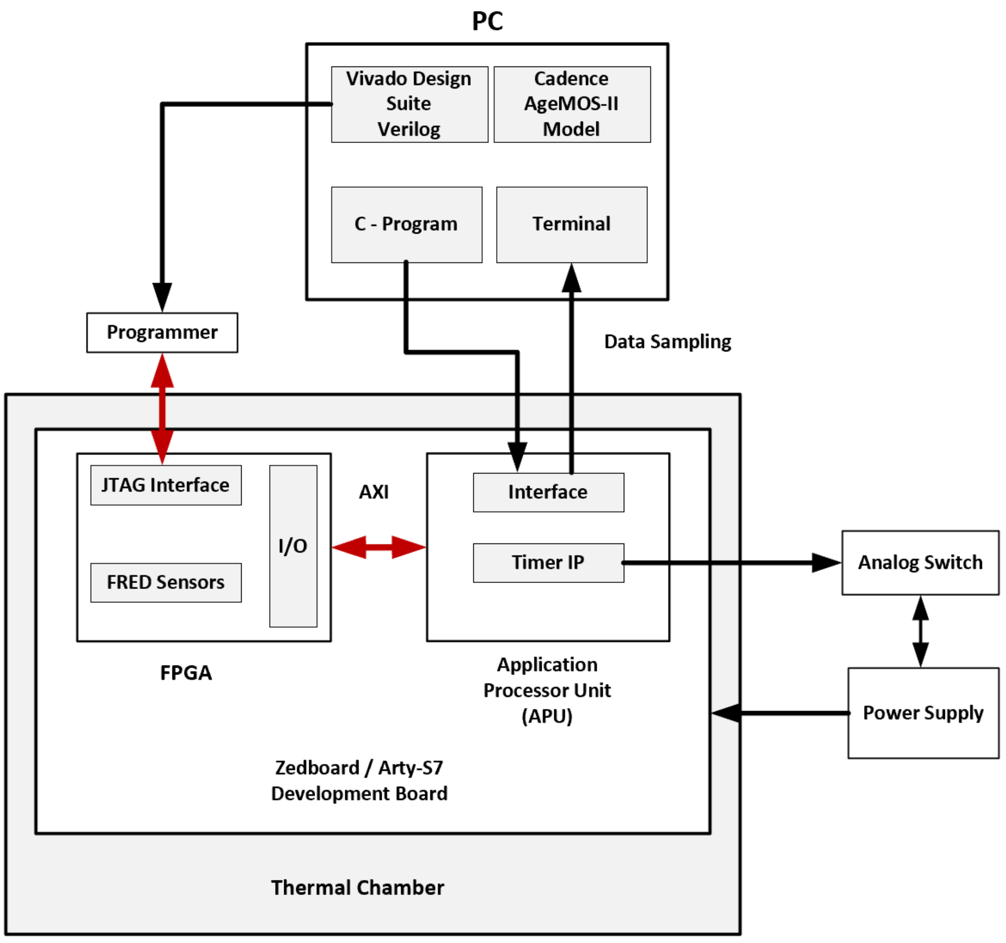


Figure 4-12 FRED Sensor Experimentation Setup and Auto-Test Flow.

AMBA AXI interconnect and also runs the software application in C that fits into the 36Kb BRAM and facilitates reading the sensor's data and decoding them, if required. In order to activate the FRED sensor, the timer IP core is employed to manage the duration of FPGA stressing, which is set to be 100 μs. Moreover, it controls the analog switch to manage the supply of required voltages to the test FPGAs. The RTL descriptions of the FRED sensor are loaded into the test FPGAs using JTAG programmer and SPI (Serial Peripheral Interface). The counter outputs are read out and saved to the PC through the board interface. Employing this test setup and methodology helps achieve the data sampling in < 3s and maintain the integrity of the FRED sensor's validation.

4.4.2.2 Results and Analysis – Experiment 1

In this experiment, the FRED sensor is implemented in the test FPGA-1 as per the floorplan shown in Figure 4-11. Both the RO-based sensor segments, FSS and DSS, are configured as high threshold voltage (HV_{th}) 9-stage (f = 450 MHz), 21-stage (f = 300 MHz), and 31-stage (f = 200 MHz) NAND configuration-based ring oscillators. With the application of the stress mode of the FRED sensor, the DSS (dynamic sensor segment – stressed) segments are enabled and stressed for the duration of 72 hours at the junction temperature of 110^{o}C (achieved through the thermal chamber) under an elevated voltage of 1.6V (provided from an external power supply). The FSS (fixed sensor segment – unstressed) segments are kept gated OFF.

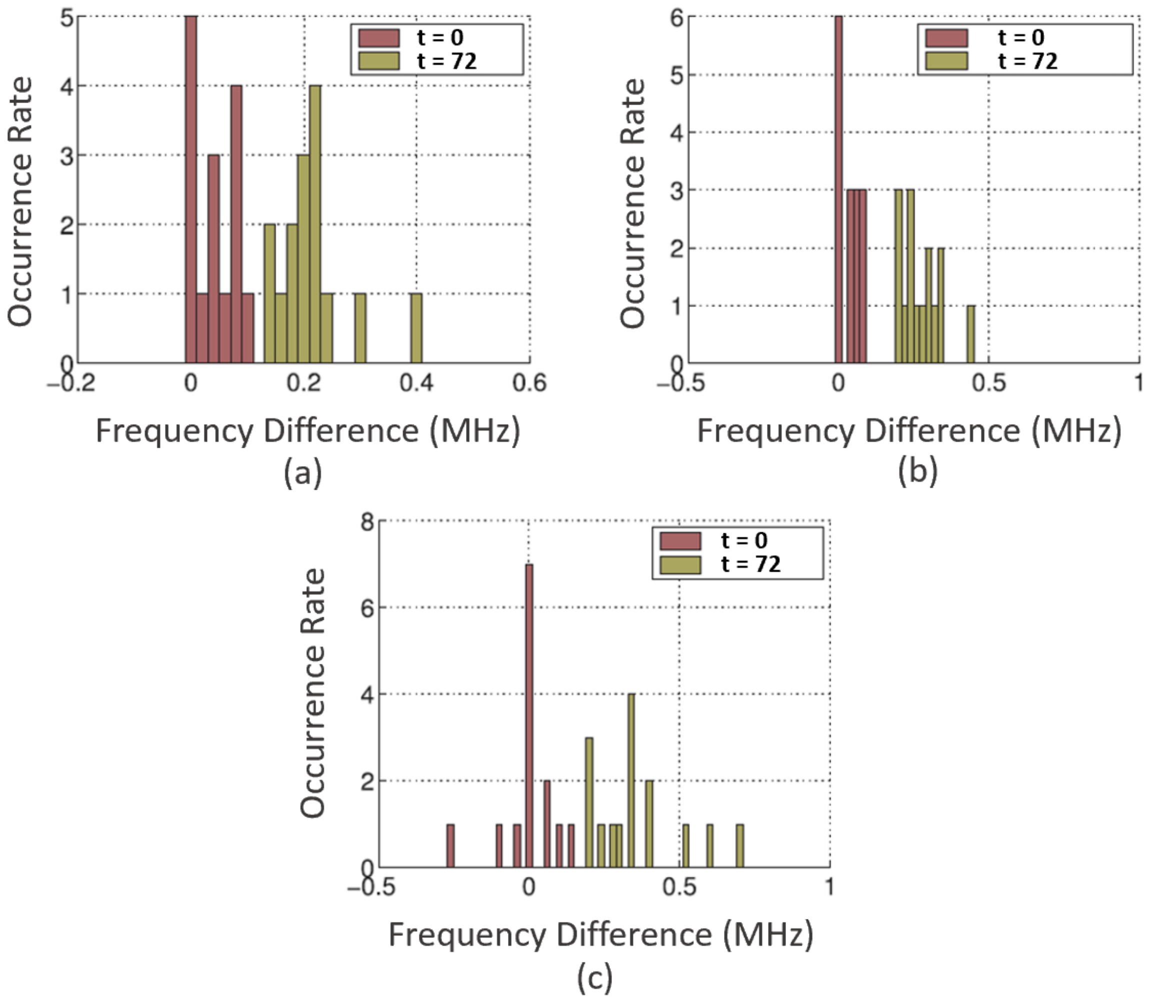


Figure 4-13 Frequency Difference Distribution – (a) 9-Stage Sensor Segment – 450MHz. (b) 21-Stage Sensor Segment – 300MHz. (c) 31-Stage Sensor Segment – 200MHz.

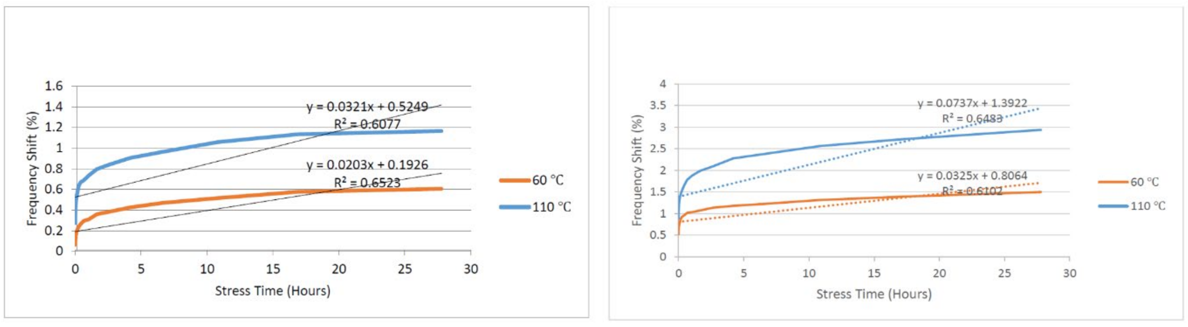
Upon completion of the stressed period, the detection mode is enabled. Accordingly, the FSS and DSS segments are also enabled and ramped down to the room temperature (around 20ºC) and nominal voltage of 1.2V. The results are shown in Figure 4-13(a) - (c). The brown bars represent the frequency difference between the FSS and DSS segments in each FRED sensor at the time t=0, whereas, the green bars are the frequency difference between the sensor segments post 72 hours of ageing stress.

4.4.2.2.1 Analysis

It can be observed that the mean frequency of the sensor segments in the test FPGA is not much different (between 250 – 300 MHz) from the one achieved during the simulations. This is, primarily, due to the same number of gates/stages used in the experiment as compared with the simulation. As a result, the detection rate of frequency degradation and delay variability remains 100% with the different stage configuration of the RO-based sensor segments. The results reveal that the average frequency degradation of 9-stage, 21-stage, and 31-stage DSS segments is 3.2%, 4.0%, and 3.8% respectively. Also, a comparison of 21-stage (HV_{th}) and 9- stage (HV_{th}) sensor segments shows that the frequency difference gap between the FPGA at t=0 and t=72 hrs. in 21-stage sensor segments is larger than that in the 9-stage sensor segment. This implies that sensor segments, with HV_{th} and larger stage gates', configuration are more effective than the ones with less stages. With respect to the detection rates, a further comparison of Figure 4-13(b) – (using 21-stage SS, composed of HV_{th} NOR gates) with Figure 4-13(c) - (using 31-stage SS, composed of HV_{th} NAND gates) reveals a minor impact on the sensitivity of the FRED sensor, which may not be much significant. In addition to this, it is observed that for 9-stage and 21-stage sensor segments, at t=0, the FSS sensor segments (unstressed) are faster than the DSS sensor segments (stressed) in most of the cases except for 31-stage sensor segment. This points toward the impact of spatial variations that come into play as the segments are not located/placed close to each other. As a result, some RO-based sensor segments tend to be more agile than the others. It may be, therefore, prudent to place the two sensor segments in a single localized module to minimize the variation between them.

4.4.2.3 Results and Analysis – Experiment 2

In line with Section on N/PBTI degradation mechanism, we conducted further sensor validation tests using the 'Stress Test Conditions - STC', given in Table 4-2. The RO-



(a) (b)

Figure 4-14 Percent Frequency Degradation with High Temperature. (a) NBTI AC Frequency Degradation (%) and (b) PBTI AC Frequency Degradation (%).

based sensor segments are stressed at high temperatures of 60ºC and 110ºC under an AC stress for over 27 hours to observe NBTI and PBTI based AC frequency degradation. The results are shown in Figure 4-14(a) and (b). As illustrated in Figure 4-14(b), PBTI causes a significantly higher frequency degradation as compared to NBTI (Figure 4-14(a)), primarily due to the high-k/metal dielectric constituting the 28 nm technology node. As can be seen, the frequency degradation goes as far as 3% at 110ºC in 27 hours duration for PBTI whereas NBTI causes 1.2% frequency shift. It must be noted that we also considered the impact of process variations '$\boldsymbol{PV_a}$', as mentioned in Table 4-1 in ascertaining the percent shifts in sensor segments' frequency from $\boldsymbol{f_{max}} =$ 300 MHz (for 21-stage NAND gate ring oscillator). R^2 in the Figure 4-14 represents the r-squared value for a linear trendline, implying that the nearer it is to 1, the more the linearity. Whereas, $Y = 0.0321x + 0.5249$ is the equation of the linear line transiting through the curve.

Similarly, the results for 'Stress Test Conditions' 2 – 4, are illustrated in Figures 4-15 (a) to (e). Figure 4-15(a) is representing the frequency degradation due to the cumulative effect of N/PBTI at 80ºC and 110ºC under the AC stress over a duration of 24 hours. It results due to the shifts in threshold voltage of P/NFETs. A frequency shift of around 2.3% and 2% is observed at 80ºC and 110ºC respectively. This is discussed in detail in Chapter-5. Likewise, for the AC and DC stresses at gate stress voltages of 1.2V, 1.4V, and 1.6 V at a nominal temperature of 25ºC, we observe the same

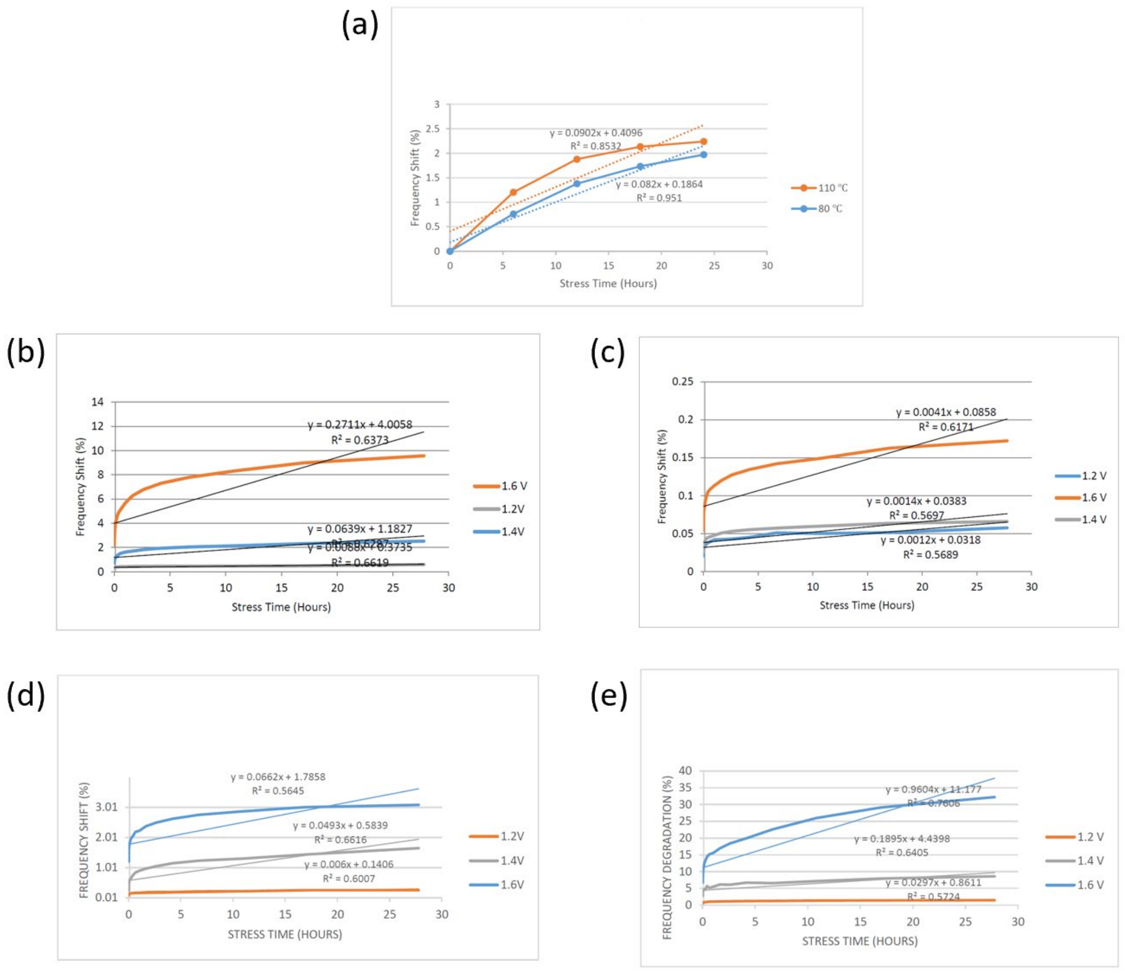
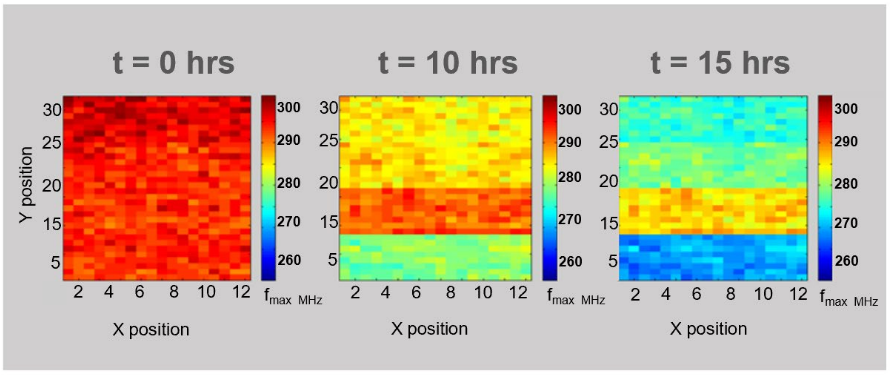


Figure 4-15 (a) AC Frequency Degradation (%) with N/PBTI. (b) PBTI Frequency Degradation (%) under AC Stress. (c) NBTI Frequency Degradation (%) under AC Stress. (d) NBTI Frequency Degradation (%) under DC Stress. (e) PBTI Frequency Degradation (%) under DC Stress.

frequency shift trend with PBTI more prominent in comparison to NBTI for 28 nm technology node.

These experiments were mainly directed to validate the FRED sensor's frequency response to NBTI and PBTI degradation mechanisms. We observed the impact of process variations ***PV**$_a$*, ***PV**$_b$*, and ***PV**$_c$* whilst conducting all different tests. By placing the two sensor segments (FSS and DSS) in close proximity to each other, we have tried to minimize the effect of process variations on the accuracy of frequency degradation measurements. However, this needs to be further studied in-depth to analyse and categorise their impact from both the spatial and temporal perspective.

In addition to the above, the implementation of 18 FRED sensor across the entire FPGA fabric helped capture the frequency degradation pattern with the conduct of the above mentioned experiments, under four different stress conditions. The results are illustrated in Figure 4-16. It is evident that N/PBTI mechanism of degradation impacts the frequency profile of FRED sensor at high temperature and bias conditions. This impacts the delay profile of applications running on the FPGA and with prolonged stress durations, the lifetime of the device tends to decrease significantly, marked by



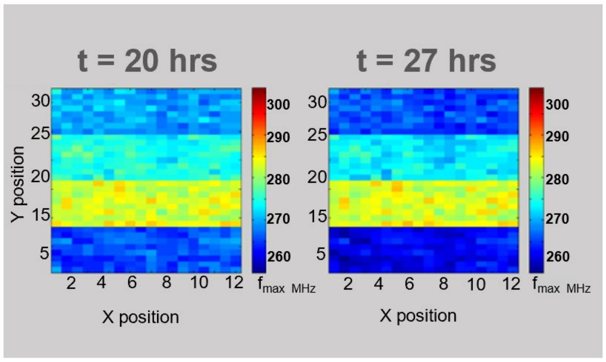


Figure 4-16 Frequency Colour Map of 28nm FPGA using FRED Sensor Under Stressed Temperature and Bias Conditions.

timing failures.

4.5 Performance Evaluation of FRED

In this section, a comparison of the proposed FRED sensor is drawn with some of the previously proposed sensors, in terms of the resource utilization and sensitivity.

4.5.1 Resource Utilization and Sensitivity

The proposed sensor occupies 8 LUTs (6 for the ring oscillators and 2 for the counter) with 21-stage length of the RO-based sensor segments. As is evident from Table 4-3, the FRED sensor utlilises modest amount of resources in comparison to other proposed sensors. In addition to the parameter of resource utilization, the most critical design and evaluation consideration is the sensitivity (S). It is a measure of the variation in the oscillation frequency of the ring oscillator (the sensor segments – FSS and DSS)/the counter value with the rise and fall of temperature over a stipulated time period or the specified range. Precisely, it is defined as the amount of reduction in RO frequency per ºC rise in the temperature. Mathematically,

$$S = \frac{F_{max} - F_{min}}{T_{max} - T_{min}} \tag{4-4}$$

where, $F_{max}\ and\ F_{min}$ represent the oscillation frequencies of the RO-based sensor segments during the time interval at $T_{min}\ and\ T_{max}$ temperatures, respectively. In the case of the proposed FRED sensor, it has been found to be 165 KHz/ºC in the three different RO lengths (9, 21, and 31) approximately. Table 4-4 provides a comparison of the FRED sensor with other sensor designs. It can be seen that with the continual downscaling of transistors' feature size, the dependency of ring oscillator frequency on temperature variation is getting low, thereby reflecting a negative trend for the RO-based sensor designs. For instance, the authors in [26] have reported 0.11%/ ºC reduction in the sensitivity of their RO-based design with a 90 nm technology node. Similarly, the sensitivity of the RO-based sensor is reduced to 0.032%/ºC for 65 nm technology node [27]. However, the FRED sensor's dual sensor segment design has

Table 4-3 A Comparison of FRED Sensor with Other Sensor Designs – Resource Utilisation

Sensor Design	Number of LUTs Utilised	RO-based Segment Stages	Target FPGA
[30]	87	31	Virtex-5
[32]	34	7	Virtex-1
[29]	24	7	Spartan-6
[27]	8	3	Virtex-5
[28]	5	3	Virtex-5
FRED (Proposed)	***8***	***21 (less random variation than above)***	***Spartan-7***

Table 4-4 A Comparison of FRED Sensor 'Sensitivity' with Other Sensor Designs.

Sensor Design	FPGA	Process (nm)	Voltage (V)	Sensitivity %/ºC
[31]	Virtex-4	90	1.2	0.11
[33]	Virtex-5	65	1	0.036
[27]	Virtex-5	65	1	0.032
[28]	Virtex-5	65	1	0.098
FRED (Proposed)	***Spartan-7***	***28***	***1***	***0.15***

enabled an improved sensitivity up to 0.15%/ºC for a 28 nm technology node. This helps detect very small changes in the frequency/delay degradation under nominal as well as stressed N/PBTI conditions.

4.5.2 Area Overhead

The authors in [28] have suggested a calculation method to determine the area overhead as $A_{OH} = N_S \times (A_{SS} + A_{Counter})$, where N_S represents the total number of sensors implemented in the FPGA, A_{SS} is the number of LUTs and FFs occupied by the RO-based sensor segments, and $A_{Counter}$ represents the number of LUTs and FFs utilised by the counter. For the proposed FRED sensor with three different sensor segments' lengths (9, 21, and 31 stages/gates) implemented over the entire FPGA as per the floorplan shown in Figure 4-11 gives the figures of 192 units, 384 units, and 544 units respectively.

4.5.3 Power Overhead

The resources occupied by RO-based sensor segments and the counters as well as the ROs' oscillations result in a considerable amount of power dissipation, that causes power overheads. Power overhead (P_{OH})is the differential average power consumption by an array of sensors and is calculated as follows:

$$P_{OH} = P_{withNet} - P_{withoutNet}$$

where, $P_{withNet}$ and $P_{withoutNet}$ are the average power consumed over the time interval *t* with and without embedding a sensor network in the FPGA, respectively, and are calculated using the following equation:

$$\boldsymbol{P_{with/withoutNet}} = \frac{1}{t}\int_0^t P(t).dt \qquad (4-5)$$

Table 4-5 gives a comparison of power overheads of four different sensor configurations with the same number of stages (21). It is evident that the ***P*** $\boldsymbol{P_{OH}}$ in the case of both the INV and NAND configuration is not much different. However, there is a significant difference between the NAND and NOR configuration of the FRED sensor segments. Based upon these performance evaluation metrics, three main aspects of the FRED sensor can be summed up as follows:

- **The Sensor Segment (SS) stage/length**: it implies that the longer the RO/SS is, 1) the more resources are occupied, and hence the $\boldsymbol{A_{OH}}$ (Area Overhead) increases, 2) the oscillation frequency drops and this decreases the thermal and power overheads, 3) the sensitivity reduces and increases frequency/delay mapping error.
- **The Sensor Segment Configuration**: it implies the type of combinatorial gates used to design the RO-based segment. Every gate has its peculiar oscillation profile. In the FRED sensor case, it is found that NAND gates are more sensitive as well as area and power efficient when compared to NOR/XOR/INV-gate configuration (as elaborated in Section 4.3.2.2.2).
- **The Number of sensors implemented**: the best proposition is to have more embedded sensors for enhanced accuracy, wider coverage, and an effective and efficient detection of frequency degradation/delay variability across the full spectrum of the FPGA fabric. It leads to increased area, power, and thermal overheads. On the other hand, if the number of sensors is less, there will be

Table 4-5 Comparison of the Normalised Area and Power Overheads of Four different RO Segment configurations

RO Segment Configuration	Area Overhead	Power Overhead
INV	0.41	0.97
XOR	0.41	1.00
NOR	0.48	1.07
NAND (Proposed)	***0.35***	***0.95***

less $\boldsymbol{A_{OH}}$ and $\boldsymbol{P_{OH}}$, however, there may be a compromise on the accuracy of detection and measurement coverage.

4.5.4 Quality Factor

In order to solve this conundrum, Quality Factor (QF) offers an efficient metric to combine the multiple criteria and observe it as a single metric to assess trade-offs between the above-mentioned evaluation metrics. Accordingly, the product of $\boldsymbol{A_{OH}}$ and $\boldsymbol{P_{OH}}$ is weighed to prove the significant relevance and authenticity of power-delay product (PDP) and energy-delay product (EDP) as follows:

$$QF = \frac{1}{\boldsymbol{A_{OH}} \times \boldsymbol{P_{OH}}} \tag{4-6}$$

Therefore, based on the QF metric, the efficiency of the three FRED sensor segments' configurations is determined and shown in Table 4-6. As can be seen, the FRED sensor with 21-stage NAND configuration and QF = 4.738 provides the most viable sensor segment design to accurately and efficiently detect variations in RO frequency and hence, delays particularly under N/PBTI stressed environment.

4.5.5 Some Limitations

With the proposed FRED sensor design, it may not be possible to achieve highest accuracy, especially in detecting malicious circuit activities that incur minor shifts in threshold voltages of N/PFETs. The intensity of these shifts resultantly causes ageing followed by the frequency and delay degradation. We observed that only half of the gates (inverters/NAND/NOR) in the dynamic sensor segment (DSS - stressed) experience N/PBTI stress during one oscillation cycle. As a result, the probability of false positives may arise that eventually affects the detection accuracy. However, this can be overcome by devising a method whereby all the gates of sensor segments

Table 4-6 A Normalised Comparison of the Quality Factor of NOR and NAND Sensor Segment Configurations for Different Segment Lengths.

Sensor Segment Length	Sensor Segment Configuration	Quality Factor
9	NOR	3.050
9	NAND	3.800
21	NOR	4.57
21	***NAND (Proposed)***	***4.738***
31	NOR	4.377
31	NAND	4.500

experience stress during the normal operation. One of the design options could be to break the connection of each gate to its prior one and connect them to the ground.

4.6 Summary

The **FRE**quency **D**egradation sensor (**FRED**) detects and provides a measure of decrement in the frequency of its uniquely designed dual delay-line based segments with the ageing of FPGA primitives due to the BTI degradation mechanisms. The fixed sensor segment (FSS) of FRED is used as a reference with near-zero stress and the dynamic sensor segment (DSS) is built to experience high temperature and voltage stresses. Configured with variant gate length and types, the sensor outputs an accurate measure of the frequency difference between the FSS and DSS segments. Due to near-zero stresses on FSS, it is equally good for the calibration of the sensor, which helps maintain the measurement accuracy. The simulation and real-time experiments under normal and accelerated temperature and voltage conditions validate the effectiveness of the sensor in detecting and measuring small to large delay variability by observing changes in the frequency difference between the FSS and DSS segments.

With low area and power overheads, high Quality Factor (QF = 4.738) , and sensitivity of up to 0.15%/ºC the FRED is a viable light-weight on-chip sensor option for modern FPGAs. Based on its capability to capture FPGA ageing in terms of the frequency and delay degradation across the whole FPGA surface, FRED can be considered a good candidate for the detection of malicious circuits, called hardware Trojans, that are based on the parametric variations (such as threshold voltage) of transistors.

Taking lead from the FRED sensor, the next chapter builds a complete threat scenario and devises an all-encompassing 'FPGA Security Scheme.' The limitations of FRED, as mentioned in Section 4.5.5, are addressed and is enhanced to cater for minor shifts in the threshold voltages of PMOS transistors under the influence of threshold-voltage triggered hardware Trojans.

REFERENCES

[1] H. Nguyen and Intel. Corporation, “Resiliency Challenges in Future Communications Infrastructure,” *IEEE Communications Quality and Reliability Workshop*, May 14, 2014.

[2] S. Khan and S. Hamdioui, “Temperature Dependence of NBTI Induced Delay,” *IEEE 16th Int. On-Line Test. Symp.*, pp. 15–20, 2010.

[3] J. M. Levine, E. Stott, G. A. Constantinides and P. Y. K. Cheung, "SMI: Slack Measurement Insertion for online timing monitoring in FPGAs," *23rd International Conference on Field programmable Logic and Applications*, Porto, pp. 1-4, 2013.

[4] D. Harris and S. Naffziger, "Statistical clock skew modeling with data delay variations," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 6, pp. 888-898, Dec. 2001.

[5] D. Ernst *et al.*, "Razor: a low-power pipeline based on circuit-level timing speculation," *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, San Diego, CA, USA, pp. 7-18, 2003.

[6] K. M. Zick and J. P. Hayes, “On-Line Sensing for Healthier FPGA Systems,” *Proc. 18th Annu. ACM/SIGDA Int. Symp. F. Program. Gate Arrays, Assoc. Comput. Mach. New York, NY, USA*, pp. 239–248, 2010.

[7] T. Rahkonen and J. Kostamovaara, "The use of stabilized CMOS delay lines in the digitization of short time intervals," *1991., IEEE International Sympoisum on Circuits and Systems*, Singapore, vol.4, pp. 2252-2255, 1991.

[8] J. V Hatfield, P. Dudek, S. Member, and S. Szczepan, “A High-Resolution CMOS Time-to-Digital Converter Utilizing a Vernier Delay Line,” in *IEEE Journal of Solid-State Circuits*, vol. 35, no. 2, pp. 240–247, 2000.

[9] J. Tschanz *et al.*, “On-Line Detection and Correction of Errors Due to Fast , Dynamic Voltage Droop Events,” *In Proceedings of Silicon,* pp. 1–4, 2010.

[10] J. Tschanz, K. Bowman, S. Walstra, M. Agostinelli, T. Karnik, and V. De. Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance. In Proc. Symposium on VLSI Circuits, pages 112–113. IEEE, 2009.

[11] J. Tschanz, K. Bowman, C. Wilkerson, S.-L. Lu, and T. Karnik, “Resilient circuits,” in ACM Intl. Conf. on Computer-Aided Design, pp. 71–73, 2009.

[12] J. M. Levine, E. Stott, G. A. Constantinides and P. Y. K. Cheung, "Online Measurement of Timing in Circuits: For Health Monitoring and Dynamic Voltage & Frequency Scaling," *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, Toronto, ON, pp. 109-116, 2012.

[13] K. Jerchel, A. Grams, N. F. Nissen, T. Suga and K. Lang, "Canary devices for through-silicon vias a condition monitoring approach," *2017 International Conference on Electronics Packaging (ICEP)*, Yamagata, pp. 282-287, 2017.

[14] P. Franco and E. J. McCluskey, “Delay Testing of Digital Circuits by Output Waveform Analysis,” in Proc. IEEE Intl. Test Conf., pp. 798-807, Oct. 1991.

[15] S. Das, et al., “Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance,” IEEE J. Solid-State Circuits, pp. 32-48, Jan. 2009.

[16] K. A. Bowman, et al., “Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance,” IEEE J. Solid State Circuits, pp. 49-63, Jan. 2009.

[17] J. Keane, T. H. Kim, and C. H. Kim, “An on-chip NBTI sensor for measuring pMOS threshold voltage degradation,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no. 6, pp. 947–956, 2010.

[18] F. Elguibaly and M. W. El-Kharashi, "Multiple-input signature registers: an improved design," *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, PACRIM. 10 Years Networking the Pacific Rim, 1987-1997*, Victoria, BC, Canada, vol.2, pp. 519-522, 1997.

[19] Wong, Justin & Cheung, Peter, "Improved delay measurement method in FPGA based on transition probability", In Proceedings of 19th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, California, pp 163-172, 2011.

[20] M. Agarwal *et al.*, "Optimized Circuit Failure Prediction for Aging: Practicality and Promise," *2008 IEEE International Test Conference*, Santa Clara, CA, pp. 1-10, 2008.

[21] A. Amouri and M. Tahoori, “A low-cost sensor for aging and late transitions detection in modern FPGAs,” *Proc. - 21st Int. Conf. F. Program. Log. Appl. FPL*, pp. 329–335, 2011.

[22] V. Bexiga, C. Leong, J. Semião, I. C. Teixeira, J. P. Teixeira, and I. S. T. Tul, “Performance failure prediction using built-in delay sensors in FPGAs," 21st International Conference on Field Programmable Logic and Applications, pp. 3–6, 2011.

[23] S. Mishra *et al.*, “Predictive TCAD for NBTI Stress-Recovery in Various Device Architectures and Channel Materials,” *IEEE Int. Reliab. Phys. Symp.*, pp. 6A-3.1-6A–3.8, 2017.

[24] S. Pae *et al.*, “BTI reliability of 45 nm high-K + metal-gate process technology,” *2008 IEEE Int. Reliab. Phys. Symp.*, pp. 352–357, 2008.

[25] P. Swierczynski, M. Fyrbiak, P. Koppe, and C. Paar, “FPGA Trojans Through Detecting and Weakening of Cryptographic Primitives,” *IEEE Trans. Comput.*

Des. Integr. Circuits Syst., vol. 34, no. 8, pp. 1236–1249, 2015.

[26] P. H. Jones, J. Moscola, Y. H. Cho, and J. W. Lockwood, “Adaptive Thermoregulation for Applications on Reconfigurable Devices,” *International Conference on Field Programmable Logic and Applications*, Amsterdam, no. September, 2007.

[27] K. M. Zick and J. P. Hayes, “Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems,” *ACM Trans. Reconfigurable Technol. Syst.*, vol. 5, no. 1, pp. 1–26, 2012.

[28] N. Rahmanikia, A. Amiri, H. Noori, and F. Mehdipour, “Performance evaluation metrics for ring-oscillator-based temperature sensors on FPGAs: A quality factor,” *Integr. VLSI J.*, vol. 57, no. December 2016, pp. 81–100, 2017.

[29] M. Zagrabski, B. Wojciechowski, M. Nikodem, K. Krzysztof, and K. S. Berezowski, “Calibration of RO-based temperature sensors for a toolset for measuring thermal behavior of FPGA devices,” Microelectronics Journal vol. 45, pp. 1753–1763, 2014.

[30] C. Tradowsky, E. Cordero, T. Deuser, M. Hübner and J. Becker, "Determination of on-chip temperature gradients on reconfigurable hardware," *2012 International Conference on Reconfigurable Computing and FPGAs*, Cancun, pp. 1-8, 2012.

[31] A. H. Ajami, K. Banerjee, S. Member, and M. Pedram, “Modeling and Analysis of Nonuniform Substrate Temperature Effects on Global ULSI Interconnects,” in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 849–861, 2005.

[32] López-Buedo, Sergio & Boemo, E., "Making visible the thermal behaviour of embedded microprocessors on FPGAs. A progress report". In Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays. 12. pp 79-86, 2004.

[33] P. Mangalagiri, S. Bae, R. Krishnan, Y. Xie, and V. Narayanan, “Thermal-aware reliability analysis for platform FPGAs,” *IEEE/ACM Int. Conf. Comput. Des. Dig. Tech. Pap. ICCAD*, pp. 722–727, 2008.

5 FPGA SECURITY SCHEME

This chapter is the mainstay of our research work constructed in-line with the integrated FPGA health management (IFHM) framework. The disposition of the chapter is depicted in Figure 5-1. Section 1 is an introduction to the threat environment FPGAs are exposed to due to the miniaturisation of technology nodes and ever-growing cyber and hardware attacks. Section 2 gives information on the related work with a brief critique. Section 3 delineates the FPGA Security Scheme with description about the design, simulation, and implementation of Threshold Voltage-triggered hardware Trojan, '***HT_{Vth}***,' in a 28 nm technology node FPGA. Section 4 presents the design and implementation of a Threshold Voltage-aware sensor (***S_{Vth}*** - the hardware Trojan Detector) and discusses various options tested to achieve high sensor accuracy. In Section 5, the mitigation technique based on online transistor dynamic scaling (auto-resizing) and its correlation with NBTI-induced performance degradation are highlighted. Section 6 puts forth the implementation and optimization of the HT mitigation scheme and provides its simplistic comparison with some of the state-of- the-art reliability and security solutions. And finally, Section 7 summarises the chapter.

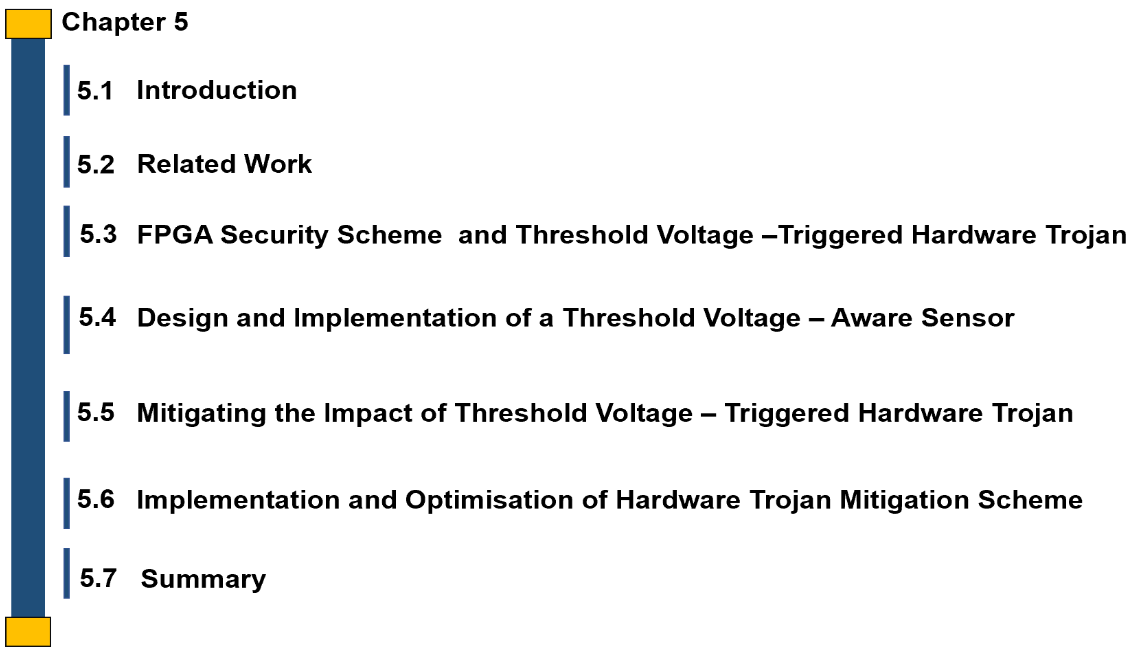

Chapter 5
5.1 Introduction
5.2 Related Work
5.3 FPGA Security Scheme and Threshold Voltage –Triggered Hardware Trojan
5.4 Design and Implementation of a Threshold Voltage – Aware Sensor
5.5 Mitigating the Impact of Threshold Voltage – Triggered Hardware Trojan
5.6 Implementation and Optimisation of Hardware Trojan Mitigation Scheme
5.7 Summary

Figure 5-1 The Disposition of Chapter 5.

5.1 Introduction

A modern FPGA is not merely an emulator but a hardware accelerator with heterogenous hard IP cores, such as complex memory blocks, multiple processors, and DSP blocks. Systems on chip (SoC), network on chip (NoC), and adaptive compute acceleration platform (ACAP) are the significant performance and functional enhancements of FPGAs, that have been made possible due to the continual shrinking of transistor sizes down to the scales of 10 nm and below. The performance benefits, however, are limited by power and timing closures. Similarly, the geometric structures of FPGAs with much less silicon and relatively more oxide and moulding compound complicate the heat conduction paths [1]. On the one hand, where it may deteriorate the worst-case heat dissipation route, a given power density, on the other hand, produces a significant temperature variability [2]. This results in a higher temperature for the same amount of power dissipation. It is, therefore, essential to consider thermal variation as an on-going challenge for advanced technology nodes alongside the associated issues of power and timing closures.

Looking at the FPGA fabric, we find a mesh of layers comprising a substrate, high-k dielectric interfaces, and metal interconnects. Each layer has a varying range of thermal conductivity with silicon dioxide sitting at 1.3-0.3W/mK, and copper metal interconnects going as high as 400 W/mK [3]. These differences in thermal conductivity affect the heat transfer and introduce variations in temperature across the FPGA area, thereby creating hotspots as can be seen in Figure 5-2. The resultant

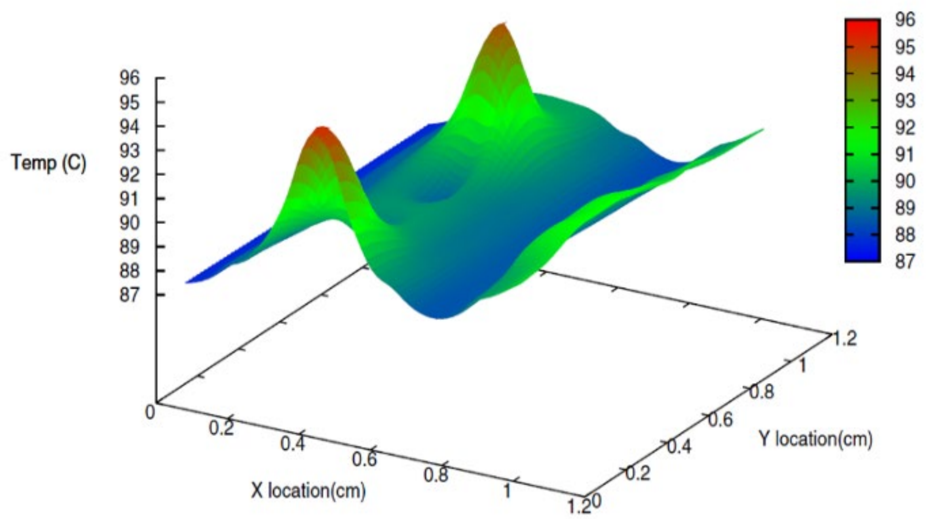
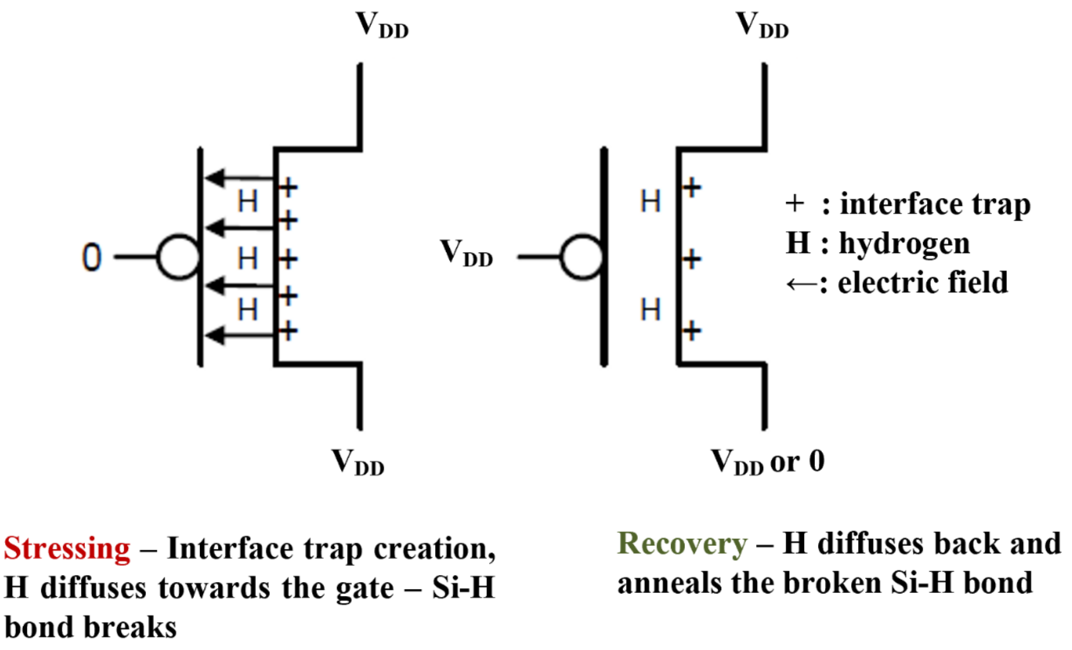


Figure 5-2 Thermal profile depicting hotspots in an FPGA [4].

increase in temperature and appearance of hotspots across the FPGA surface causes non-negligible variations in the timing and power domains of the design [4]. This non-uniform thermal dissipation aggravates the ageing mechanism of negative bias temperature instability (NBTI) and leads to accelerated ageing of the FPGA fabric.

The NBTI ageing mechanism is dominated by a negative shift in threshold voltage (***V_{th}***) of pMOSFETs that make up the FPGA, along with nMOSFETs. The change in threshold voltage is in response to biasing in the strong inversion region, which causes the disintegration of Si-H bonds at the oxide interface due to the presence of holes within the pMOS inversion layer, as is evident in Figure 5-3. This bond disintegration process creates positively charged interface traps, which, along with new or existing traps within the oxide, increases the threshold voltage [5], [6], [7].

Undeniably, NBTI is well known to researchers and manufacturers alike as a dominant ageing mechanism in all different configurations of integrated circuits (ICs). For instance, in the post-IC manufacturing period of 7 to 10 years, accelerated ageing due to NBTI has been reported by [5] and [8] as degradation in threshold voltage up to 50 mV. Speed degradation (of 20%) follows these shifts in threshold voltage and, therefore, shows a strong correlation between NBTI prompted delay and threshold voltage shift.



V_{DD}
V_{DD}
0
H
H
H
V_{DD}
H
H
+ : interface trap
H : hydrogen
←: electric field
V_{DD}
V_{DD} or 0
Stressing – Interface trap creation, H diffuses towards the gate – Si-H bond breaks
Recovery – H diffuses back and anneals the broken Si-H bond

Figure 5-3 NBTI mechanism in a PMOS transistor.

It is important to note that the non-uniformity of NBTI (*due to different thermal conductivity patterns*) across the chip surface affects various blocks within the FPGA differently. As a result, the delay variations induced by NBTI, across the FPGA surface, could potentially generate new critical paths, which, in turn, may prevent an efficient and balanced timing closure [9]. In the case of data paths, for instance, an increase in gate delays causes a late transition of an input signal at the flip-flop. Such varying transitions violate the flip-flop setup and hold time that eventually results in the sampling of flawed values at the output of the data path.

These variations, apart from being the primary source of FPGA reliability concerns, also affect the integrity of logic applications and aggravate to levels that may lead to system failures. More alarming is the **hardware security threat** that can leverage the dwindling reliability of an FPGA device under NBTI influence. It can jeopardise FPGA's optimal performance with the insertion of malicious and stealthy circuitry, called hardware Trojan – designed by exploiting stochastic and systematic variation patterns that exist within the FPGA.

The exacerbation of NBTI, owing to the continual transistor miniaturization, is fast becoming a major donor of the process of ageing in downscaled technology nodes. It poses a challenge for the proponents of high FPGA reliability and performance to understand the dynamics of NBTI in designing a hardware Trojan, initially, from an intruder's (*a rogue element*) perspective and lately by designing a threshold voltage-aware sensor for its detection, followed by an effective mitigation methodology from security assurance and defender's perspective.

In other words, it implies the development of an **FPGA Security Scheme** (Figure 5-4), which assumes that an intruder is capable of capturing and analysing the shifts in threshold voltage of pMOSFETs (*that result in lowering the frequency, signal path delay variations, and flawed transitions*) due to the NBTI effect. If successful, the intruder may design and insert a stealthy malicious circuit (*called hardware Trojan*) inside the FPGA. With sufficient parametric information and precise monitoring, the intruder may capitalize NBTI ageing mechanism to activate a dormant hardware Trojan. This is further elaborated in the threat model described in Section-2.

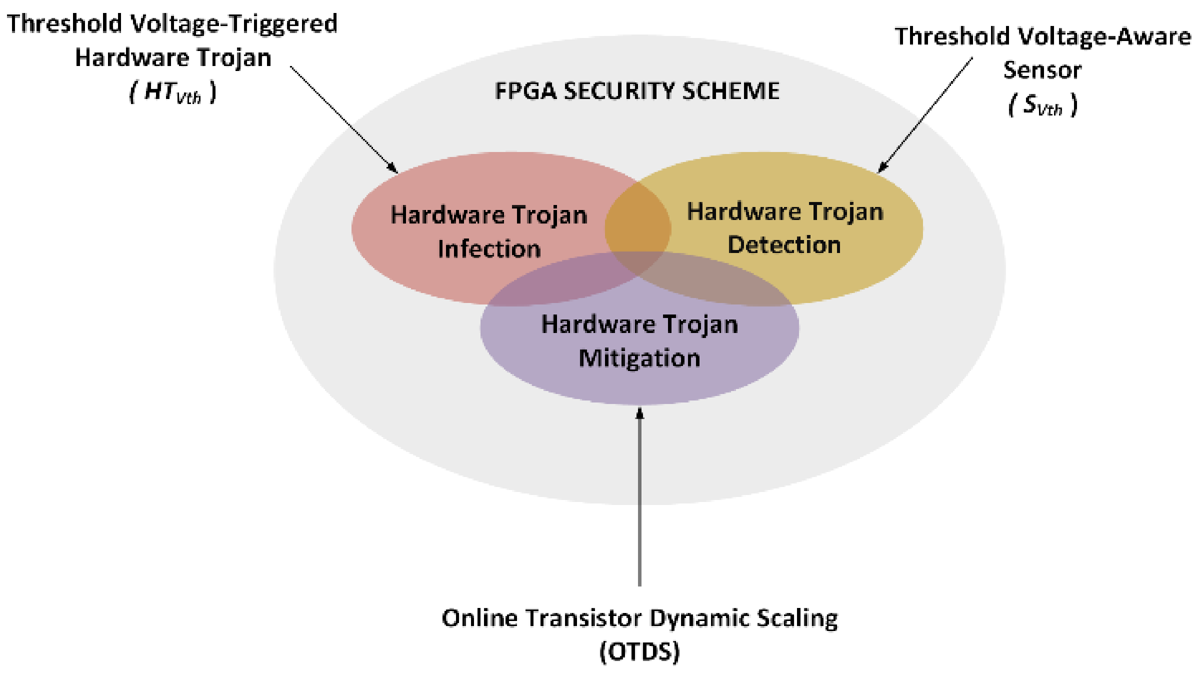

Threshold Voltage-Triggered Hardware Trojan (HT_{Vth})
FPGA SECURITY SCHEME
Threshold Voltage-Aware Sensor (S_{Vth})
Hardware Trojan Infection
Hardware Trojan Detection
Hardware Trojan Mitigation
Online Transistor Dynamic Scaling (OTDS)

Figure 5-4 FPGA Security Scheme comprising hardware Trojan Infection, Detection, and Mitigation sub-schemes.

It is well established that the detection of such hardware Trojans is difficult using testing techniques like built-in self-test (BIST) because no test vector can activate an ageing effect [10]. The process of accelerated stress and ageing test on the affected node may, however, reveal such Trojans. On the contrary, the process, when performed on a complete integrated circuit, is time and cost-intensive [11].

In this chapter, we direct the FPGA security scheme, shown in Figure 5-4, towards the design and implementation of a threshold voltage-triggered hardware Trojan in a lower technology node (*28 nm FPGA*). A degradation in the drain current, oscillation frequency, and the subsequent increase in the response time (*due to shift in threshold voltage*) of the 28 nm FPGA is observed through a novel sensor. An effort is also made to mitigate the impact of a hardware Trojan by introducing a method of compensation that enhances the current flow and lowers the rise in delay due to NBTI. This includes an online transistor dynamic scaling (OTDS) approach as a mitigation methodology to counter hardware Trojans.

The proposed designs and implementations are verified and validated using post-layout, and Monte Carlo simulations with Cadence Virtuoso ADE tools, followed by real-time experiments on Global Foundries fabricated 28 nm technology node. Threshold voltage-triggered hardware Trojan, '***HT_{Vth}***,' operates in a threshold voltage region of 0.45V-0.998V, consuming ultra-low power (10.5nW), and remaining stealthier within an area overhead of as low as 1.5%. The Threshold Voltage-aware sensor, *‘$\boldsymbol{S_{Vth}}$,’* utilizes 3% of die resources and achieves the detection sensitivity of 0.251mV/nA. OTDS enables the auto-resizing of transistors to mitigate the impact of hardware Trojan payload due to NBTI-based threshold voltage shifts falling between 10% and 90%.

5.2 Related Work

Extensive research has been undertaken to present a detailed analysis of ageing and performance degradation in integrated circuits. It mainly involves the fingerprinting of ICs’ electrical parameters (voltages, currents, frequencies, and EM signals) by retrofitting well-designed on-chip sensors and structures. Be it the detection of counterfeits, recycled ICs, or detection and mitigation of hardware Trojans; the same parameters are manipulated by researchers to understand different undesired behaviour patterns and anomalies in ICs (ASICs, FPGAs, and Microprocessors) for remediation and building effective countermeasures.

In [12], Karhunen Loéve theorem is used to study the power consumption behaviour of hardware Trojan infected FPGA to determine the possibility of its detection. This technique considers the impact of process variations that occur within the FPGA; however, it avoids the noise factor and is limited to simulation analysis. Similarly, the researchers in [13] have again simulated and analysed the occurrence of path delays in the signals of various logic applications using the embedded monitors. Both of these techniques do not provide real-time analysis. An integrated hardware system capable of monitoring the behaviour of critical interconnects (*wires*) is proposed in [14]; however, it does not provide sufficient information on the efficiency of this method. In [15], a test methodology to ease hardware Trojan triggering by increasing its electrical activity is proposed for early detection. In [16], an attempt to carry out precise measurement of an IC’s operating frequency, maximum frequency (f_{max}), and its dynamic power consumption is made by lowering the impact of process variations. However, the calculation of the accurate value of f_{max} is quite challenging and also susceptible to ‘false positives.’

The use of ring oscillators’ sensitivity to variations in temperature and power enables the detection of medium-to-heavyweight hardware Trojans, however, not effective against the small-sized/lightweight hardware Trojans [17]. The researchers in [18] have created a network of ring oscillators spread across the FPGA surface to capture the changes in their oscillation frequency due to the presence of hardware Trojan. This is validated using a digital storage oscilloscope (DSO) and later analysed using the principal component analysis to differentiate between the genuine and the HT infected FPGA. However, when applied to an ASIC [19], this technique suffers from the lower levels of measurement accuracy due to the usage of an 8-bit counter instead of a digital storage oscilloscope, questioning the accuracy of on-chip designs.

In [20], the clustering methodology is proposed, whereby dedicated sensors are embedded in the power grids of different voltage islands in FPGA, to enhance HT detectability. However, it does not provide adequate experimental evidence to evaluate the efficacy of this methodology. The capturing of electromagnetic signatures of target applications in ICs has also been studied for hardware Trojan and anomaly detection. For instance, a method based on electromagnetic (EM) cartography is proposed in [21], but then again, due to inappropriate method of interpretation of EM traces, the detection of hardware Trojans remains low. On the other hand, in [22], the researchers have devised an improved technique that interprets the EM traces optimally. By controlling and maintaining the temperature during EM measurements, this technique improves the probability of detecting lightweight hardware Trojans. Further to this, the researchers in [23] are able to differentiate between the healthy and HT infected population of FPGAs through a comprehensive analysis of EM signatures.

A reasonable amount of work has also been undertaken to design and develop various sensing techniques and frameworks for the detection and mitigation of the NBTI mechanism and its noticeable impact. In [24], an analog supply-devoid NBTI sensor is proposed to eliminate noise; however, the input of other external signals makes its operation very complicated during the stress and recovery as well as measurement modes. This reduces its overall measurement accuracy. In [25], the dynamic reliability of the device is managed using NBTI and HCI (Hot Carrier Injection) sensors. In this case, the threshold voltage of the stressed device is measured and transformed into the delay function. However, these sensors are less sensitive to temperature variations and occupy large device area with high power consumption. In another study [26] an NBTI sensor is designed to measure the standby leakage current (I_{ddq}).

Designed explicitly for SRAM cells, this sensor monitors the leakage current, characterising temporal degradations. It, however, requires an additional bias generator to maintain active load on the sensor, which results in non-linearity and reduced sensitivity to the input signal. Researchers in [27] have used the current-mirroring technique to capture NBTI based degradation. The power supply current is mirrored and subsequently transformed into voltage. The drawback of this approach lies in the usage of power gating that slows down the response time of the sensor. However, its performance is relatively more stable than the $\boldsymbol{I_{ddq}}$ based sensor.

To mitigate NBTI ageing and degradation impact on the reliability and performance of an IC, we have come across the concept of one-time design constraints put forth by various researchers. For instance, [28], [29] suggest an increase in supply voltage to manage and control NBTI. This may, however, lead to power and thermal overheads – an undesirable design feature. Whereas [30] and [31] propose transistor oversizing and reduction in the clock frequency, respectively as an optimum NBTI mitigation. The thermal management of ICs via different cooling arrangements is also proposed to contain and reverse the NBTI impact [32]. Gate replacement technique is proposed in [33] that attempts to optimize the NBTI ageing effect. Techniques on the balancing and removal of stress to control short-duration threshold voltage instability are suggested by Choi et al. [34]. These, however, fail to consider the critical factor of prolonged ageing effect at high temperatures. In [35], Kiamehr et al. have highlighted the use of ageing-aware library standard cells to mitigate BTI impact on the rise and fall times of different signals. The threshold voltage shift is, initially, measured and later used to optimize the width ratio (W_p/W_n) of each transistor to counter the ageing effect. However, its applicability for IC run-time is not considered. Another study by Zhang et al. [36] describes the techniques that involve the identification of critical gates and their replacement with NBTI-tolerant gates. The use of dynamic voltage scaling and data flipping has also been proposed by [37] to recover the static noise margin in the case of SRAMs.

The measurement of a beat frequency between the reference and stressed ring oscillators using a silicon odometer is also proposed in [38] to keep track of degradation due to NBTI. Similarly, a hybrid scheme comprising ring oscillators and delay line based online-ageing monitoring is presented in [39] for the measurement of degradation. These sensor schemes are, however, focused on ensuring precise measurements rather than triggering accelerated degradation to detect the presence of any notable anomaly. In order to fill in this gap, a low-cost and lightweight structure consisting of ring oscillator based sensors for in-field capturing of IC/FPGA ageing is proposed in [40] to enhance the granularity of detection.

More recently, authors in [41] have proposed a multitype hardware Trojan protection framework, called RG-Secure. This framework is designed and validated to provide RTL and gate-level security to FPGA based SoCs (*deployed in IoT environment*) against different types of hardware Trojans by merging 3PIP (third party intellectual property) trusted design approaches with the scan-chain netlist feature analysis. Employing tree-based learning algorithms, they have shown a good hardware Trojan detection coverage at RTL and gate-levels, with 100% true positive rate and 94% true negative rate accuracies. In our opinion and analysis, this method/framework holds true for less complex netlist structures and scan-chain features. However, it may not be effective against parametric hardware Trojans (*e.g., threshold voltage-triggered*) that have netlists of distinct structure and trigger behaviour.

Our work, however, follows an integrated approach, as mentioned earlier, and encompasses three elements namely, HT insertion (*infection*), its detection, and mitigation. We build these elements considering the limitations and strengths of the abovementioned techniques and different on-chip sensors' architectures, with FPGA security and reliability in perspective.

5.2.1 Threat Model

Hardware Trojan, a stealthily malicious entity, capable of inflicting performance degradation, sensitive information disclosure, and functional disorder at the micro-architectural level in FPGAs, continues to challenge the efforts toward strengthening hardware security. In an attempt to control its increasing threat, we construct a threat scenario/model to understand its implications for a high-end defence asset - a naval warship, fitted with an 'Integrated Self-Protection System' (ISPS) and eventually develop a full-spectrum FPGA security scheme.

ISPS is a real-time functional integration of electronic warfare systems used onboard naval warships and fighter aircraft as well. It comprises Electronic Support Measures' (ESM) systems like Radar Warning Receivers (RWR), System Processor for threat environment assessment and asset assignment, and Electronic Counter Measures' (ECMs) systems like Jammers and Chaff launchers.

We, however, focus on System Processor Module and regard it as a vulnerable entity in ISPS system due to its high probability of infection with security-compromised FPGAs. The threat scenario, as depicted in Figure 5-5, has three main elements, namely: 1) the naval warship, 2) the Defence Systems Manufacturer - X, and 3) the FPGA Supplier - Y. The red sphere with letter 'R' represents the 'Rogue Element' that could be working with malicious intentions on its own, as a state-sponsored VLSI design specialist, or an anti-state element/enemy. We assume its presence at FPGA Supplier premises in 'Design House,' 'Fabrication Facility,' and 'SoC Integration Section' - all representative of the FPGA supply chain. The green sphere with the letter 'D' represents the authors' recommendation on forming a 'Security Assurance and

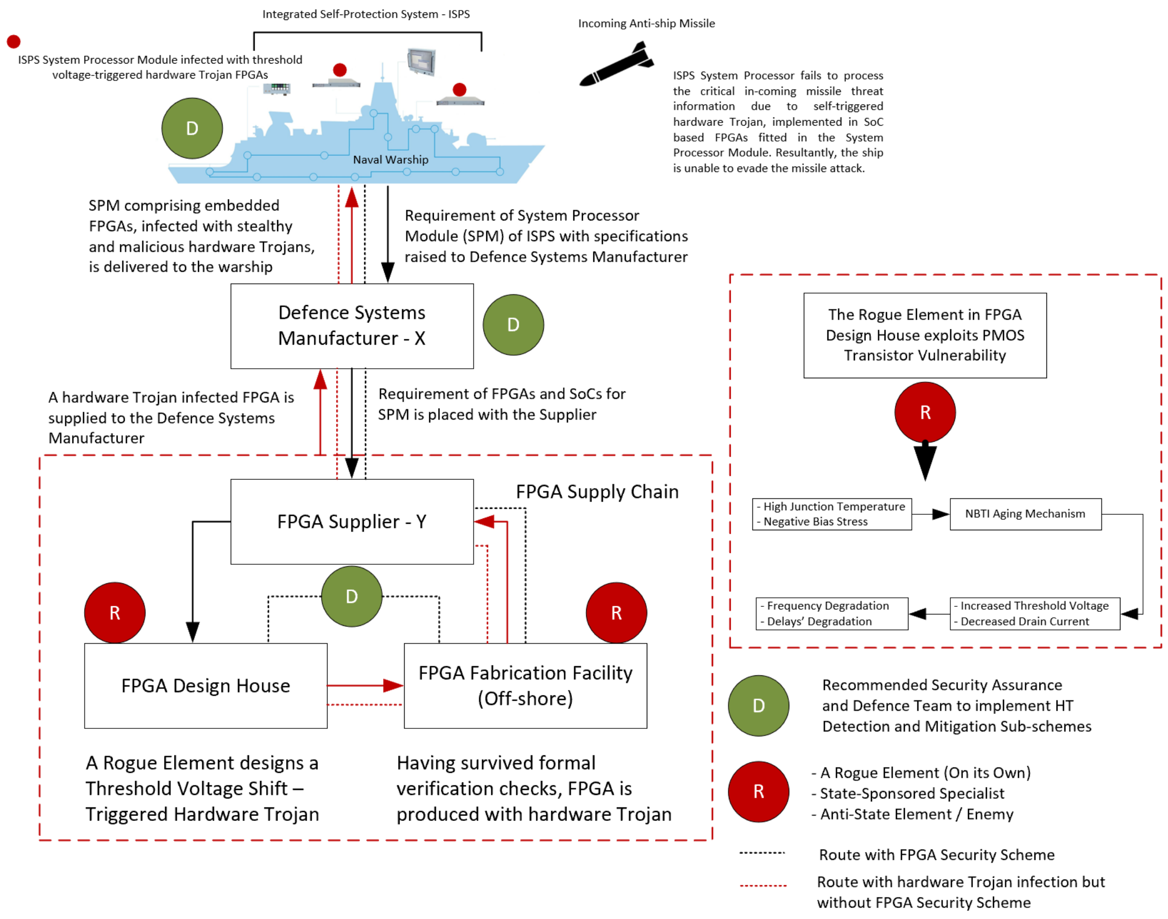

Integrated Self-Protection System - ISPS
Incoming Anti-ship Missile
ISPS System Processor Module infected with threshold voltage-triggered hardware Trojan FPGAs
D
Naval Warship
ISPS System Processor fails to process the critical in-coming missile threat information due to self-triggered hardware Trojan, implemented in SoC based FPGAs fitted in the System Processor Module. Resultantly, the ship is unable to evade the missile attack.
SPM comprising embedded FPGAs, infected with stealthy and malicious hardware Trojans, is delivered to the warship
Requirement of System Processor Module (SPM) of ISPS with specifications raised to Defence Systems Manufacturer
Defence Systems Manufacturer - X
D
A hardware Trojan infected FPGA is supplied to the Defence Systems Manufacturer
Requirement of FPGAs and SoCs for SPM is placed with the Supplier
FPGA Supply Chain
FPGA Supplier - Y
R
D
R
FPGA Design House
FPGA Fabrication Facility (Off-shore)
A Rogue Element designs a Threshold Voltage Shift – Triggered Hardware Trojan
Having survived formal verification checks, FPGA is produced with hardware Trojan
The Rogue Element in FPGA Design House exploits PMOS Transistor Vulnerability
R
- High Junction Temperature
- Negative Bias Stress
NBTI Aging Mechanism
- Frequency Degradation
- Delays' Degradation
- Increased Threshold Voltage
- Decreased Drain Current
D
Recommended Security Assurance and Defence Team to implement HT Detection and Mitigation Sub-schemes
R
- A Rogue Element (On its Own)
- State-Sponsored Specialist
- Anti-State Element / Enemy
Route with FPGA Security Scheme
Route with hardware Trojan infection but without FPGA Security Scheme

Figure 5-5 Threat Model: A novel self-triggered Threshold Voltage-Shift based Hardware Trojan '*HT*$_{Vth}$' is designed and implemented by a rogue element in a 28 nm FPGA used in System Processor Module of ISPS (Integrated Self Protection System) of a Naval Warship.

Defence Team' to counter the malicious insertion in FPGA by the rogue element. Its presence is recommended in all the three elements.

The threat process begins with the naval warship placing the requirement of a new System Processor Module (installed with n-number of FPGAs, providing vital electronic warfare functions) for the ISPS system from the Defence Systems' Manufacturer-X. Subsequently, the FPGA supplier -Y is sub-contracted by X to provide FPGAs built on 28 nm process technology. A rogue element R, stationed in a Y design house, receives the task of designing the FPGA. Here, we assume that R is an expert FPGA designer with sufficient working knowledge of FPGA design flow, specific to the insertion of stealthy hardware Trojan based on the threshold voltage shifts in PMOS transistors. Such type of hardware Trojans corresponds to the functionality level parametric characterization [42] and are targeted at paralysing device/system functionality. To maintain undetectability, R employs **'Split hardware Trojan Insertion'** methodology, whereby a part of a hardware Trojan circuit is built at the design stage in the design house. Post design and successful simulation, the design file (GDSII) is forwarded to the FPGA fabrication facility for manufacturing. Here, the remaining part of hardware Trojan is added (at the RTL and Gate level) post-manufacturing reliability tests by another rogue element (collaborator) at the FPGA fabrication facility to evade detection. As per our recommendation (mentioned in Figure 5-5), if D is also stationed at the design house, it will design detection and mitigation circuitry in addition to the hardware Trojan circuit design by R (with both D and R remaining oblivious of each other's work). The newly fabricated chips are now ready for installation on the system processor module at X. The security assurance and defence team D at X carries out pre-installation security tests to observe anomalies specific to hardware Trojan based on threshold voltage shifts. If the tests are clear, the FPGA is installed on the system processor module and delivered to the end-user - the naval warship. At this point, we make two assumptions. Firstly, if the detection and mitigation circuitry fails and the hardware Trojan gets triggered, the damage to ISPS operation ability will occur. Secondly, if the detection and mitigation circuitry successfully detects and mitigates the hardware Trojan, the ISPS system will continue performing efficiently without any hindrances, provided some other faults that are not related to hardware Trojan erupt. As can be seen in Figure 5-5, we have also recommended the placement of D in the naval warship. So, before installing the system processor module in the ISPS system for harbour and sea acceptance trials (HATs and SATs), D must carry out security tests to challenge the first assumption and in case of it holding true, return the module to X for replacement.

In a nutshell, as shown in Figure 5-5, if the 'red-dotted line' route (*containing the FPGA infected with hardware Trojan but without any detection and mitigation component of FPGA security scheme*) is adopted, the hardware Trojan would remain undetected and get triggered with pre-defined threshold voltage shift, thereby causing ISPS system performance degradation and leaving the ship vulnerable to a devastating missile attack. On the other hand, if the 'black-dotted line' route (*containing a robust FPGA security scheme*) is assumed, the hardware Trojan can be easily detected and denied triggering, thereby keeping the ISPS system proficient in thwarting any external threat to the ship.

Considering the above threat scenario/model, we, in the following sections, make an effort to sequentially develop a realistic FPGA security scheme for the security assurance and defence team to not only provide security and dependable redundancy to critical systems like ISPS but also augment the post-manufacturing tests regime (*security tests, in specific*) employed by FPGA manufacturers. The first step, in this regard, is the design and implementation of a hardware Trojan itself, followed by detection and mitigation circuitries based on the Trojan's impact on target FPGA applications.

5.3 FPGA Security Scheme and Threshold Voltage – Triggered Hardware Trojan

In line with the FPGA security scheme (Figure 5-4), we define the contours of the hardware Trojan (HT)-infection scheme. It encompasses an operational system's FPGA (28 nm technology) vulnerable to ingress of hardware Trojan, which in turn, inflicts operational and functional damages to the system and its various components.

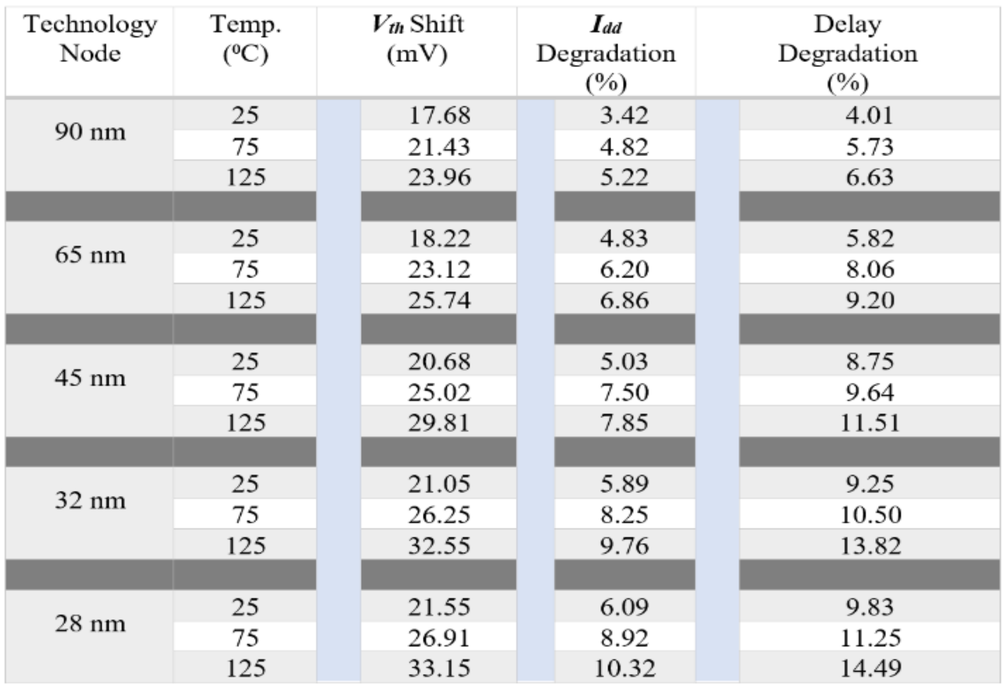
Beginning with HT-infection scheme, we construct a hardware Trojan with details as follows.

5.3.1 Design Considerations

As mentioned earlier, the high temperature activates the NBTI mechanism in the FPGA silicon fabric. Resultantly, it accelerates the process of ageing and leads to undesirable characteristics. For instance, temperature changes beyond 75ºC between different layers of a substrate could cause variations in interconnect delays up to 31-38% [43]. Subsequently, the device tends to operate slower with delays also observable in the control and data signals. Such timing inconsistencies cause synchronous circuits transit into redundant states or momentary glitches. However, to avoid failures, the clock period can be managed to counter the system glitches. The authors in [44] have, nevertheless, suggested that despite clock management, the period of momentary glitches tends to increase with NBTI and may set off pre-determined activity related to malicious circuitry.

Tabular analysis (Table 5-1) of the results obtained by [45] reveals that:(a) the shift in threshold voltage (***V_{th}***) and drain current (***I_{dd}***) is a function of high temperature and is observed to increase for ***V_{th}*** and decrease for ***I_{dd}*** at temperatures ≥ 60ºC, (b) an approximate rise of 4% in the threshold voltage shift is evident with the scaling down of technology nodes [46]. The rate of decrease in ***I_{dd}*** is, however, less than the rate of ***V_{th}*** increase, and (c) eventually, the propagation delays increase with the aforementioned trends of variation in ***V_{th}*** and ***I_{dd}***.

Table 5-1 Impact of NBTI aging mechanism on PMOS transistor parameters.



Technology Node	Temp. (ºC)	V_{th} Shift (mV)	I_{dd} Degradation (%)	Delay Degradation (%)
90 nm	25	17.68	3.42	4.01
	75	21.43	4.82	5.73
	125	23.96	5.22	6.63
65 nm	25	18.22	4.83	5.82
	75	23.12	6.20	8.06
	125	25.74	6.86	9.20
45 nm	25	20.68	5.03	8.75
	75	25.02	7.50	9.64
	125	29.81	7.85	11.51
32 nm	25	21.05	5.89	9.25
	75	26.25	8.25	10.50
	125	32.55	9.76	13.82
28 nm	25	21.55	6.09	9.83
	75	26.91	8.92	11.25
	125	33.15	10.32	14.49

In light of the above, the essential design targets for threshold voltage-triggered hardware Trojan (***HT**$_{Vth}$*) are set accordingly such that: (a) the transfer function of the Trojan circuit must be linear. (b) sensitivity to temperature and threshold voltage changes should be significantly high, (c) the change in the output should be significantly high for a change in the input, and (d) negligible temporal degradation and tolerance to process variations should be maintained.

Additionally, the element of stealthiness and undetectability of hardware Trojan is highly significant (*primarily from the perspective of a rogue element*). Hardware Trojan, by definition, has to be stealthy to escape detection. In order to achieve this, we have ensured during design and implementation stages (*described in the following sections*) that the size of the circuitry is as small as possible with equally low power consumption and without compromising the effectiveness of its payload. Regarding the area and resource utilization at the circuit and RTL/Gate level, we have used as minimum instantiation as possible to ensure low area and power overheads. These have been measured to be at just **1.5 %** of the total available resources on a 28 nm process technology. With such a small percentage, it is highly unlikely that the added circuitry of hardware Trojan would be discovered either during post-manufacturing tests or during run-time monitoring. Hence in a multi-million gates chip, it can hide easily. Also, more importantly, the proposed threshold voltage triggered Trojan does not draw any extra current while dormant; therefore, it becomes challenging even to detect it through power signature analysis.

5.3.2 Architecture of Threshold Voltage Triggered Hardware Trojan (HT$_{Vth}$)

We propose a circuit implementation of threshold voltage-triggered hardware Trojan, ***HT**$_{Vth}$*, which is valid for CMOS devices. The implementation is demonstrated for both the sequential and combinatorial logic as follows:

5.3.2.1 Conceptualising Hardware Trojan in Combinatorial Circuits

Considering the **combinatorial** circuit for hardware Trojan, a 2-input NAND gate is designed to have two PMOS transistors M1 and M2 parallel to one another. These are then connected in series to two NMOS transistors M3 and M4, as shown in Figure 5-6. The drain terminals of both M1 and M2 are shared and connected to the source terminal of M3. The output of the NAND gate is tapped out at M3. Another

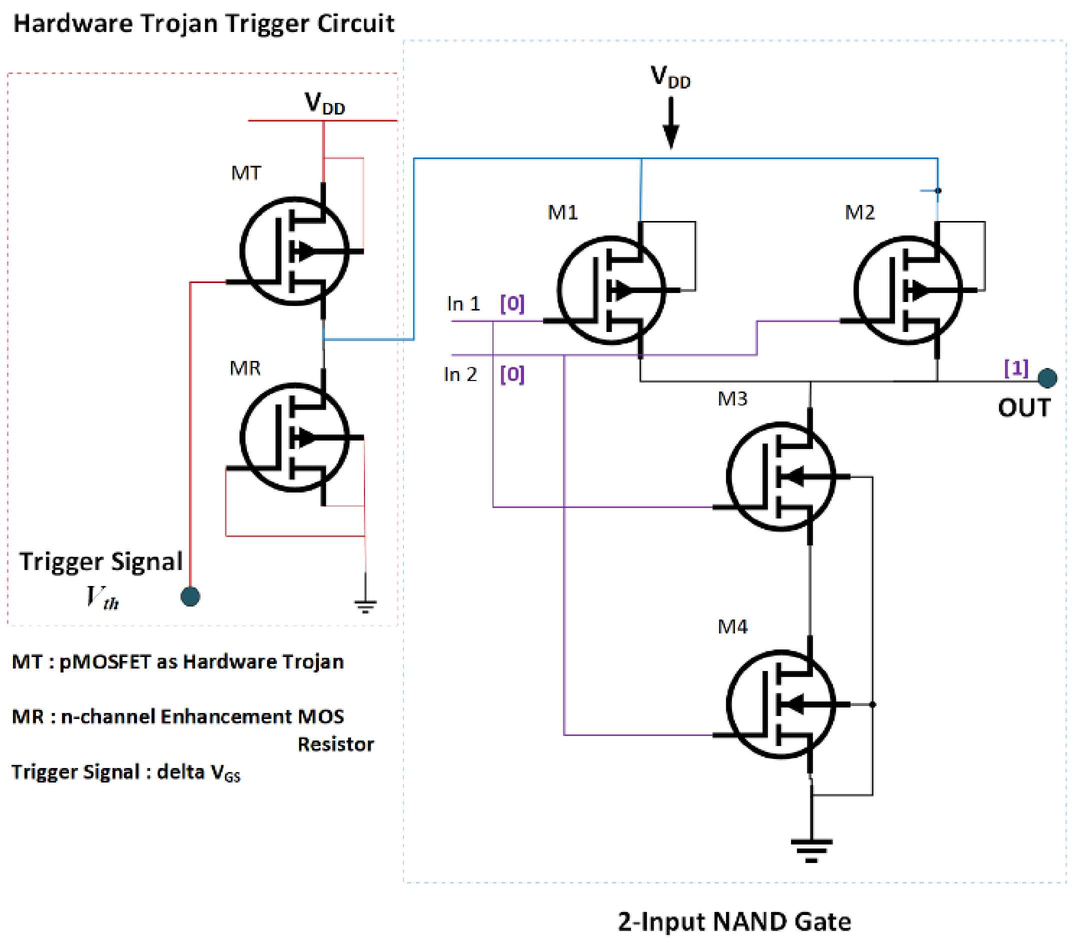

Hardware Trojan Trigger Circuit
VDD
MT
MR
Trigger Signal
Vth
MT : pMOSFET as Hardware Trojan
MR : n-channel Enhancement MOS Resistor
Trigger Signal : delta VGS
VDD
M1
M2
In 1 [0]
In 2 [0]
[1]
OUT
M3
M4
2-Input NAND Gate

Figure 5-6 Schematic of a threshold voltage-triggered hardware Trojan (*HT_{Vth}*) in a combinatorial circuit (2-input NAND gate).

PMOS transistor, MT (Trojan Transistor – occupying an area of **7.25 µm^2**), is constructed in series with a MOS resistor (MR) to work as a hardware Trojan. The MOS resistor acts as a current limiter as soon as the triggering signal is received at the MT gate terminal. A compact silicon area of **50µm^2** is occupied by this circuitry with a low power consumption of **1.05µW**.

Operationally, the Trojan is kept in the 'ON' stealthy state so that the transistors M1 and M2 remain connected to the power supply **(*V_{DD}*)**. The output of the NAND gate, on the other hand, is '0' when both of its inputs In 1 and In 2 are '1'. Otherwise, the output always remains at '1'. As MT, the hardware Trojan receives an NBTI induced shift in threshold voltage (triggering signal) at its gate terminal; it initiates the process of accelerated device ageing with elevated temperatures and reduced frequency of the NAND gate circuitry. The shift in the threshold voltage, which acts as a trigger for the hardware Trojan, needs to be measured very carefully. For this purpose, we have also designed a threshold voltage measuring circuit, termed as '**Threshold Voltage Meter**' (described in section 5.3.3). With the value of threshold voltage (**V_{th}**) exceeding the pre-defined level (***pre-Trojan Trigger Threshold Voltage- V_{th_ptt} – 0.45V***), a triggering signal is generated at the gate terminal of MT. This active high triggering signal switches the MT 'OFF' and leaves the PMOS transistors M1 and M2 without power, thereby affecting the operation of the NAND gate.

5.3.2.2 Conceptualising Hardware Trojan in 4-BIT Ripple Carry Adder

In this subsection, we have attempted to demonstrate how a threshold-voltage triggered HT impacts a typical logic function implemented in a 28 nm process technology. For this purpose, we consider building a full adder (4-BIT ripple carry adder), regarded as the main building block of arithmetic logic unit (ALU) which realizes a set of basic arithmetic operations, such as addition, subtraction, multiplication and division. The performance of a computation system depends on the efficiency of arithmetic operation executed by the full adder. A hardware Trojan when added to full adder can severely disrupt its operation and affect its performance as well as the overall reliability of FPGA device.

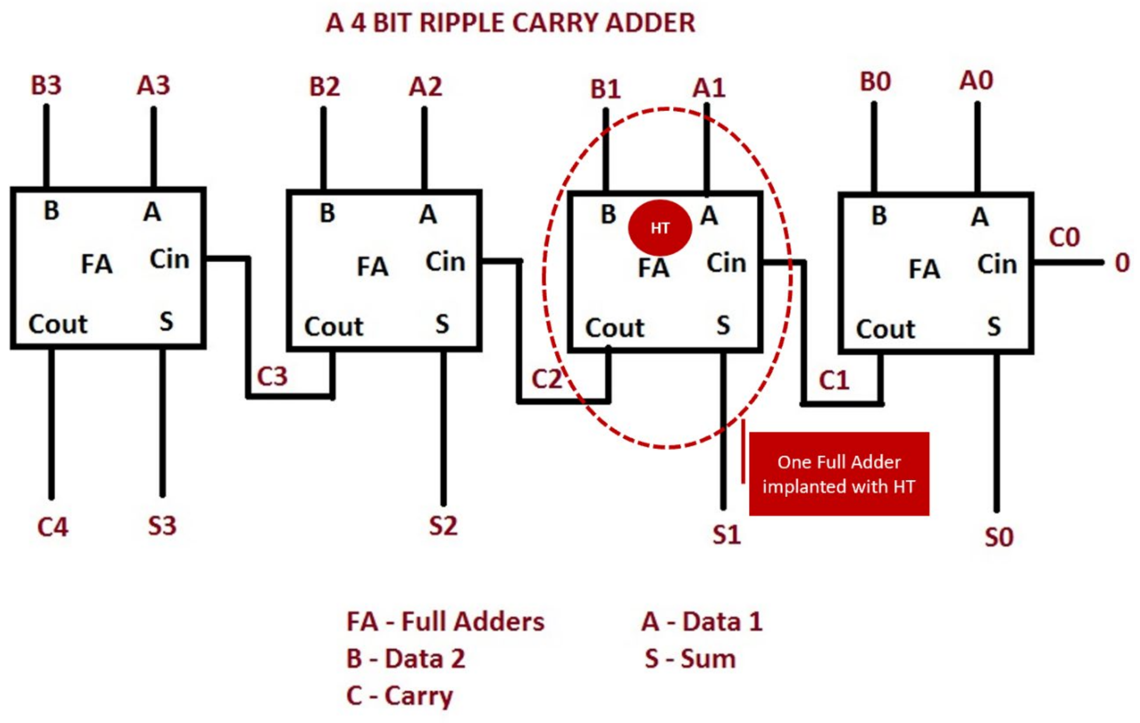


Figure 5-7 A block diagram of a 4-BIT ripple carry adder. Full adder with C1 is implanted with a threshold-voltage shift triggered hardware Trojan.

A block diagram of 4-BIT ripple carry adder implanted with a stealthy malicious transistor in one of its four full adders (FA) is shown in Figure 5-7. A ripple carry adder is basically a logic circuit in which the **carry-out** of each full adder is the **carry-in** of the succeeding next most significant full adder. It is called a ripple carry adder because each carry bit gets rippled into the next stage. The adder consists of four 1-BIT full adders, which are series connected through **C$_{in}$** and **C$_{out}$** outputs.

We have implemented one full adder with two XOR gates for sum output calculation and a multiplexer for obtaining carry output, as shown in the gate-level diagram in Figure 5-8. For Trojan, we have employed pass transistor logic (with one as a malicious transistor – the hardware Trojan) as it helps reduce the number of transistors, in our case to eight, so as to achieve low power consumption and less

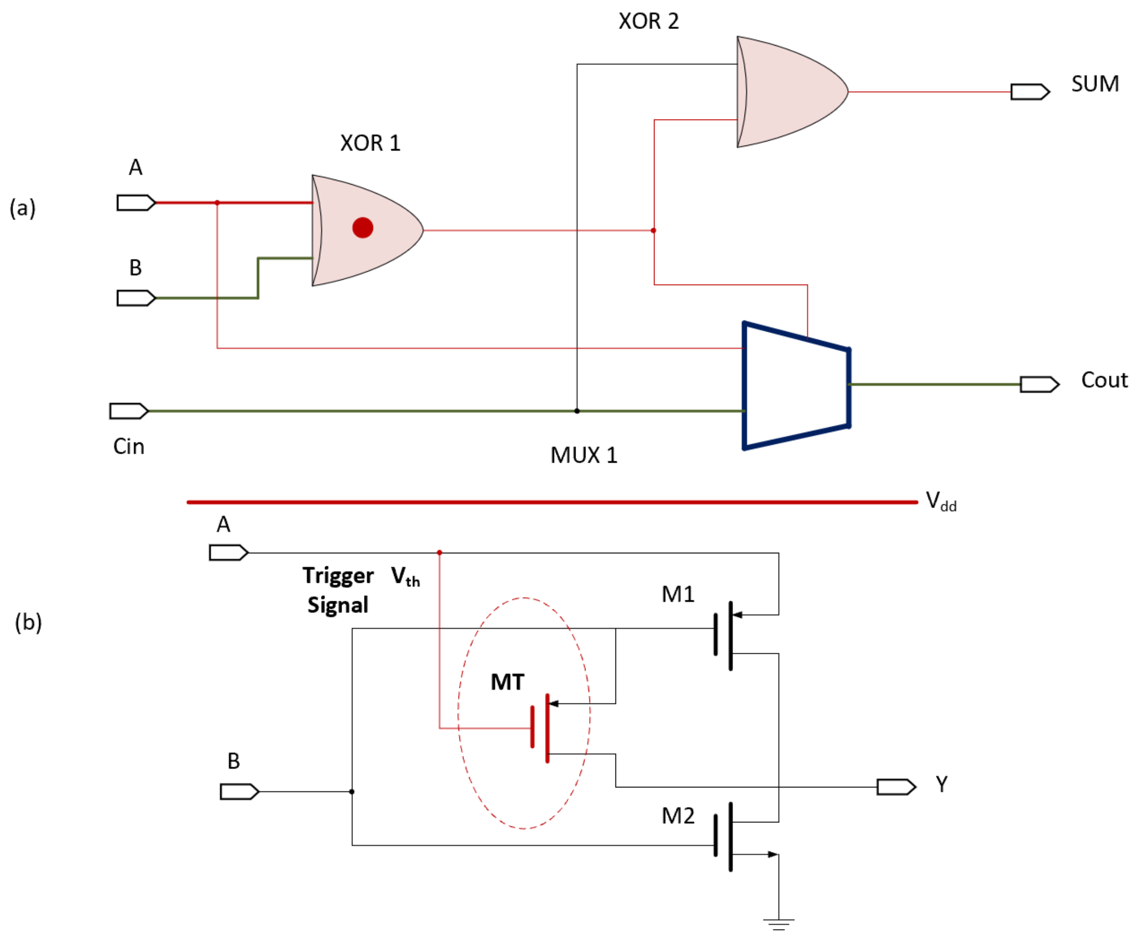


Figure 5-8 (a) Gate level diagram of Full Adder with XOR1 implanted with hardware Trojan, represented by a red dot. (b) Transistor level circuit of XOR gate with malicious transistor (MT) that receives V$_{th}$ trigger at its gate.

area on the chip (as compared to conventional CMOS full adder design that requires 28 transistors [47], [48]), and hence low detectability.

In this case, the first XOR gate is an inverter modified with PMOS pass transistor **MT (occupying an area of 2.45 μm^2),** acting as a Malicious Transistor – the hardware Trojan. Under normal operation, with **high** logic level at **input A** this inverter works as a common CMOS inverter for the **low** logic level at **input B**. When there is **Low** logic level at **A,** the inverter goes to a state with **high impedance**. In such a case, output **Y** gets the same logic value as at input **B** due to open PMOS pass transistor **MT**. However, as the input **A** (connected to MT) gets a triggering threshold voltage shift signal (beyond the pre-defined value of 0.45V) from the 'Threshold-Voltage Meter' (described in section 5.3.3), MT becomes active high and causes sudden voltage degradation at output **Y**. As a result, voltage degradation also occurs at the multiplexer, thereby resulting in the overall failure of the ripple carry adder circuitry.

5.3.2.3 Conceptualising Hardware Trojan in Sequential Circuits

In order to build a **sequential** circuit for hardware Trojan demonstration, we consider adding two flip flops (K and L) to the combinatorial circuit, as shown in Figure 5-9 and Figure 5-10. The binary decoding with two bits X and Y as the most significant bit (MSB) and the least significant bit (LSB), respectively, are used for the flip flops. An inactive hardware Trojan, MT (occupying an area of **7.25 μm^2**), is embedded into the flip flop K ***(overall area of this circuitry raises to 75μm^2, consuming power of 1.25μW***). Under no-triggering and normal operating conditions, the sequential circuit functions optimally without any effect on the dynamic power consumption. As the MT

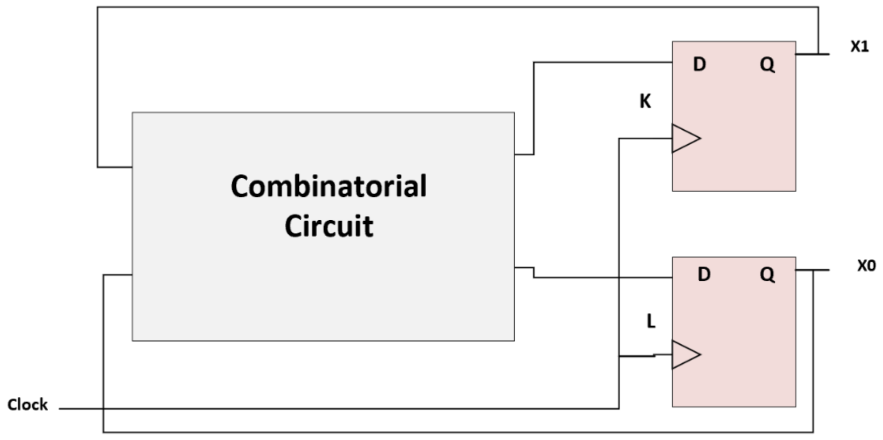


Figure 5-9 Block diagram representation of a sequential circuit.

is triggered, the supply voltage (***V_{DD}***) feeding the flip flop 'K' is cut off, resulting in the malfunction of the flow of finite state machine (FSM). Although the flip flop 'L' remains unaffected and healthy, the failing of flip flop 'K' reduces the FSM states to only two high impedance states - z_0 and z_1.

The above structure is further elaborated by constructing a true single-phase clock (TSPC) based flip flop. The payload is the same PMOS transistor MT with a MOS resistor (MR) connected in series to it, as shown in Figure 5-6. MT, acting as a switch, controls the connection of the body and source of all PMOS transistors (M1, M2, M4, M7, and M10) in the flip flop. The bodies of all NMOS transistors (M3, M5, M6, M8, M9, and M11) are grounded permanently. When the switch MT is 'ON,' all the PMOS transistors remain connected to ***V_{DD}.*** On the contrary, when the switch MT is in 'OFF' state, the body and the source of all PMOS transistors are shorted to ground through the resistor, leaving the flip flop without power supply and resulting in circuit malfunction. Similar to the triggering of MT in the combinatorial circuit, the shift in threshold voltage due to NBTI is designed to initiate MT triggering here in the sequential structure as well. A Global Foundries 28 nm process technology is used to accomplish circuit implementations and subsequent logic applications.

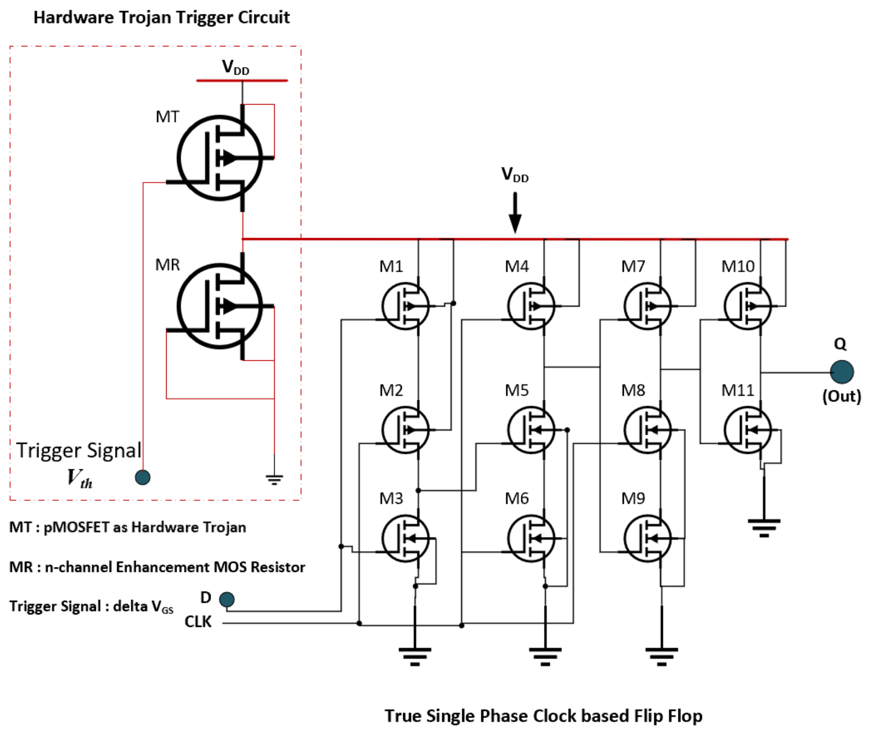
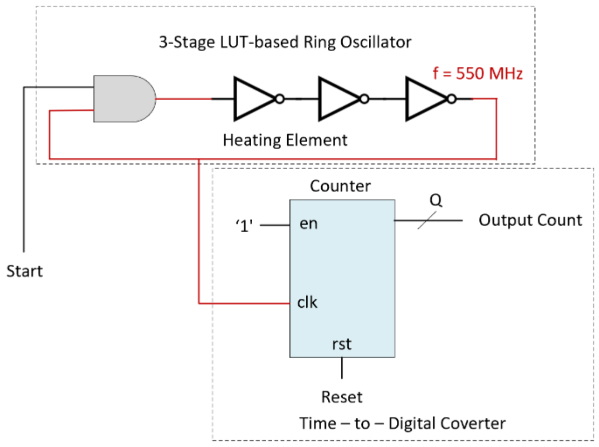
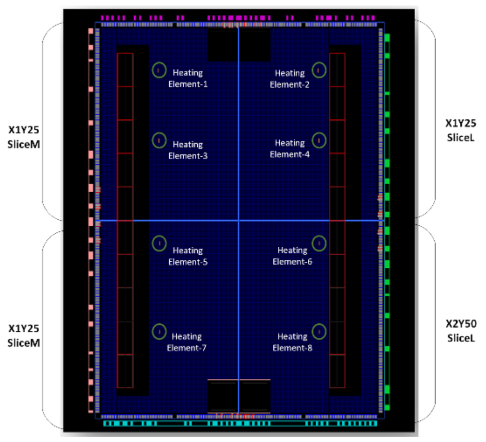

Hardware Trojan Trigger Circuit
VDD
MT
MR
Trigger Signal
Vth
MT : pMOSFET as Hardware Trojan
MR : n-channel Enhancement MOS Resistor
Trigger Signal : delta VGS
D
CLK
VDD
M1
M4
M7
M10
M2
M5
M8
M11
M3
M6
M9
Q
(Out)
True Single Phase Clock based Flip Flop

Figure 5-10 Schematic of threshold voltage-triggered hardware Trojan in a Sequential Circuit (TSPC based Flip Flop).

5.3.2.4 Adding Ring Oscillator Based Heating Element for Accelerated NBTI Impact

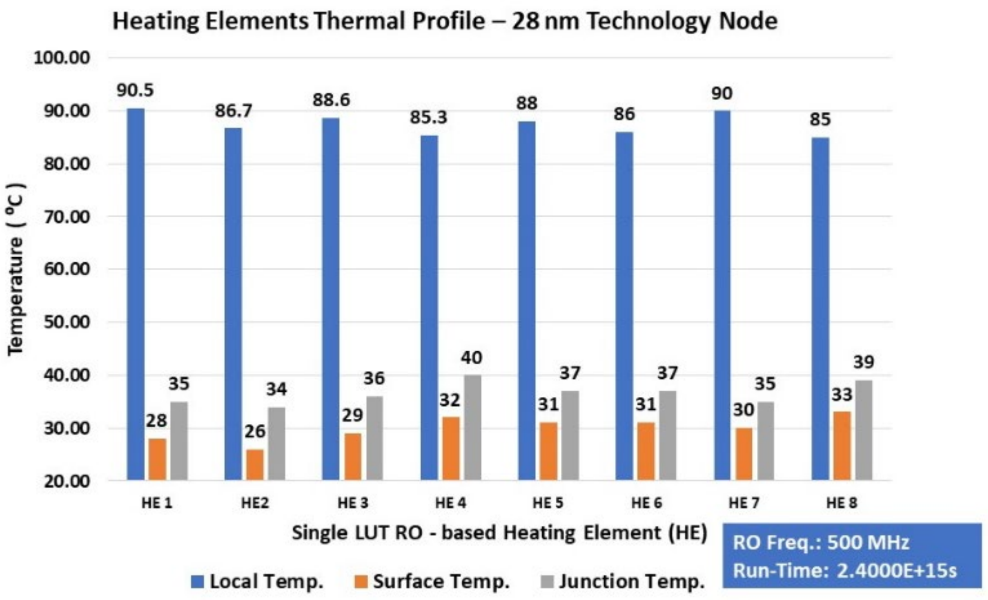
To accelerate the NBTI ageing mechanism and observe a corresponding shift in threshold voltage 'V_{th},' we designed and implemented a LUT-based ring oscillator to act as a heating element for raising the temperature high enough to trigger NBTI. The architecture of the heating element is shown in Figure 5-11(a). It is important to note





(a)

(b)



(c)

Figure 5-11. (a) Schematic of a 3-stage Ring Oscillator-based heating element with Time-to-Digital Converter. (b) 28 nm technology node floor-planned with 08 x heating elements. (c) Thermal profile of FPGA (28 nm technology node) with 08 x heating elements.

here that this heating element is designed and implemented as an integral part of the hardware Trojan infection scheme.

As stated earlier, there exists a strong correlation between the shift in threshold voltage and the die temperature. Taking this into account, a set of eight controllable ring oscillators (ROs), comprising 3-inverter stages and a time-to-digital converter (TDC) each, are implemented across the FPGA fabric (*28 nm technology node*) at locations shown in Figure 5-11(b) using the Vivado design suite. It is noteworthy that the number of stages in a ring oscillator determines the toggling frequency and hence, the corresponding amount of heat generation, measurable as a variation in temperature [49]. In order to disrupt the ISPS system, the toggling frequency of an RO must be high enough to generate a large amount of heat per micron for high temperatures. Accordingly, only a single LUT is used to implement RO with 3-inverter stages and a TDC.

We define the area-constraint for our heating elements to only 8 LUTs (0.00025%) out of the total 32,000 LUTs constituting the CLBs. The built-in system monitor is then programmed to access XADC sensor readings of the thermal diode in FPGA. The heating element is enabled/disabled by a time-driven program running on the FPGA, which also keeps reading the temperature values and transmitting them to the workstation via the JTAG interface.

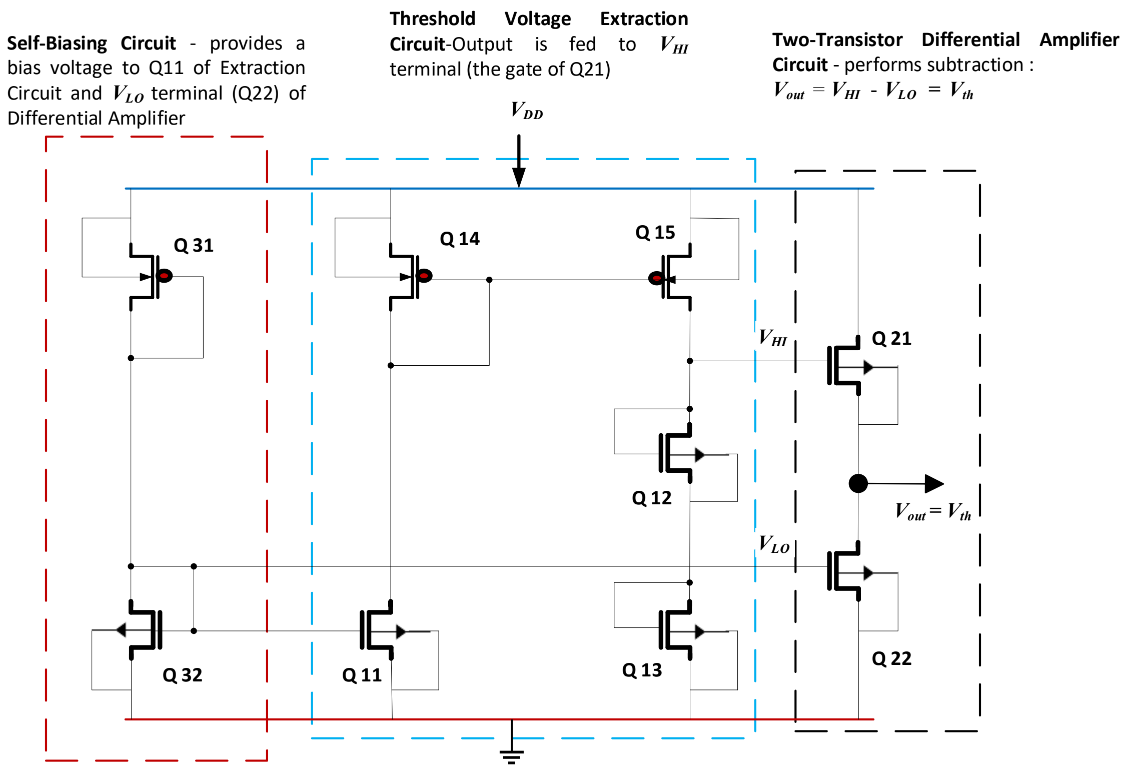
The execution of the experiment is organized in such a way that the die temperature of the FPGA is allowed to stabilise for a period of 35 minutes before enabling the heating element for a period of 40 minutes. Upon completion of this operational phase, the heating element is disabled and allowed to rest for 35 minutes. During this period, the fall in temperature is observed to assess the behaviour of the heating element. Finally, the heating element is again enabled for another 40 minutes to affirm the repeatability and validity of the experiment.

We tested the LUT based ring oscillators (the heating elements) spread over eight different locations on the FPGA as per the procedure mentioned in the previous paragraph and measured it toggling at 550 MHz. The temperature measurements were made using the FPGA's internal thermal diode (*for the whole FPGA*), on-chip thermal sensors (*the LUT based RO connected to the counter for local temperature*), and the external laser-based IR temperature gun, positioned over the FPGA package.

Initially, the temperature is stabilised to an idle FPGA state, meaning when it is powered up and configured, with the negligible workload, and without the heating elements enabled. The idle temperature for the whole die (*junction temperature*) is measured to be **10.5ºC**, the local RO **10ºC**, and the surface **5ºC**. The heating elements are subsequently enabled with clock disabled to achieve asynchronous behaviour of LUT based RO and toggle as fast as physically possible without any clock constraint. Upon enabling the heating elements one by one for a period of 40 minutes each, the local, junction, and surface temperatures depicting the thermal profile of an FPGA is obtained, as shown in Figure 5-11(c). It can be seen that the temperatures rise considerably higher to cause shifts in the threshold voltage and accelerate the NBTI degradation mechanism. The threshold voltage meter, described later, continuously measures the voltage till the time the hardware Trojan circuit is triggered at a value above the nominal 'V_{th}' value (0.45V).

5.3.3 Threshold Voltage Meter

As mentioned earlier, the shift in threshold voltage 'V_{th}' is the manifestation of the ageing mechanism of NBTI in PMOS transistors that make up the FPGA fabric and its

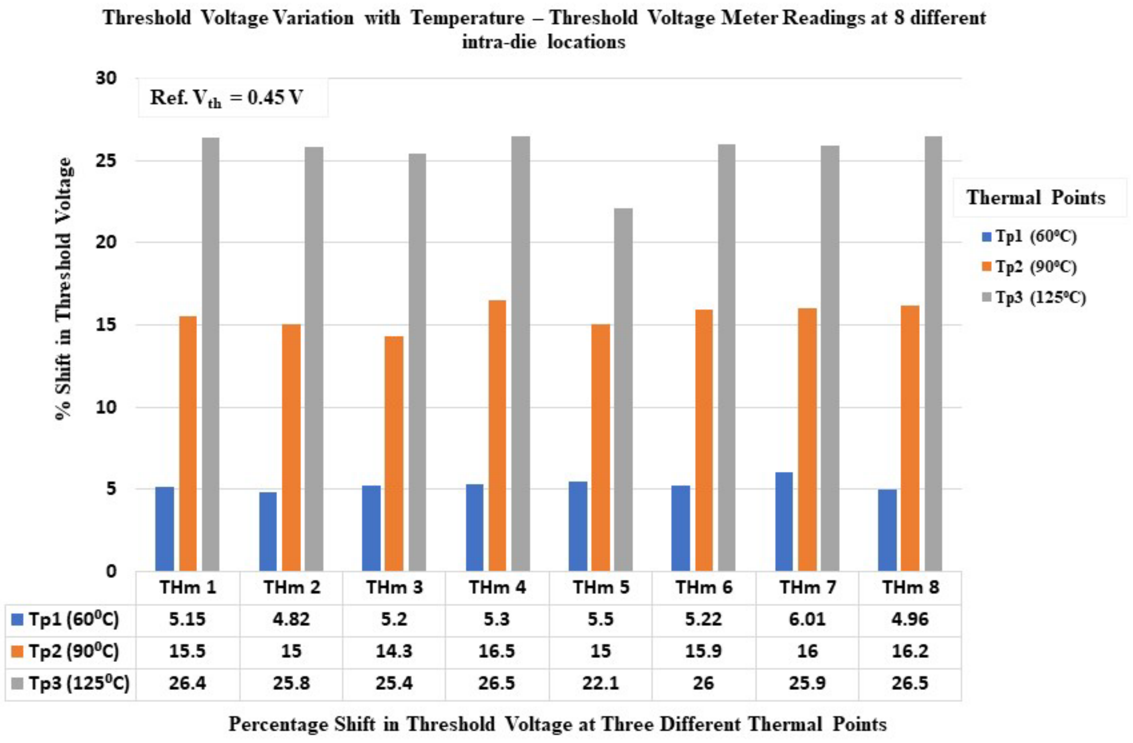


Self-Biasing Circuit - provides a bias voltage to Q11 of Extraction Circuit and V_{LO} terminal (Q22) of Differential Amplifier
Threshold Voltage Extraction Circuit-Output is fed to V_{HI} terminal (the gate of Q21)
Two-Transistor Differential Amplifier Circuit - performs subtraction : $V_{out} = V_{HI} - V_{LO} = V_{th}$
V_{DD}
Q 31
Q 14
Q 15
V_{HI}
Q 21
Q 12
$V_{out} = V_{th}$
V_{LO}
Q 32
Q 11
Q 13
Q 22

Figure 5-12. Schematic of Threshold Voltage Meter. The output of the Differential Amplifier is the Threshold Voltage (V_{th}).

primitives. Therefore, the precise measurement of '***V_{th}***' is critical for triggering the threshold voltage based hardware Trojan. Accordingly, we design and implement a threshold voltage meter that directly generates an output voltage '***V_{out}***,' equal to '***V_{th}***.' Figure 5-12 shows the schematic diagram of the meter. As is evident, this circuit has no reference voltage '***V_{ref}***' input and is, therefore, a 3-terminal circuit. The transistors Q31 and Q32 provide a bias voltage at the gate of Q11; this voltage is then applied to the low voltage '***V_{LO}***' terminal of the differential amplifier, i.e., at the gate of Q22. Whereas, the transistors Q11-Q15 implement a circuit whose output is applied to the high voltage '***V_{HI}***' terminal of the differential amplifier at the gate of Q21. Eventually, the Differential amplifier comprising Q21 and Q22 performs the subtraction process outputs '***V_{th}***' at the drain of Q22, as shown in Figure 5-12.

In order to validate the operation-ability, functionality, and accuracy of the designed hardware Trojan, an experiment consisting of all elements of HT infection scheme (***RO based heating elements, threshold voltage meter, and the trojan circuit)*** is performed. It ascertains whether a triggering signal, ***a shift (increment) in pre-defined threshold voltage***, can be latched or not. Furthermore, in case of being



	THm 1	THm 2	THm 3	THm 4	THm 5	THm 6	THm 7	THm 8
🟦 Tp1 (60°C)	5.15	4.82	5.2	5.3	5.5	5.22	6.01	4.96
🟧 Tp2 (90°C)	15.5	15	14.3	16.5	15	15.9	16	16.2
⬜ Tp3 (125°C)	26.4	25.8	25.4	26.5	22.1	26	25.9	26.5

Figure 5-13. % Shift in threshold voltage with rise in temperature across 8 different intra-die locations. Threshold voltage meter is used to read V_{th}. Reference V_{th} is pre-defined at 0.45V.

latched, ascertain whether the payload (***accelerated ageing***) of the hardware Trojan gets activated. A controlled temperature environment is ensured using a thermal chamber with an HT infection scheme-implemented FPGA (*28 nm technology node*) placed inside it. The external temperature (*i.e., thermal chamber temperature*) is maintained between 5-10ºC (***a typical warship computer control room temperature***). The JTAG interface is used for programming and bidirectional communication between the FPGA and the workstation. Digital oscilloscope, Vivado power analyser, FPGA system monitor, and integrated logic analyser (ILA) are employed to capture the threshold voltage, drain current, and thermal points.

The first stage is the initialization of FPGA under test. This involves the stabilization of the thermal chamber at 5ºC, powering up of the target FPGA, and providing an operating voltage of 1.0V. Once powered up, the LUT based ring oscillators implemented to produce heat are enabled. This leads to the second stage where the heat (rise in temperature and a corresponding shift in threshold voltage) generated by the heating elements, spread across the device at locations shown in Figure 5-11(b) is continuously measured and logged using the local as well as the system monitor. The temporal change in temperature observed is shown in Figure 5-13. As the temperature traverses the primary thermal point of ***'T_{p1}'*** (***60ºC***), the changes in threshold voltage ***'V_{th}'*** and ***'I_{dd}'*** are extracted and measured by Threshold Voltage

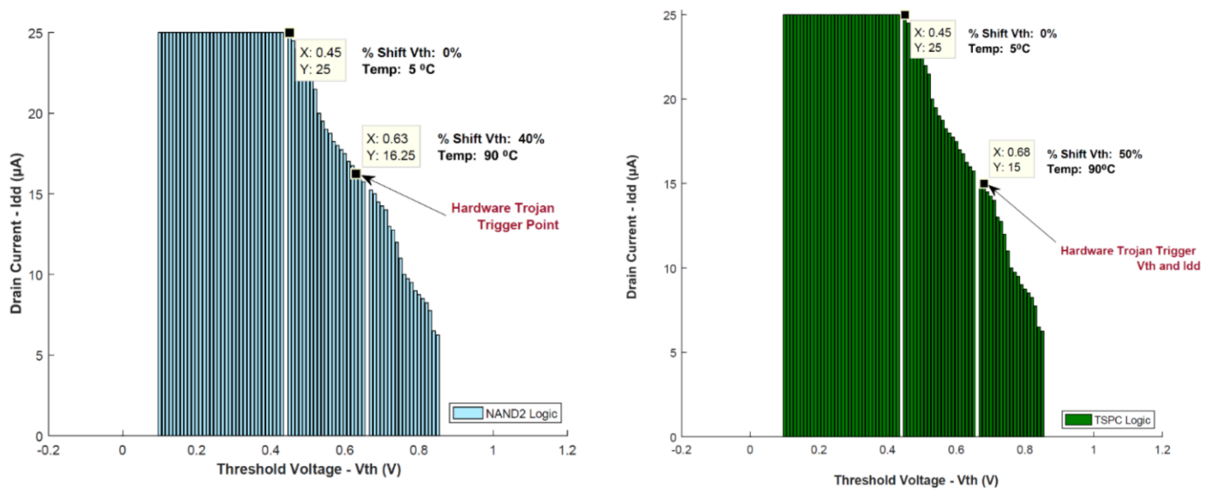
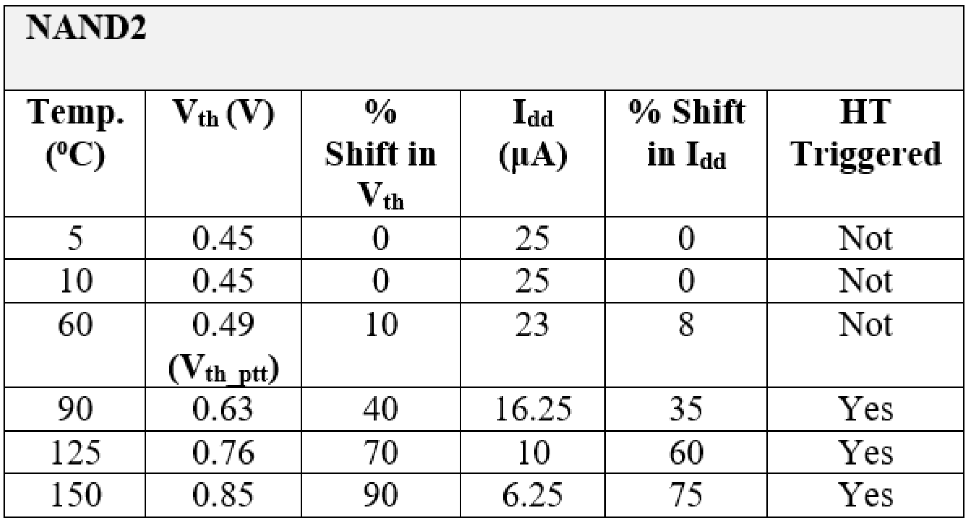


Figure 5-14. (Left) An increase of 40% shift in threshold voltage at 90ºC degrades the drain current by 35%, triggers the hardware Trojan and impairs the NAND2 logic. (Right) An increase of 50% shift in threshold voltage at 90ºC degrades the drain current by 40%, triggers the hardware Trojan and impairs the TSPC logic.

Table 5-2 Hardware Trojan Triggering Analysis in NAND2 Logic.



NAND2					
Temp. (^oC)	**V_{th} (V)**	**% Shift in V_{th}**	**I_{dd} (µA)**	**% Shift in I_{dd}**	**HT Triggered**
5	0.45	0	25	0	Not
10	0.45	0	25	0	Not
60	0.49 (**V_{th_ptt}**)	10	23	8	Not
90	0.63	40	16.25	35	Yes
125	0.76	70	10	60	Yes
150	0.85	90	6.25	75	Yes

meter. Similarly, the changes are continually observed, and measurements are taken at secondary and tertiary thermal points (***T_{p2}* -90ºC** and ***T_{p3}* -125ºC** respectively). We took 10K samples for each thermal point at all the eight different locations within FPGA. A complete mesh of plot showing the shifts in threshold voltage with change in temperature is given in Figure 5-13. In the third stage, these readings are critically analysed for false positives and accuracy for temperature variation and corresponding shifts in threshold voltage as well as ***'I_{dd}'*** to observe the presence of any process variations. Accordingly, three additional runs are undertaken to take further readings and observe intra-run deviations to establish measurement accuracy. During all these three stages, the hardware Trojan trigger circuit remains silent connected with the NAND gate and

TSPC PMOS transistors till the time the hardware Trojan trigger circuit experiences a shift in threshold voltage from ***0.45V to 0.63V*** (***40%***) in NAND2 and ***0.67V*** (***50%***) in TSPC logic. Consequently, the trigger circuit of hardware Trojan causes corresponding significant ***I_{dd}*** degradation, as can be seen in Figure 5-14(left) and Figure 5-14(Right) respectively. This, eventually cuts off the ***V_{DD}*** connection of the PMOS transistors, which constitute the NAND gate and TSPC. As a result, the whole logic is deactivated, thereby crippling its critical function. The quantitative representation of the percentage shift in threshold voltage (*an increase in this case*) of MOSFETs that triggers the stealthy hardware Trojan is given in Tables 5-2 and 5-3.

Before approaching a trigger percentage shift in ***V_{th}***, a gradual increase in signal delays is also observable, for instance, with a ***50%*** shift in the threshold voltage and corresponding ***40%*** shift in ***I_{dd}*, **the increase in the rise and fall times from 20.5 ps and 26.7 ps respectively to 22.9 ps and 28.0 ps is recorded. TSPC and NAND circuits remain stable with no triggering of hardware Trojan. However, the slowing down of switching control is observable. As the threshold voltage shift hits ***50%*** of the nominal threshold value of ***0.45V***, the hardware Trojan gets activated. The same is observed for ***70%*** to ***100%*** shifts in the nominal threshold voltage. This experimental result is in

Table 5-3 Hardware Trojan Triggering Analysis in True Single Phase Clock (TSPC) Logic.

True Single Phase Clock (TSPC) Logic					
Temp. (^oC)	**V_{th} (V)**	**% Shift in V_{th}**	**I_{dd} (μA)**	**% Shift in I_{dd}**	**HT Triggered**
5	0.45	0	25	0	Not
10	0.45	0	25	0	Not
60	0.54 (V_{th_ptt})	20	22	10	Not
90	0.68	50	15	40	Yes
125	0.81	80	8.75	65	Yes
150	0.90	100	3.75	85	Yes

consonance with the Monte Carlo simulation carried out by sweeping parameter values using Gaussian distribution. For the simulation purposes, the mean value is set to the nominal threshold voltage value (***0.45V***), whereas the standard deviation **(±σ)** is kept at **±0.1V** of the mean value.

5.4 Design and Implementation of a Threshold Voltage-Aware Sensor

The requirement of a lightweight and highly sensitive sensor for the detection of shifts in threshold voltage much earlier than the triggering of hardware Trojan is a critical design consideration. This is to ensure that the hardware Trojan never gets triggered, provided its presence in FPGA has been accurately assessed. We draw the attention of readers to the vital nature of a naval warship defence capability that should not get compromised due to faltering EW-ISPS system dependent on system processor, housing an FPGA. Therefore, the design and implementation of a highly sensitive sensor that detects minor shifts in threshold voltage due to the NBTI effect captures the corresponding frequency shifts and signal path delays and monitors the resultant ageing of the device to provide high confidence in ISPS system performance is paramount. This forms the whole concept of the HT-detection scheme, which is designed and implemented at the recommended placements of security assurance and defence teams, **D** (Figure 5-5).

5.4.1 Threshold Voltage Based Sensor Architecture

In continuation to the next stage of the threat model and keeping in perspective the techniques mentioned in [50] and [51], we propose a lightweight sensor that consists of two segments of ring oscillators (ROs), namely the ***'Fixed Sensor Segment (FSS)'*** and the ***'Dynamic Sensor Segment (DSS)'*** as shown in Figure 5-15. The fixed sensor segment is designed to experience shifts in threshold voltage at a slower rate as compared to the dynamic sensor segment, which is made to undergo thermal stresses put through the hardware Trojan infection scheme. This must lower the oscillation frequency of the dynamic sensor segment while the fixed sensor segment exhibits a negligible change in its oscillation frequency. With the increasing disparity between the oscillation frequencies of these two segments, the signs of FPGA ageing and hence signal path delays provide a precursor to the inserted hardware Trojan triggering and payload activity.

It is pertinent to mention that the accuracy of a sensor is susceptible to large process variations (PVs) that exist in lower technology nodes. When process variations outpace shifts in oscillation frequency and threshold voltages, it becomes challenging to differentiate the impact of NBTI from that of the global and local process variations

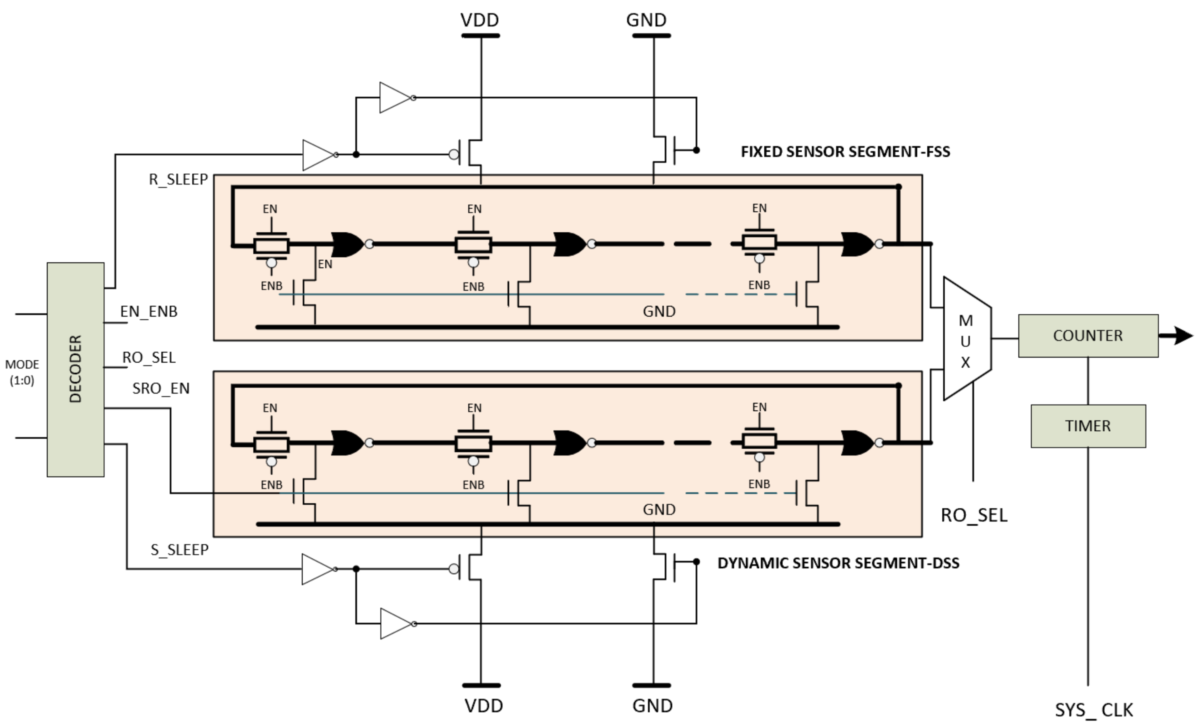


Figure 5-15. The architecture of Threshold Voltage-Aware Sensor.

(and this impacts the accuracy of detection and parametric measurements). We overcome this by placing the two segments of ROs very close to each other to zeroise PV and any environmental variation other than the one generated by the hardware Trojan insertion scheme (i.e., the rise in temperature).

The detailed architecture of the proposed sensor is shown in Figure 5-15. As can be seen, the dynamic sensor segment is sensitized by introducing a pass transistor between inverters and pulling down the inputs of all inverters to the ground through a network of nMOS transistors. In order to keep all the electrical parameters like node capacitance, resistance, etc. closely matched to the dynamic sensor segment, the same structure is maintained within the fixed sensor segment. Such an arrangement helps ensure that at the time ***'t_0'***, when there is no shift in threshold voltage, the difference of oscillation frequency between the two segments is minimal. The only impact observable could be the small variations present between the ROs of the two segments.

In order to implement a specific mode of operation, a decoder circuit is inserted before the two sensor segments to generate the corresponding internal signals, as shown in Table 5-4. For instance, when enable EN is set to '0', the RO segments start oscillating while the pass transistors stay 'ON.' A timer-controlled counter is placed at the segments' output to enable an instant measurement of their respective cycle counts.

Table 5-4 Binary Modes of Operation.

Binary Mode	Signals					Explanation
	R_SLEEP	EN	RO_SEL	SRO_EN	S_SLEEP	
0:0	0	X	X	X	0	RO segments are in dormant phase as their connection to the power and ground line is cut off.
0:1	0	0	X	1	1	Fixed sensor segment (FSS) remains dormant whereas the dynamic sensor segment (DSS) assumes the threshold voltage-aware mode
1:0	1	1	0	0	1	Detection and measurement mode activated. Oscillation frequencies/cycle counts of Fixed Sensor Segment – FSS (RO segments) are measured.
1:1	0	1	1	0	1	Detection and measurement mode activated. Oscillation frequencies/cycle counts of Dynamic Sensor Segment - DSS (RO segments) are measured.

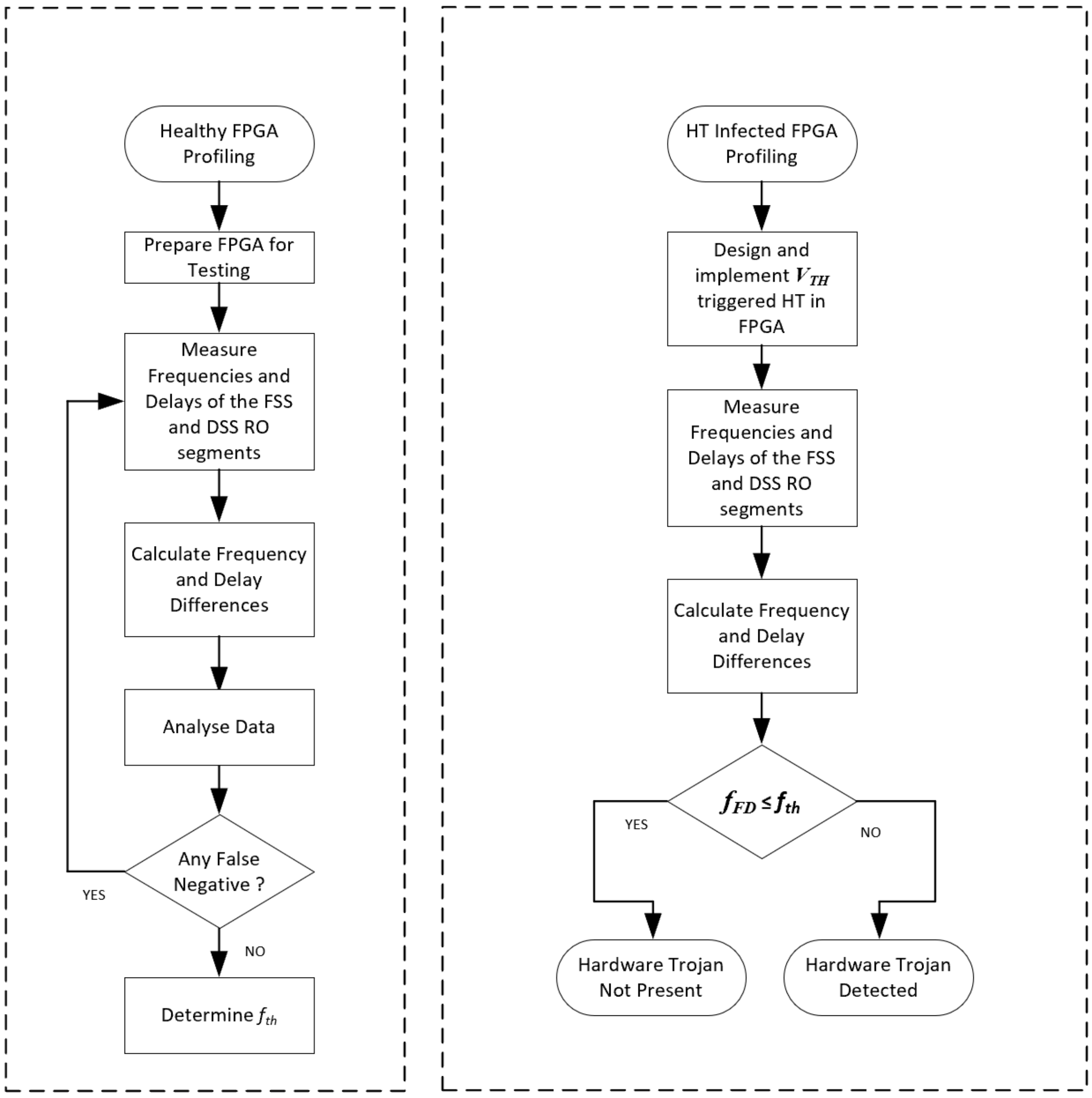

Healthy FPGA Profiling
Prepare FPGA for Testing
Measure Frequencies and Delays of the FSS and DSS RO segments
Calculate Frequency and Delay Differences
Analyse Data
Any False Negative ?
YES
NO
Determine f_{th}
HT Infected FPGA Profiling
Design and implement V_{TH} triggered HT in FPGA
Measure Frequencies and Delays of the FSS and DSS RO segments
Calculate Frequency and Delay Differences
$f_{FD} \leq f_{th}$
YES
NO
Hardware Trojan Not Present
Hardware Trojan Detected

Figure 5-16 Process flows for the identification, authentication, and assessment of Trojan-free and Trojan-infected FPGAs using frequency and delay mapping method.

For our design of the sensor, four distinct modes of operation, as explained in Table 5-4, are considered. At mode 1 (**0:0**), both the segments are inactive or in the dormant phase as their connection to the power and ground line is cut off. This mode is valid for the duration, the heating elements are silent, i.e., during the stabilization phase of the thermal chamber. As the heating element is enabled, and it approaches the primary thermal point ($\boldsymbol{T_{p1}}$ **- 60°C**), operation mode 2 (**0:1**) is enforced. In this mode, the fixed sensor segment (FSS) remains dormant (0), whereas the dynamic sensor segment (DSS) assumes the threshold voltage-aware mode (1). Every inverter in DSS is now subjected to dc stress (induced by gradual shifts in threshold voltage) by pulling its input to the ground. This causes changes in its oscillation frequency/cycle count and induces signal delays. When the secondary thermal point ***T*p2 -90ºC** is reached, the operation modes 3 (**1:0**) and 4 (**1:1**) are activated, and oscillation frequencies/cycle counts of both RO segments are measured. This process of measurement continues

until the FPGA junction temperature reaches the tertiary thermal point ***T*p3 -125ºC**. It must be noted here that these measurements are aimed at (1) testing and validating the threshold voltage-aware sensor's efficiency in terms of power and area consumption, (2) determining the frequency threshold of a hardware Trojan-free FPGA at varying locations, and (3) the impact of process variations (PVs) on sensor's accuracy.

5.4.2 Determining Threshold Frequency for Correlation and Authentication

In order to develop a trustworthy threshold voltage triggered hardware Trojan detection scheme, we have defined Trojan-free and Trojan-infected process flows to establish the presence of hardware Trojan in an FPGA. Figure 5-16 shows the two processes. The main purpose behind the Trojan-free frequency mapping is to determine the threshold frequency ***'f_{th}'*** corresponding to pre-Trojan trigger threshold voltage ***'V_{th_ptt}'*** and provide a reference to compare the frequency differences of FSS and DSS **'*f_{FD}*'** with it. If '***f_{FD}***' is greater than ***'f_{th},'*** we consider this as an indication of ***'HT_{Vth}'(threshold voltage-triggered hardware Trojan)*** presence and a precursor to its triggering and

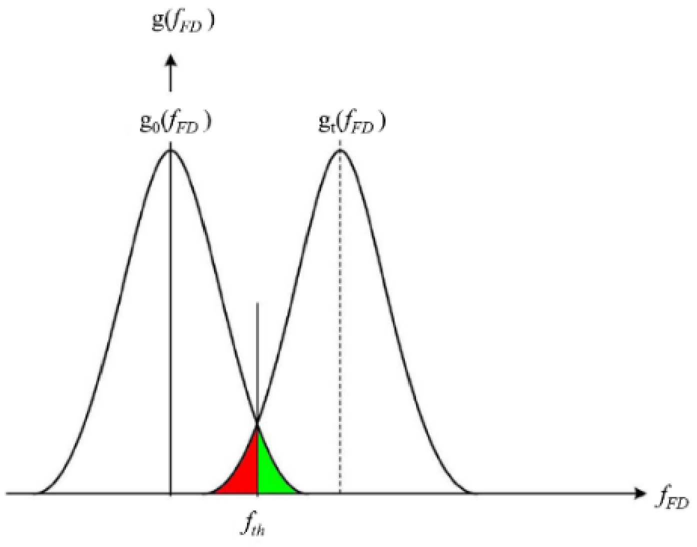


Figure 5-17. Probability density function *f_{FD}* at times 0 g_0(*f_{FD}*) and t g_t(*f_{FD}*).

payload effect. During the Trojan-free frequency mapping phase, a 28 nm FPGA is used to generate the requisite distributions to determine the threshold frequency ***'f_{th}.'*** The Trojan-free phase implies that the Trojan circuit is already inserted and present in the FPGA but lying in a dormant state.

Although the two RO segments are placed very close to each other to zeroise the difference of oscillations ***'f_{FD}'*** between them, yet due to process variations, it will not be zero. Also, a Gaussian distribution of ***'f_{FD}'*** is observed during the tests. A simplified representation of the two distributions as probability density functions of ***'f_{FD}'*** at times ***'0'*** $g_0(f_{FD})$ and ***'t'*** $g_t(f_{FD})$ is shown in Figure 5-17. The frequency differences between the two RO segments ***'f_{FD}'*** are represented by the x-axis, whereas the y-axis represents the relative distribution function. The overlapping area gives the false prediction of the presence of hardware Trojan or vice versa. The red area '**θ_a**' represents the probability of detecting Trojan-infected FPGA as ***'HT-free,'*** whereas the green area **θ_b** denotes the probability of identifying the Trojan-free FPGA as ***'HT-infected.'*** Mathematically,

$$\theta_a = \int_{-\infty}^{f_{th}} g_t(f_{FD})\, d\, f_{th} \qquad (5-1)$$

$$\theta_b = \int_{f_{th}}^{\infty} g_0(f_{FD})\, d\, f_{th} \qquad (5-2)$$

Where, **$g_0(f_{FD})$** and **$g_t(f_{FD})$** correspond to the distribution of frequency differences for Trojan-free (dormant) and Trojan-infected FPGAs, respectively. The threshold frequency ***'f_{th}'*** is considered to be a point where both distributions intersect one another, hence representing the frequency difference that reduces the total probability of error **($\theta_a + \theta_b$)**.

5.4.3 Reducing the Rate of False Prediction

When the application risk is as critical as in our ISPS case, it is not prudent to let the false prediction, as identified earlier, result in the system failure by failing the proposed sensor to detect hardware Trojan. The repercussions of such a failure may include the collapse of a defence system of the warship and fatal impact on human and material assets. We have, therefore, devised a process of minimizing (***zeroising***) the level of false prediction of the presence of hardware Trojan and vice versa, as shown in Figure

5-18(a)-(c). We observe that false prediction is generated due to the overlap of FSS

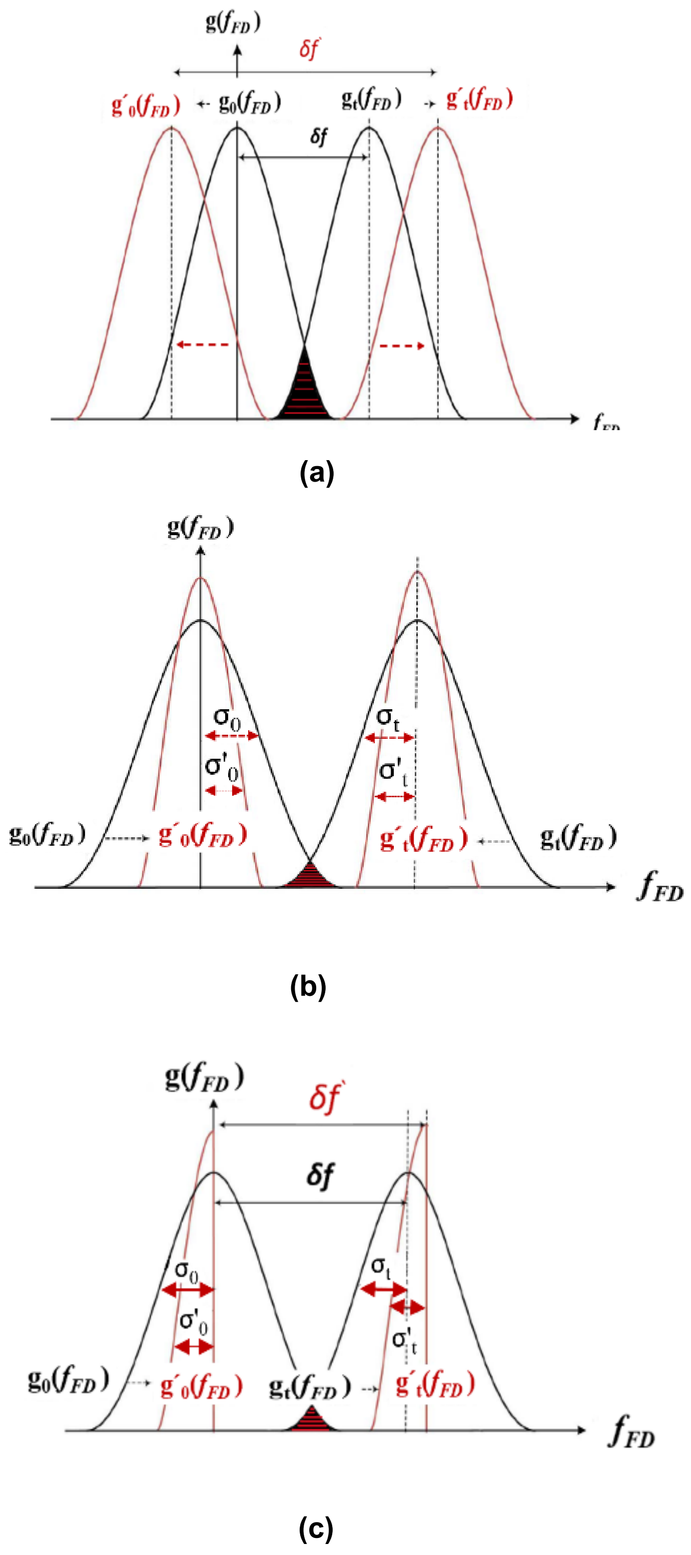


Figure 5-18. Reduction of false prediction - represented by the overlapped area. (a) Moving the FSS and DSS distributions away from their respective positions.(b) Minimizing their spread. (c) Minimal spread with a shift of the mean of FSS and DSS distributions.

and DSS ROs' frequency difference distribution at time ***'0'*** $g_0(f_{FD})$ and at time ***'t'*** $g_t(f_{FD})$, which, in this case, is the 'delay' replica of $g_0(f_{FD})$. It implies that if this overlapping region is reduced, the critical issue of false prediction can be resolved.

Accordingly, as a first step, we increase the separation of these distributions, which represents the delay degradation ***'δf'***, by shifting the distribution $g_0(f_{FD})$ to the left $g'_0(f_{FD})$ or alternatively shifting the distribution $g_t(f_{FD})$ to the right $g'_t(f_{FD})$ or by implementing both simultaneously as shown in the Figure 5-18(a). We observed an improved detection of shifts in frequency corresponding to gradual shifts in the threshold voltage as the distribution $g_t(f_{FD})$ is shifted to the right. Secondly, we consider reducing the spread of FSS and DSS frequency difference distributions. The spread is observed due to the variances of distributions **(σ_0^2 and σ_t^2).** As can be seen in Figure 5-18(b), there is no overlap between $g'_0(f_{FD})$ and $g'_t(f_{FD})$, where $\sigma'_0 < \sigma_0$ and $\sigma'_t < \sigma_t$. This arrangement also helps to minimise the false prediction rate. Thirdly, we reduce the spread and increase the separation of these two distributions simultaneously, as depicted in Figure 5-18(c), instead of managing them individually. In such a case, we discard the right-hand side and reduce the spread of $g_0(f_{FD})$ on the left-hand side. It helps reduce the overall spread. The separation, on the other hand,

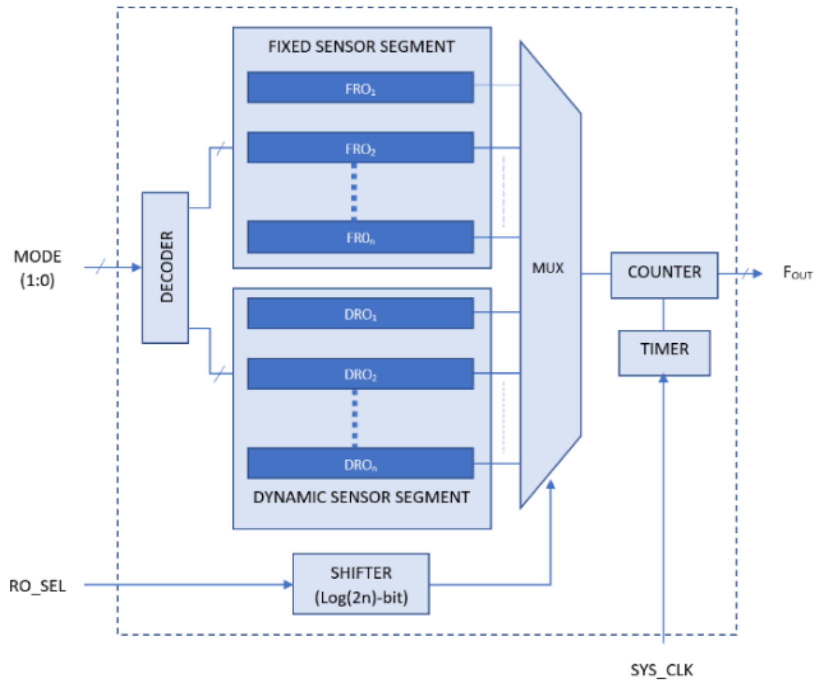


Figure 5-19. Threshold Voltage-aware sensor with enhanced detectability of hardware Trojan due to additional RO pairs architecture.

is simultaneously increased by shifting ***$g_t(f_{FD})$*** to the right-hand side. This technique provides the best detection of frequency degradation and hence, the delay - a pointer towards hardware Trojan activity and corresponding ageing of an FPGA under test. For a detailed account of determining maximum frequency degradation through the application of 'Averaging and Selection' methods, please refer to **Appendices A** and **B**.

5.4.4 Re-architecting the Sensor with Additional Ring Oscillator Segments

Based on the mathematical mean and variance derivations for FSS and DSS segments with additional RO pairs ***(explained in detail at Appendix A)***, we re-architectured the sensor, as shown in Figure 5-19. It consists of the same segments but with two additional threshold voltage shift-aware RO pairs in both. The decision to implement an additional number of RO pairs is primarily aimed at enhancing detectability of abnormal frequency degradation in the shortest amount of time with a negligible false prediction. The results of our experiment show that by the addition of two more RO pairs in both the segments, the detectability of hardware Trojan based on shifts in threshold voltage is unerring.

Looking further at the architecture of the proposed sensor in Figure 5-19, it can be seen that the outputs of all the three RO pairs in both the segments are fed to a multiplexer. A shift register of **$\log_2(2n)$** bit facilitates the Mux. input selection and helps minimise the I/O pin count for the sensor. This register is activated using a 'serial-in RO_SEL' pin. The Decoder, as mentioned earlier, is designed to generate all the internal inputs/signals for the FSS and DSS RO based segments. It is noteworthy that all the RO pairs in each segment utilize the same internal signals generated by the Decoder, and it is not essential to generate the control signals for each RO pair. The operation of the Counter and Timer is the same as elaborated in Section 5.4.1 of this chapter.

In order to achieve high detection and measurement accuracy, we, besides adopting the averaging strategy, also consider the selection strategy as depicted in the process flow in Figure 5-20. The selection strategy implies finding a DSS RO that experiences maximum frequency degradation/delay and hence the ageing due to the NBTI mechanism. For this purpose, the DSS RO pair is compared with the FSS RO pairs

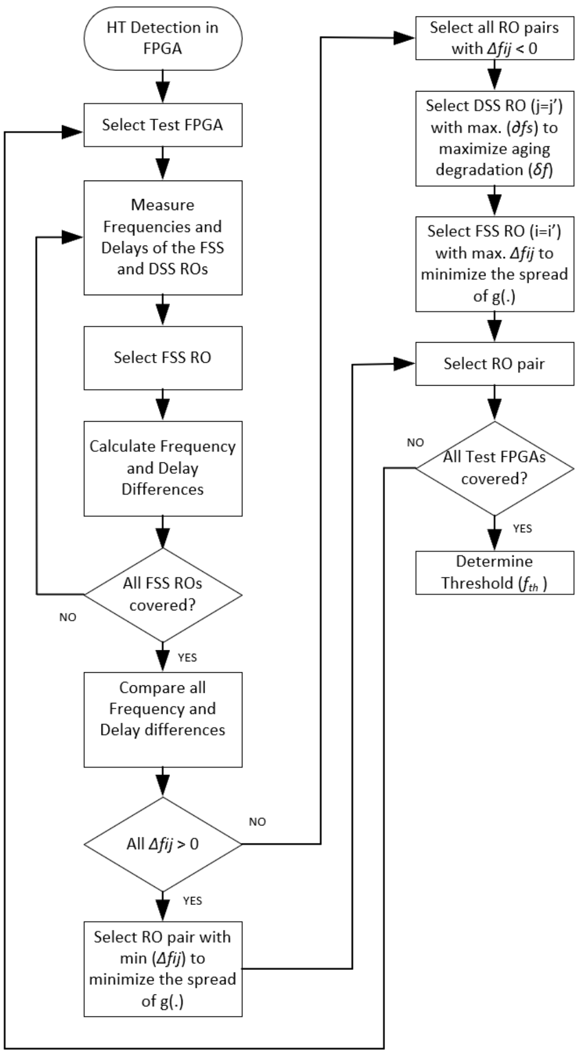

HT Detection in FPGA
Select Test FPGA
Measure Frequencies and Delays of the FSS and DSS ROs
Select FSS RO
Calculate Frequency and Delay Differences
All FSS ROs covered?
NO
YES
Compare all Frequency and Delay differences
All Δfij > 0
NO
YES
Select RO pair with min (Δfij) to minimize the spread of g(.)
Select all RO pairs with Δfij < 0
Select DSS RO (j=j') with max. (∂fs) to maximize aging degradation (δf)
Select FSS RO (i=i') with max. Δfij to minimize the spread of g(.)
Select RO pair
NO
All Test FPGAs covered?
YES
Determine Threshold (fth)

Figure 5-20. Process flow for enhanced detectability of hardware Trojan using optimum-performing RO pairs' selection strategy.

even though they remain dormant during normal operations. It is, therefore, essential to find an FSS RO pair that is slower than the DSS RO pairs during the time ***'0'*** to design a higher sensitivity sensor that enables the detection of hardware Trojan activity well before its onset.

5.4.5 Sensor and Hardware Trojan Detection Scheme – Testing and Analysis

The correct verification of the effectiveness and sensitivity of threshold voltage based sensor for a hardware Trojan detection scheme is, therefore, critical. Consequent to the optimisation of sensor accuracy described in the above section, we implemented the improved sensor design (*with additional RO segments*) in a 28 nm FPGA technology node. The experiment was set up to provide and emulate the ISPS system

Table 5-5 Intra-die process variations–Transistor length and oxide thickness.

Intra-die Process Variations	Parameter	
	Transistor Length (L) %	Oxide Thickness (T_{ox}) %
PV_a	1.5	0.75
PV_b	2.5	1.5
PV_c	8	3.75

environment onboard a naval vessel for realistic side-channel measurements. A nominal supply voltage of 1.0V is provided from a benchtop power supply having basic voltage setting accuracy and voltage readback accuracy of 0.03%. With the enabling of heating elements (*following the same phase -1 process with Negative bias '-1.2V' and Tp '60ºC',* as *described in Section 5.4.1)*, the first set of readings (*including threshold voltage, oscillation frequency/count, and corresponding signal delays*) is taken at stabilised negative bias and primary thermal point, using DL850E ScopeCorder with sample rates up to 100 MS/s.

Similarly, the experiments were conducted for the second and third phases of the scheme. Although the impact of PVs is minimal as the two sensor segments are placed very close to each other, we did, however, consider the impact of process variations on the detection sensitivity of the sensor in terms of percentage, as given in Table 5-5.

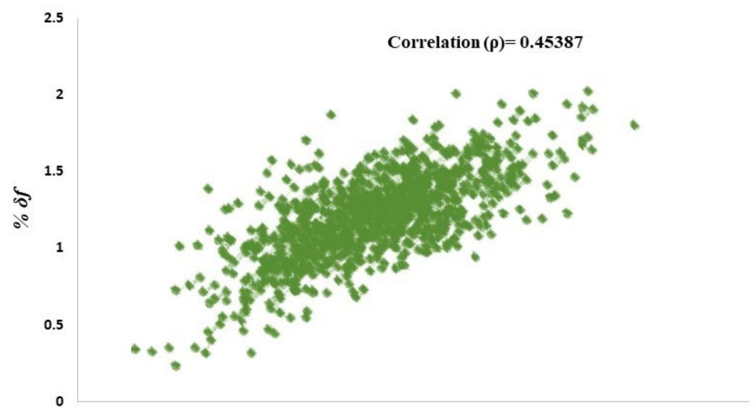


Figure 5-21. Scatter plot of correlation between dynamic frequency degradation (% δf) and percentage frequency difference (% $\partial f_{t\,DSS}$) of DSS ROs (Refer to Appendix B).

These tests were repeated to establish the consistency of results and assure the robustness of the developed scheme. The synopsis of test results is given in Figure 5-21 and Figure 5-22 (a) – (f). The frequency difference of FSS and DSS '$\mathbf{f_{FD}}$' is

represented along the x-axis, and the y-axis represents the frequency of occurrence/the number of test samples. Three different threshold voltage shift states '$\mathbf{V_{th1}}$, $\mathbf{V_{th2}}$, and $\mathbf{V_{th3}}$' corresponding to '$\mathbf{f_{FD}}$' are representative of $\mathbf{V_{th}}$ distribution.

The green ($\mathbf{V_{th1}}$=0%) distribution plot for '$\mathbf{f_{FD}}$' is centred at 0 Hz. Whereas, the distributions in pink and blue corresponding to $\mathbf{V_{th2}}$=40% and $\mathbf{V_{th3}}$=70% respectively shift to the right. This is because the oscillation frequency/count of DSS slows down

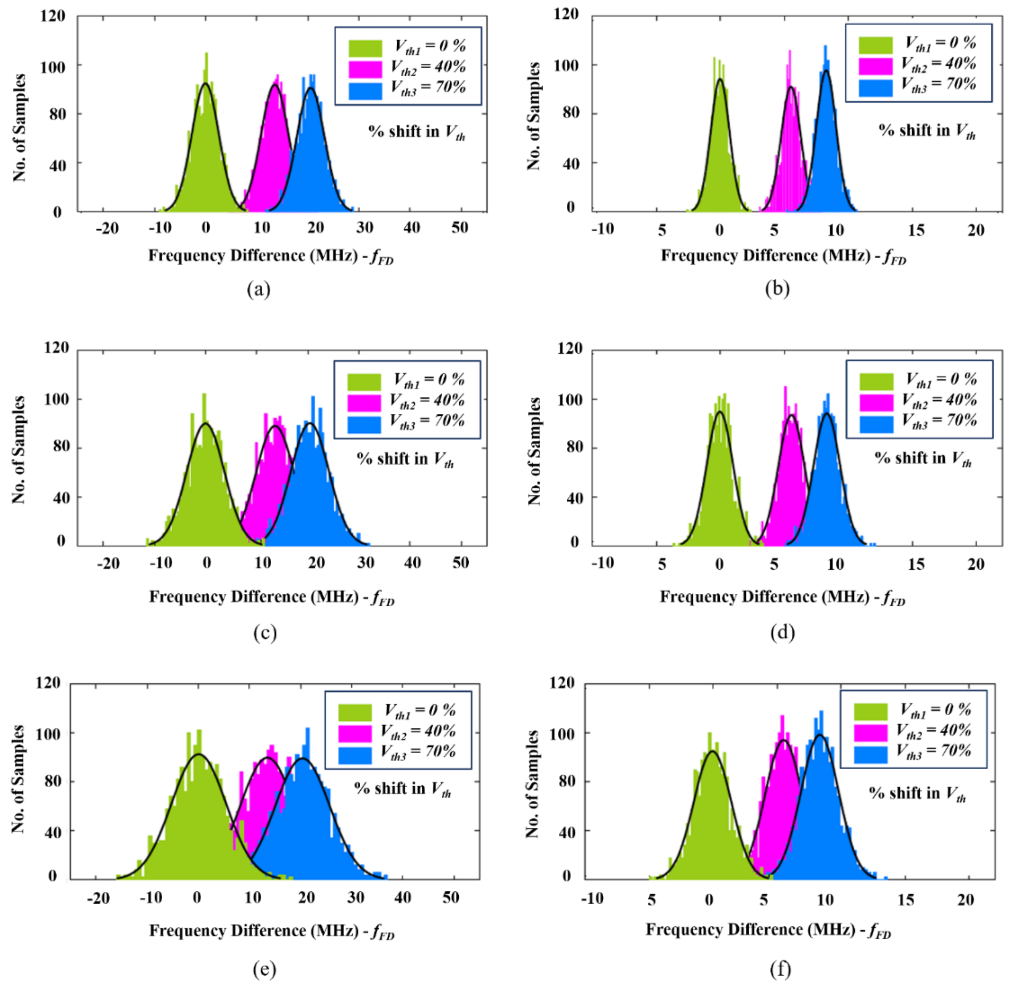


Figure 5-22. Distribution of frequency differences between FSS and DSS, f_{FD}, with percentage shifts in threshold voltage in the presence of process variations PV_a, PV_b, and PV_c and changing number of RO stages (9 and 31) in sensor segments. (a) PV_a: 9-stage RO, (b) PV_a: 31-stage RO, (c) PV_b: 9-stage RO, (d) PV_b: 31-stage RO, (e) PV_c: 9-stage RO, (f) PV_c: 31-stage RO.

Table 5-6 False Prediction Rates (Probability of Error).

No. of RO stages in Sensor Segments	θ_a (%) – Probability of HT-infected FPGA						θ_b (%) – Probability of HT-free FPGA					
	V_{th2}			V_{th3}			V_{th2}			V_{th3}		
	PV_a	PV_b	PV_c	PV_a	PV_b	PV_c	PV_a	PV_b	PV_c	PV_a	PV_b	PV_c
9-stage RO	0.45	2.35	6.29	0	0.12	1.42	0.31	2.19	6.74	0	0.15	1.37
31-stage RO	0	0.22	1.56	0	0	0.11	0	0.25	1.25	0	0	0.13

and results in a much larger change in frequency difference **f_{FD}**. With no distinct overlap of distributions (at **V_{th1}=0%** and **V_{th2}=40%** and **V_{th3}=70%**), there is a strong indication of the presence of hardware Trojan. We can, therefore, positively detect the presence of hardware Trojans with **V_{th2}=40%** in an FPGA under test (28 nm node).

In order to correctly estimate the percentage of false prediction, which is represented by the distributions' overlap, we use Gaussian fit to determine the mean and variance of these distributions to calculate the overlapped area. At this stage, the process variations mentioned in Table 5-5 are taken into account. These variations being part and parcel of every silicon die, tend to affect electrical parameters invariably from die to die and intra-die as well. With **PV_a,** we consider the probability of false prediction as negligible, and the same was observed during the test. The measured false prediction rates of the sensor relating to **HT-free (θ_a)** and **HT-infected (θ_b)** FPGA are elaborated in Table 5-6. These correspond to the process variations mentioned in Table 5-5. It can be seen that the false prediction rate with ***PV_c*** is higher due to a significant difference in frequencies of the 28-nm FPGA under test with a higher percentage of process variations. As a result, the overlapped area between the two distributions grows significantly, thereby reflecting the increase in the probability of error **(θ).** We

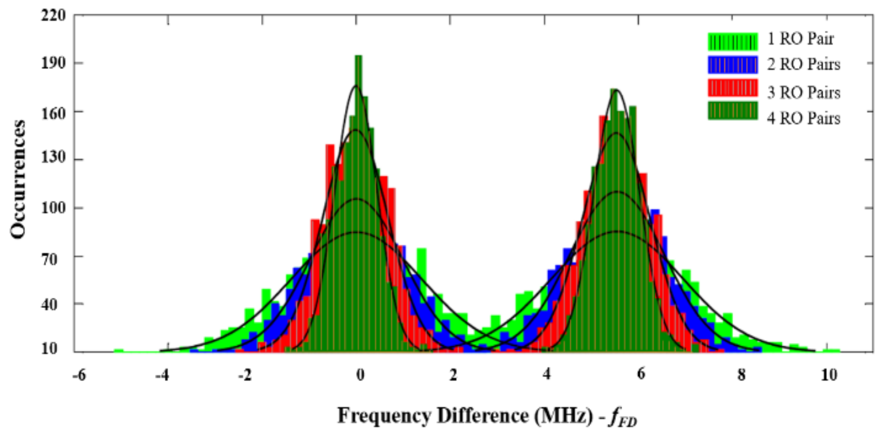
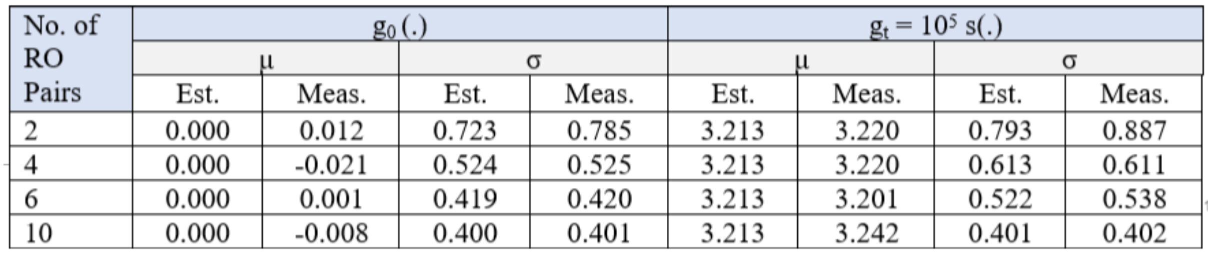


Figure 5-23 Gaussian distribution of frequency difference 'f_{FD}' at PV_c of V_{th}-aware sensor with different number of RO-pairs.

Table 5-7 Mean and Variance Frequency Distribution of Threshold Voltage Aware Sensor.



No. of RO Pairs	g_0 (.)				$g_t = 10^5$ s(.)			
	μ		σ		μ		σ	
	Est.	Meas.	Est.	Meas.	Est.	Meas.	Est.	Meas.
2	0.000	0.012	0.723	0.785	3.213	3.220	0.793	0.887
4	0.000	-0.021	0.524	0.525	3.213	3.220	0.613	0.611
6	0.000	0.001	0.419	0.420	3.213	3.201	0.522	0.538
10	0.000	-0.008	0.400	0.401	3.213	3.242	0.401	0.402

provided remediation by placing the two sensor segments very close to each other, as mentioned earlier. Besides, we increased the number of RO stages in both the segments from 9 to 31 and then observed any reduction in false prediction rate. A

significantly lower false prediction rate is noted (at worst case ***PV_c* – 1.42% to 0.11%**) in the case of **θ_b**, and a similar trend is noted for **θ_a** (at worst case ***PV_c* – 1.37% to 0.13%**).

The histogram plot giving the average frequency difference between the FSS and DSS sensor segments for the different number of pairs is shown in Figure 5-23. We observe a substantial reduction in the spread of the distributions with the increase in the number of RO-pairs. The separation between the two distributions, however, remains the same. At this point, the threshold frequency f_{th} is measured for all the RO-pairs of the two segments and is found to be equal to 2.5 MHz. It becomes crucial at this stage to analyse the changes in the mean **(μ)** and variance (**σ**) values of the frequency difference distribution of sensor segments to estimate the false prediction accuracy to assess any requirement to increase the number of RO-pairs for achieving a negligible false prediction rate. We took the measurements of the mean and variance of different distributions with different numbers of RO-pairs using the '*normfit* MATLAB function' to determine the accuracy of our process flows.

The measured values of the mean and variance are given in Table 5-7. The analysis revealed an error in the expected value when compared with the actual value *(<0.4% for μ and <6% for σ)*. In light of this analysis, we created another histogram plot, as shown in Figure 5-24(a)-(c), based on the frequency difference between the selected RO-pairs of FSS and DSS sensor segments to determine the most efficient and error-free hardware Trojan detection pair. We observe a significant overlap gap between the two distributions at time ***t=0*** and time ***t***.

Also, the increase in the separation between the distributions is found to be positively correlated to an increase in the number of RO-pairs. The threshold frequency f_{th}, in this case, is measured to be 2 MHz. We found the two RO-pairs combination to be the

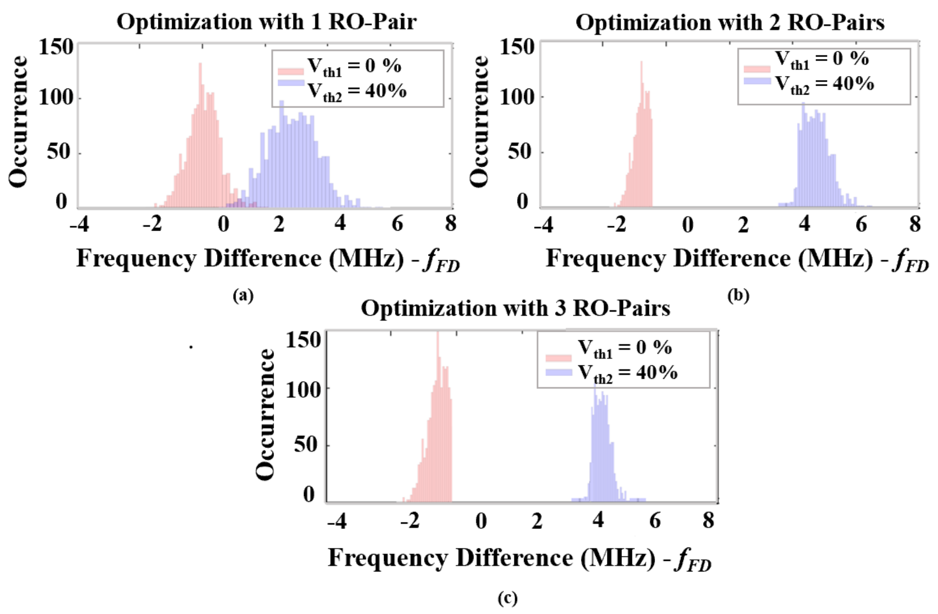


Figure 5-24 Histograms of frequency difference distribution f_{FD} at PV_c of V_{th}-aware sensor with different number of RO-pairs. (a) Optimization with 1RO-pair. (b) Optimization with 2 RO-pairs. (c) Optimization with 3 RO-pairs.

most appropriate with zero-false prediction. The detection accuracy of the sensor is presented in Table 5-8. The rate of false prediction is calculated as:

$$\theta_a = \frac{no.of\ test\ samples\ with\ f_{FD} < f_{th}}{Total\ test\ samples} \times 100\% \qquad (5-3)$$

Table 5-8 Analysis of False Prediction – Improving Sensor Accuracy with RO-pairs scaling and selection process.

Percent Shift in Threshold Voltage (V_{th})	RO Pairs (n)	Probability of Error		f_{th} (MHz)
		θ_a (%)	θ_b (%)	
5%	1	13.07	12.85	0.4
	2	5.46	5.51	0.1
	3	2.73	2.77	0
10%	1	9.0	8.9	0.7
	2	2.8	2.79	0.2
	3	1.1	1.15	0
40%	1	5.58	5.38	1.0
	2	1.24	1.12	0.5
	3	0.32	0.34	0
70%	1	3.91	3.75	1.0
	2	0.64	0.58	0.5
	3	0.12	0.14	0
100%	1	1.82	1.65	1.4
	2	0.14	0.14	1.0
	3	0	0	0

$$\theta_b = \frac{no.of\ test\ samples\ with\ f_{FD} > f_{th}}{Total\ test\ samples} \times 100\% \qquad (5-4)$$

At different threshold voltage shift states, the impact on sensor accuracy with varying number of RO-pairs corresponding to each sensor segment is shown. As mentioned in previous paragraphs, we have implemented a maximum of 3 pairs of ROs each in two sensor segments, FSS and DSS. With a configuration of 2 RO-pairs, we can characterise the sensor to determine threshold frequency **'f_{th}'** corresponding to pre-Trojan trigger threshold voltage **'V_{th_ptt}'** and provide a benchmark to compare the frequency differences of FSS and DSS **'f_{FD}'** with it for the detection of hardware Trojan, once triggered without any probability of error. It is essential to set the threshold frequency cautiously to ensure that the value of the probability of error of FPGAs falsely identified as **HT-free (θ_a)** is similar to the value of the probability of error of FPGAs falsely identified as **HT-infected (θ_b)**.

5.4.6 Area Overhead Analysis

The implementation of a threshold voltage triggered hardware Trojan detection scheme is optimized to utilize minimum resources of 28 nm technology node FPGA. Accordingly, the area overhead analysis of both the infection and detection schemes is shown in Table 5-9. We implemented IWLS 2005 benchmarks of various sizes from low to high to assess the area overhead - the ratio of the size versus area of the sensor with the size versus area of the benchmark. As is evident, when used with a 31-stage sensor in HT detection scheme, the area overhead is approximately **1.25% for n = 2** (2 RO-pairs) for smaller sized benchmarks like i2c, spi, and b14. We observe that it

Table 5-9 Area Overhead Analysis of Threshold Voltage-Aware Sensor (S_{vth}).

Benchmark	Size (No. of Gates)	Sensor [46]	Area Overhead (%)		
			Proposed V_{th}-aware Sensor (31-stages)		
			n=2	n=4	n=6
Vga_lcd	124031	5.23	0.189	0.363	0.93
Ethernet	46771	0.13	0.465	0.996	1.24
DSP	32436	0.19	0.604	1.25	2.32
b15	12562	0.50	1.867	3.387	5.11
b14	8679	0.73	1.258	5.047	2.89
spi	3277	1.92	1.474	3.37	3.90
i2c	1142	5.52	1.23	3.35	5.12

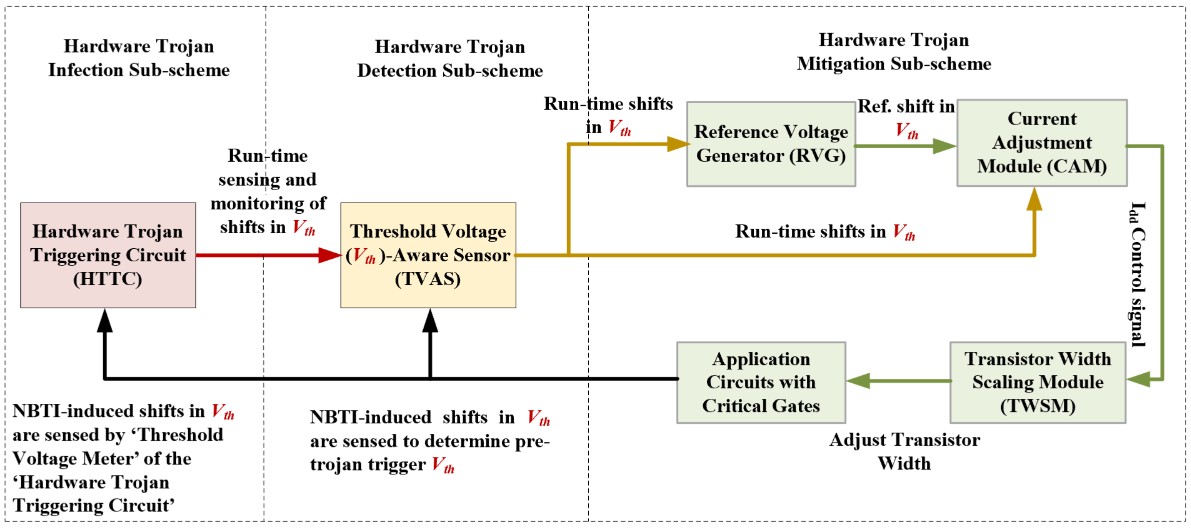

Hardware Trojan Infection Sub-scheme
Hardware Trojan Detection Sub-scheme
Hardware Trojan Mitigation Sub-scheme
Run-time shifts in V_{th}
Reference Voltage Generator (RVG)
Ref. shift in V_{th}
Current Adjustment Module (CAM)
Run-time sensing and monitoring of shifts in V_{th}
Hardware Trojan Triggering Circuit (HTTC)
Threshold Voltage (V_{th})-Aware Sensor (TVAS)
Run-time shifts in V_{th}
I_{dd} Control signal
Application Circuits with Critical Gates
Transistor Width Scaling Module (TWSM)
NBTI-induced shifts in V_{th} are sensed by 'Threshold Voltage Meter' of the 'Hardware Trojan Triggering Circuit'
NBTI-induced shifts in V_{th} are sensed to determine pre-trojan trigger V_{th}
Adjust Transistor Width

Figure 5-25. Block diagram representation of FPGA security scheme highlighting hardware Trojan mitigation sub-scheme.

does not impact the overall area of small as well as medium and larger designs, implemented for heavy systems like the system processor module of ISPS, in our case. On average, the overall area occupied by the HT-detection scheme is measured to be **125µm^2**, whereas the power consumption reads **3.8µW**, which is considered compatible with the designs discussed in Section-5.2.

5.5 Mitigating the Impact of Threshold Voltage – Triggered Hardware Trojan

The final proposition of FPGA security scheme (Figure 5.4) is the design and implementation of hardware Trojan mitigation strategy. We propose a circuit design technique, which endures threshold voltage-triggered hardware Trojans. The internal module structure and control process flow devised for this purpose are depicted in Figure 5- 25 and Figure 5-26 respectively. For this scheme, we target the monitoring of drain current ***'I_{dd}'*** as a parameter that contributes to performance degradation as a result of shifts in threshold voltage. A mechanism is proposed whereby a change in the threshold voltage is sensed and a corresponding adjustment in ***I_{dd}*** is made to compensate for current variations in critical circuit nodes implemented in FPGA.

The main elements added to form the mitigation scheme are the 'Current Adjustment Module,' 'Reference Voltage Generator,' and the 'Transistor Width Scaling Module.' IWLS 2005 benchmark 'vga_lcd' is used as a test circuit implemented in 28-nm FPGA to validate the HT mitigation scheme. It also includes the process of pinpointing the

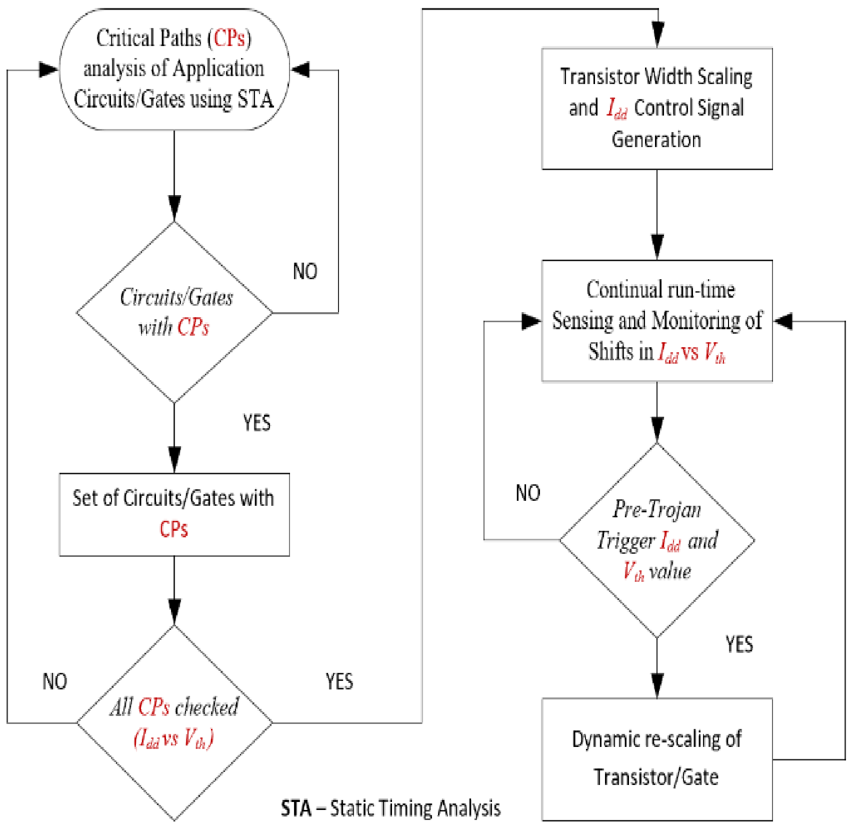

Critical Paths (CPs) analysis of Application Circuits/Gates using STA
Circuits/Gates with CPs
NO
YES
Set of Circuits/Gates with CPs
All CPs checked (I_{dd} vs V_{th})
NO
YES
STA – Static Timing Analysis
Transistor Width Scaling and I_{dd} Control Signal Generation
Continual run-time Sensing and Monitoring of Shifts in I_{dd} vs V_{th}
Pre-Trojan Trigger I_{dd} and V_{th} value
NO
YES
Dynamic re-scaling of Transistor/Gate

Figure 5-26. The Process Flow of Hardware Trojan Mitigation Scheme.

potential critical gates that experience frequency degradation due to the impact of NBTI through shifts in threshold voltage. The Current Adjustment Module (CAM) gauges the acceptable limits and ranges of shifts in threshold voltage, fanned out by the sensor (in our case, the $\boldsymbol{V_{th\_ptt}}$). If $\boldsymbol{V_{th\_ptt}}$ *(pre-trojan trigger threshold voltage)* is out of the acceptable limit, the control signal is given to the Transistor Width Scaling Module (TWSM), which increases the transistor width to counter the excess threshold voltage shift and prevent the triggering of hardware Trojan.

5.5.1 Earmarking the Potential Critical Gates

We implemented the IWLS 2005 benchmark 'vga_lcd' in 28-nm FPGA using the Vivado design suite and applied the algorithm defined in [52] to pinpoint its potential critical gates using static timing analysis. We conclude that only 2.5% of the total gates are identifiable as the potential critical gates, based on the worst-case frequency/delay degradation. The worst-case degradation is set against the $\boldsymbol{V_{th\_ptt}}$. Accordingly, a reserve transistor width is allocated to the earmarked critical gates to increase **'I_{dd}'**

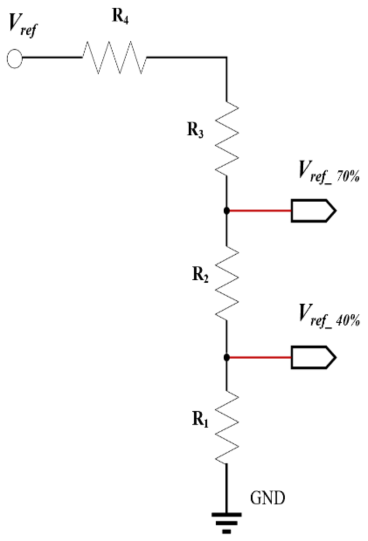


Figure 5-27. Resistive Voltage Divider for Reference Voltage Generator (Rvg).

and counter the impact of the increased threshold voltage. The details of the implementation are described later in Section 5.5.4.

5.5.2 Reference Voltage Generator

The measurement of the threshold voltage is done using 'Threshold Voltage Meter'(Figure 5-12). Although we have used the percentage frequency differences corresponding to specific threshold voltage shifts in the HT detection scheme, we consider it prudent to quantify the impact of shifts in threshold voltage due to NBTI, while devising HT mitigation scheme. In this regard, we propose the implementation of a 'Reference Voltage Generator' comprising a resistive-based voltage divider. The schematic of the generator is shown in Figure 5-27. While calculating the reference voltages, the effect of resistive tolerance is taken into account. Resultantly, for the threshold voltage shifts of 40% and 70%, for instance, we represent them correspondingly as $\boldsymbol{V_{ref\_40\%}}$ and $\boldsymbol{V_{ref\_70\ \%}}$. In order to determine the effect of resistive tolerance variations, we carried out Monte Carlo simulation, taking into account the process and environmental variations as well. A maximum change in reference voltage ΔV_{ref} of less than 4mV is observed at a **worst-case** resistive variation of ± 5%. Whereas at **nominal (± 3%)** and **best case (± 0.5%)** variations, $\boldsymbol{\Delta V_{ref}}$ **of less than 2mV** and **0.75mV** respectively are noted.

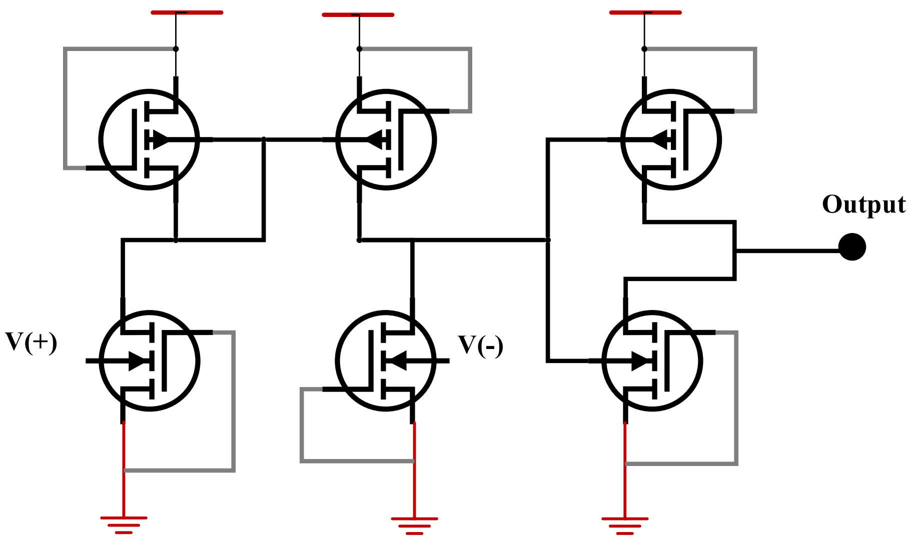


Figure 5-28. A Comparator circuit with current-mirror based differential amplifier.

5.5.3 Current Adjustment Module

Since the shift in threshold voltage of a PMOS device results in the reduction of drain current and the subsequent slowing down of the circuit speed, it is possible to reverse or mitigate this phenomenon by increasing the drain current. In order to achieve this, a comparator circuit comprising current-mirror based differential amplifier is implemented as a current adjustment module. The schematic of this module is shown in Figure 5-28. Here, the output of the HT detection scheme and the reference voltage

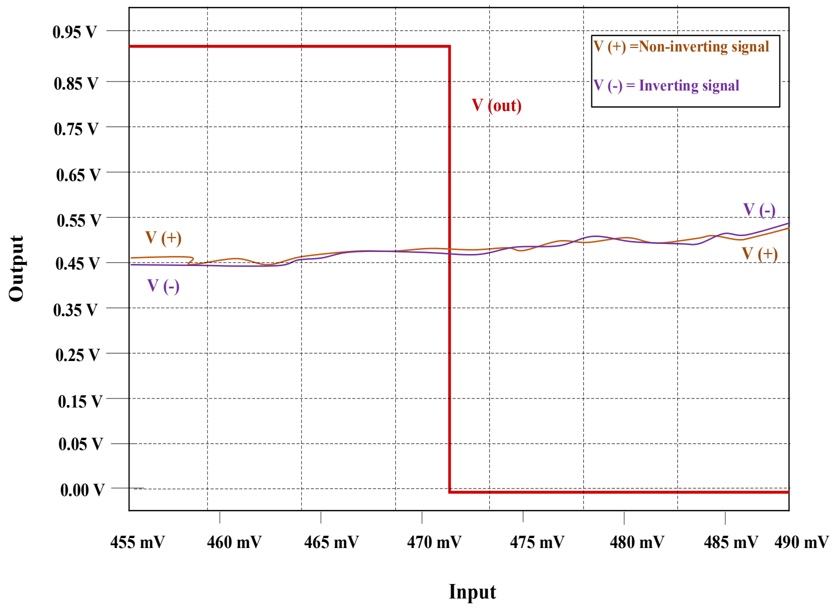


Figure 5-29 Input / Output Response of a Comparator.

generator drive the inputs of the current adjustment module. A control signal from the current adjustment module is provided to the TWSM module, which subsequently increases the width of the transistor to counter the frequency degradation/delay impact of the NBTI mechanism.

In order to check the operation-ability of this module, we induce a fractional change at the inverting and non-inverting inputs of the comparator, as shown in Figure 5-29. When the voltage on the inverting terminal of the comparator is made higher as compared to its non-inverting terminal, the comparator switches to logic ‘0’ and vice versa. We considered the impact of process variations as well and found the comparator sensitive up to **1.5mV** of variation between inverting and non-inverting terminals.

5.5.4 Transistor Width Scaling Module

Increasing the transistor width to let more current pass through the transistor can be implemented as a countermeasure against the threshold voltage triggered hardware Trojans to mitigate the latency induced by the shift in threshold voltage [53]. However, designing transistor width increment as a one-time design rule makes it ineffective against the long-run online performance degradation caused by NBTI ageing mechanism [53]. Also, device upsizing could inflict constraints on the design specification during the design stage. Many design metrics, like impedance matching and Q point of V-I curve, may be affected, which may result in excess drain current values. It is for these reasons, we propose a hardware Trojan mitigation scheme that adjusts the width of transistors dynamically (i.e., during run-time) and named as **‘Online Transistor Dynamic Scaling (OTDS)**. We divide OTDS into two implementation phases as follows:

5.5.4.1 Design Phase

In the design phase, we define the dimensions of the 2.5% of identified critical gates of IWLS 2005 benchmark ‘vga_lcd’ in-line with its I/O functional specification. Additionally, we provide the threshold voltage compensation dimensions/sizing as a backup for the potential critical gates. As per the design, the dimensions of the transistor forming the critical gate remain fixed until it is sensitized by a significant NBTI impact on the design embedded in FPGA.

5.5.4.2 Dynamic Phase

As mentioned in the above paragraphs, when threshold voltage begins to change (increase with NBTI), a runtime decision will be asserted to increase the width of the critical transistors. With an increase in transistor width, the device is supported with a corresponding increase in its drain current and hence, balances and mitigates the impact of threshold voltage shifts.

The concept is illustrated in Figure 5-30. It shows an inverter having a PMOS double the size of its NMOS counterpart. Under the normal situation, the pull-up network possesses two unconnected parallel widths (2xW2 and 2xW3). Similarly, the pull-down network consists of two unconnected parallel widths (W2 and W3). We gated the additional PMOS widths, 2xW2 and 2xW3, using transistors Q1 and Q3, respectively. Similarly, the additional NMOS widths W2 and W3 are also gated using the transistors Q2 and Q4 respectively. The transistors Q1 and Q2 are set to share the same triggering signal from node X whereas Q3 and Q4 share the identical signal from node Y. Under the normal condition, defined as $\boldsymbol{V_{th} < V_{th\_ptt}}$, all these transistors remain dormant ('Off State') and are considered to be a unit sized transistors. As the threshold voltage is shifted $\boldsymbol{(V_{th} \geq V_{th\_ptt})}$ with bias and temperature stressed NBTI, the OTDS technique tries to compensate its impact by selecting transistors of larger widths. At this stage, the reference voltage generator provides steps of percentage voltage corresponding to percentage shifts in threshold voltage. When an increase of 30% in

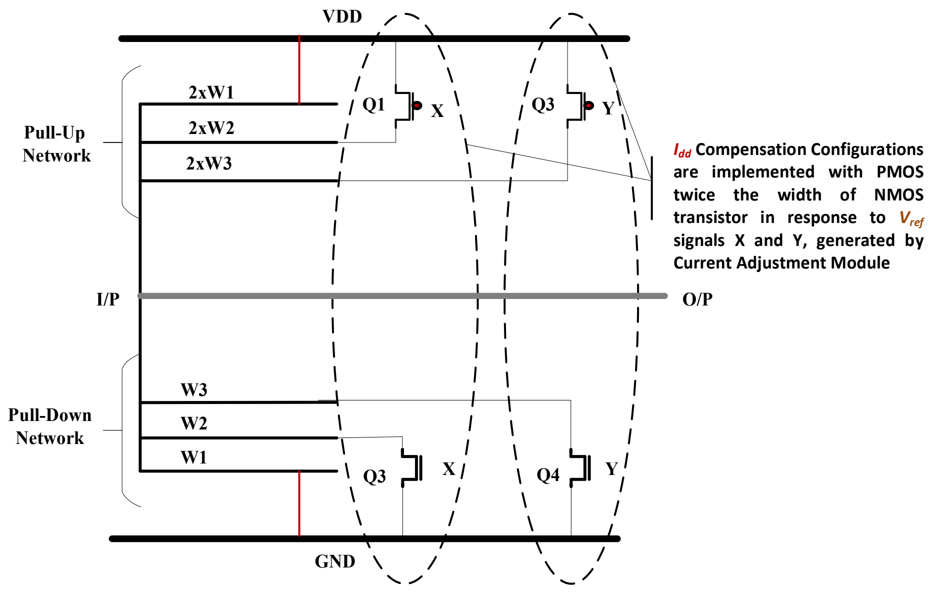

VDD
2xW1
2xW2
2xW3
Pull-Up Network
Q1
X
Q3
Y
I_{dd} Compensation Configurations are implemented with PMOS twice the width of NMOS transistor in response to V_{ref} signals X and Y, generated by Current Adjustment Module
I/P
O/P
Pull-Down Network
W3
W2
W1
Q3
X
Q4
Y
GND

Figure 5-30. Online Transistor Dynamic Scaling using Pull-Up and Pull-Down Networks.

the threshold voltage of the PMOS transistor is reached, the transistor width is incremented to counter the shift in threshold voltage to prevent HT triggering.

It is vital to have an accurate reference voltage step generation for effective mitigation of the increased threshold voltage and the frequency/delay degradation of the circuit application. For that purpose, we assume the reference voltages to be fixed and the run-time or dynamic state decision is made using the values of threshold voltage measured by the HT detection scheme sensor. During the experiment, we observe that as the threshold voltage rises by 5%, the current adjustment module with a corresponding reference voltage (V_{ref}) generates a signal X, which activates the transistors Q1 and Q2 and turns them 'ON.' At this point, the width of the Pull-Up network, shown in Figure 5-30, increases by 2xW2 and so does the width of the Pull-Down network by W2. In the same way, at some instances of the time interval, the signal Y gets triggered with a specific reference voltage, which in turn, activates the transistors Q3 and Q4, having widths as shown in Figure 5-30. This leaves the Pull-Up and Pull-Down networks with improved speed and stability.

5.6 Implementation and Optimisation of Hardware Trojan Mitigation Scheme

It is well established that the drain current ***'I_{dd}'*** and the response time of a MOSFET are directly proportional to its width. Therefore, increasing the transistor's width will subsequently increase the drain current as well as its response time. So, in order to double the transistor width, we may use an equal width transistor to widen the

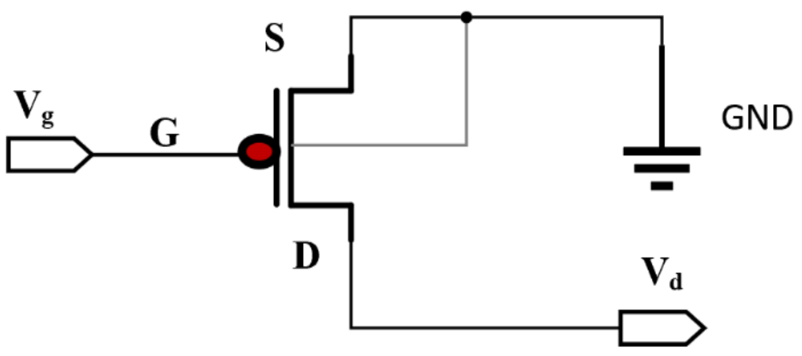


Figure 5-31. Circuitry for Transistor Width Parametric Analysis.

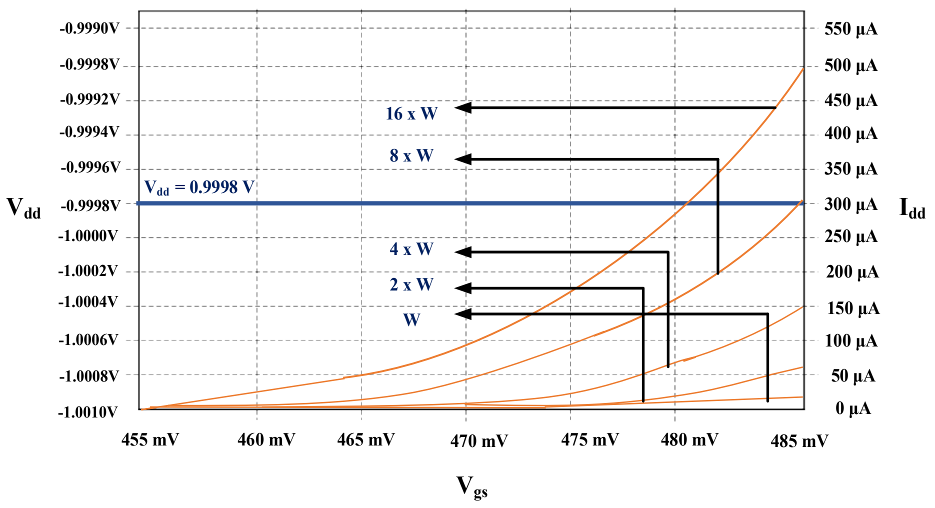


Figure 5-32. I_{dd} vs V_{gs} Curves Showing Online Transistor Width Increment to Compensate for Threshold Voltage-Triggered Hardware Trojan (Ht_{vth}) Attack.

MOSFET by sharing the drain and source terminals between MOSFETs. It also helps in minimising the layout area.

Before deciding the extent of increasing the width of the transistor to reverse current reduction due to NBTI, we quantify the reduction in drain current ***'I_{dd}'***. Accordingly, we measure ***'I_{dd}'*** at 0%, 10%, 30%, 60%, and 90% of shift in ***V_{th}***. The measurement results are listed in Table 5-10. Based upon these measurements, a width-based parametric analysis of the PMOS transistor is undertaken to make a correct assessment of the extent of its width increment required to reverse ***'I_{dd}'*** reduction, corresponding to percentage shifts in ***V_{th}***. This analysis is enabled by the circuitry shown in Figure 5-31. As can be seen, we kept the gate and source voltages of the PMOS transistor constant at -1V and 0V, respectively and noted the variation in width (W) of the transistor. The results are shown in Figure 5-32. It is evident that for a given gate and source voltages, the drain current increases two-fold as the width of the PMOS device is doubled. So, accordingly, we come up with the requisite percentage of width

Table 5-10 Measured values – PMOS I_{dd} reduction with increase in V_{th}.

% age Shift in Threshold Voltage V_{th} (Increment)	PMOS Drain Current I_{dd} (µA) (28 nm – V_{gs}=0.4V and V_{ds}=0.9V)
0	25
10	20
30	13
60	7
90	2.5

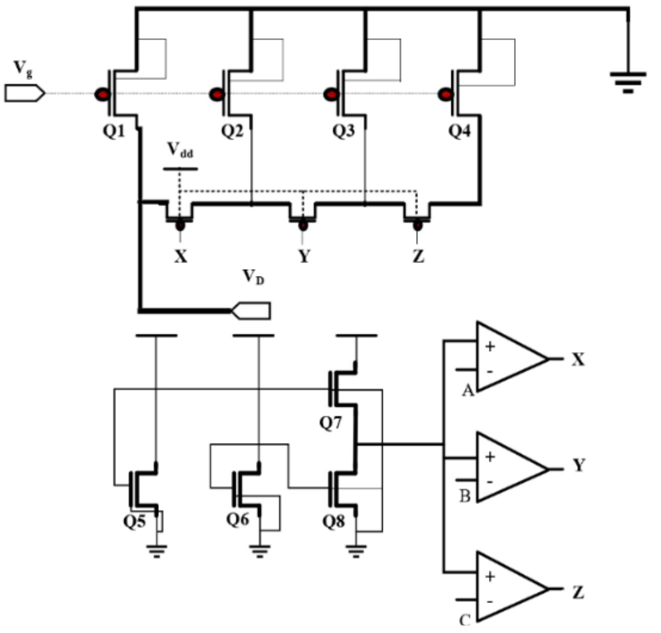


Figure 5-33. Threshold voltage-triggered hardware Trojan mitigation circuitry of 'HT-Mitigation Scheme'

increment, which is added in parallel for each value of shift in threshold voltage to increase the transistor's width and the current flow through it. The implementation of this scheme is elaborated in Figure 5-33.

We employ the unit size transistor as a switch to manage and control the connectivity of a transistor width for compensation. As seen in Figure 5-33, Q1 represents the critical gate, and Q2, Q3, and Q4 are the widths reserved to compensate for the reduction of ***'I_{dd}'*** due to percentage ***V_{th}*** shifts. As mentioned earlier, the sizes of Q2, Q3, Q4, and Q5 are defined at the design phase. The same are given in Table 5-11.

In order to validate the mitigation scheme, the circuitry in Figure 5-33 is applied to a flip flop with true single-phase clocking function. We measure the rise and fall times of the flip flop as they change with changes in the threshold voltage. The results show an increase in the rise and fall times with an increase in ***V_{th}*** shifts. The exact values

Table 5-11 Measured values – width increment (Fanout-4) with shifts in V_{th}.

% age Shift in Threshold Voltage V_{th} (Increment)	PMOS Drain Current I_{dd} (μA) (28 nm – V_{gs}=0.4V and V_{ds}=0.9V)	Transistor Width Increment (FO-4)
0	25	NR
10	20	≃1.0
30	13	≃1.2
60	7	≃1.5
90	2.5	≃2.2

Table 5-12 Timing delays in TSPC due to V_{th}-triggered hardware Trojan payload.

V_{th} (V)	% age Shift in Threshold Voltage V_{th}	Rise Time Delay (ps)	Fall Time Delay (ps)
0.450	0	22.5	28.7
0.495	10	23.1	30.5
0.585	30	25.7	32.3
0.720	60	29.4	35.6
0.855	90	34.7	40.5

are covered in Table 5-12. We observe that as a result of this increase, momentary state transitions occur in FSM, which may lead to changing the output state. Also, we note that as the duration of this output state is extended, it gets latched and may result in the activation of malicious and stealthy hardware Trojan. This, however, is prevented by increasing the device width and resultantly, the triggering signal for the Trojan is silenced.

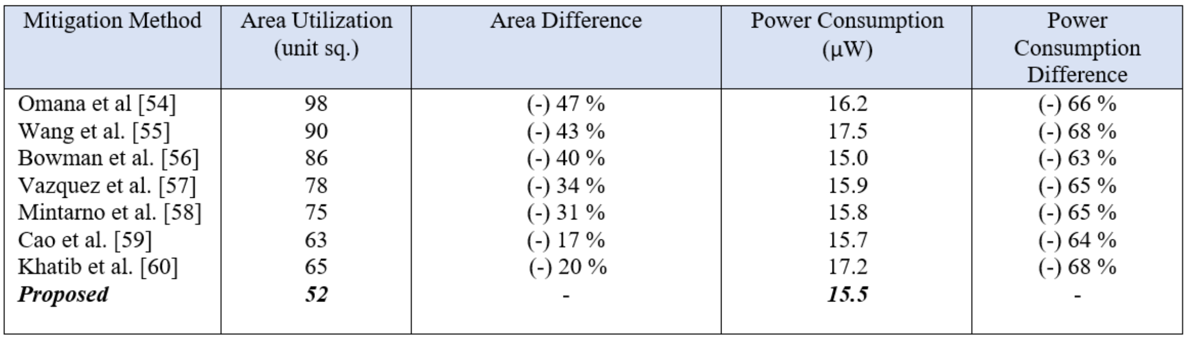
In a nutshell, adding extra reserve width for Pull-Up and Pull-Down network in the design phase provides a viable mitigation technique, which increases the transistor width dynamically during the run-time.

5.6.1 Comparative Analysis with Contemporary Mitigation Techniques

We have presented a holistic FPGA security scheme to detect and mitigate the ingress of threshold voltage triggered hardware Trojans in its fabric. In doing so, we have designed, implemented, and validated HT-infection, HT-detection, and HT-mitigation schemes, with novel sensing and monitoring elements. We have highlighted its significance in the ship-defence environment by providing a threat scenario/model based on an 'Integrated Self-Protection System (ISPS).' This is a unique effort that puts forth an integrated approach towards visualising and addressing a probable hardware Trojan presence in a security-sensitive and mission-critical defence system with accurate and resource-efficient detection and mitigation circuitry in a 28 nm technology node based FPGA.

As discussed in Section 5.2, a significant amount of research work has been undertaken to develop effective methods and circuits. In this section, we make a comparative analysis of our work with other existing methods for the mitigation of the NBTI effect in integrated circuits. For instance, in [54], the adaptive clock scheme

Table 5-13 Area and Power consumption comparison of the proposed Threshold Voltage (V_{th}) -shift based HT Mitigation Scheme.



Mitigation Method	Area Utilization (unit sq.)	Area Difference	Power Consumption (μW)	Power Consumption Difference
Omana et al [54]	98	(-) 47 %	16.2	(-) 66 %
Wang et al. [55]	90	(-) 43 %	17.5	(-) 68 %
Bowman et al. [56]	86	(-) 40 %	15.0	(-) 63 %
Vazquez et al. [57]	78	(-) 34 %	15.9	(-) 65 %
Mintarno et al. [58]	75	(-) 31 %	15.8	(-) 65 %
Cao et al. [59]	63	(-) 17 %	15.7	(-) 64 %
Khatib et al. [60]	65	(-) 20 %	17.2	(-) 68 %
Proposed	***52***	-	***15.5***	-

entails increasing the clock time to address the worst-case performance (in terms of signal path time delays) degradation due to NBTI. This scheme is, however, hardware-intensive with a high area overhead. Also, it degrades the device performance as a result of time guard banding. Another technique [36] implies the replacement of aged gates to reverse delay degradation but, again, it results in high area overhead. Our work, on the contrary, addresses performance degradation by changing the transistor width dynamically (during the runtime). This entails low area overhead and enhanced device performance.

In another scheme [55], device ageing due to NBTI is countered through standard-cell sensor-facilitated measurement of frequency degradation. It is followed by inducing additional timing margin for the critical path to prevent device failure due to continued ageing. However, the provision of redundancy in terms of extra timing margin is not always valid. Moreover, such kind of schemes is resource-intensive with increased area overheads–an undesired feature in modern technology nodes.

Table 5-13 summarises the analysis in terms of efficiency with respect to area overhead and power consumption. We find the HT-mitigation component of our FPGA security scheme more resource-efficient with compatible power consumption. It augments the device performance by zeroing the impact of shifts in threshold voltage through responsive and dynamic scaling of transistor width rather than the replacement of the gate/transistor.

5.7 Summary

The miniaturised form factor of modern FPGAs provides enhanced performance as compared to their predecessors. However, high-temperature stresses coupled with longer heat dissipation paths may cause undesired stochastic variations like signal delays. Primarily, this is attributable to the negative bias temperature instability (NBTI) ageing mechanism that comes into play as a result of elevated temperature and negative bias stress conditions. Consequently, the threshold voltage increases, which in turn, leads to reduced drain current and delay degradation.

Keeping the aforementioned in perspective, we have investigated the impact of threshold voltage shifts due to the degradation mechanism of NBTI in a 28 nm technology node and constructed an FPGA security scheme around it to counter potential hardware Trojan (HT) threats. The development of a threat scenario/model encompassing a naval warship's integrated self-protection system (ISPS), with its processor module in focus, reinforces the need for a holistic approach to hardware Trojan threats. We have shown how a rogue element in a design house can make use of knowledge about the shifts in threshold voltage of a PMOS transistor to design and implement a stealthy hardware Trojan scheme comprising heating elements, threshold voltage meter, and the Trojan circuit. The area and power consumption for this scheme are kept as low as **50µm^2** and **1.05 µW** for NAND2 and **75µm^2** and **1.25µW** for TSPC, with the hardware Trojans triggering at **40%** and **50%** of the shift in threshold voltages, respectively. It results in the total collapse of the circuit functionality, thereby confirming the paralysing effect it can have on the ISPS system capability of a warship. Acting as a defender, we have created hardware Trojan detection and mitigation schemes as an integral part of the overall FPGA security scheme. The HT-detection scheme is composed of a highly sensitive (**100 KHz/0.5 mV**) ring oscillator pair-based sensor. It measures frequency degradation in a dynamic sensor segment (DSS) RO pair equivalent to the shifts in threshold voltage and compares it with the fixed sensor segment (FSS). The sensor is tested and calibrated to detect frequency degradation at the pre-Trojan Trigger threshold voltage 'V_{th_ptt}' and Trojan Trigger threshold voltage 'V_{th_tt}.' The detection and measurement accuracy is achieved by reducing the false prediction rate to zero. Area overhead of **125µm^2** and compatible power consumption of **3.8µW** are noted for the HT-detection scheme.

The final part of our FPGA security scheme is HT-mitigation by online transistor dynamic scaling (OTDS). Here, we leverage the reduction in drain current with an increase in threshold voltage to dynamically adjust the transistor width and reverse the HT triggering process. Post parametric analysis of the changes in the transistor width, we conclude that increasing the transistor width improves its drain current flow, which in turn, helps maintain the performance of the FPGA and avoid HT triggering. We correlated and back annotated the requisite increment/decrement in the transistor width to compensate for the drain current loss due to shifts in threshold voltage. Accordingly, a range of transistor widths that compensates for the reduction in drain current has been determined in the FPGA under test. This HT-mitigation scheme occupies an area of **150μm^2** with power consumption at **15.5 μW**.

The whole FPGA security scheme is built on changes in the threshold voltage of the PMOS transistor. It provides a unique and integrated strategy for thwarting the probable infection of threshold voltage-triggered hardware Trojans in advanced re-programmable devices used in security-critical defence systems.

REFERENCES

[1] S. F. Mossa, S. R. Hasan, and O. Elkeelany, “Hardware trojans in 3-D ICs due to NBTI effects and countermeasure,” *Integr. VLSI J.*, vol. 59, pp. 64–74, 2017.

[2] Y. Wang *et al.*, “High Temperature Thermal Management with Boron Nitride Nanosheets,” *Nanoscale*, pp. 167–173, 2017.

[3] E. A. Scott, J. T. Gaskins, and S. W. King, “Thermal conductivity and thermal boundary resistance of atomic layer deposited high- k dielectric aluminum oxide , hafnium oxide , and titanium oxide thin films on silicon,” APL Materials 6, 2018.

[4] P. Mangalagiri, S. Bae, R. Krishnan, Yuan Xie and V. Narayanan, "Thermal-aware reliability analysis for Platform FPGAs," *IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, pp. 722-727, 2008.

[5] Y. Wang, H. Luo, K. He, R. Luo, H. Yang, and Y. Xie, “Temperature-aware NBTI modeling and the impact of standby leakage reduction techniques on circuit performance degradation,” *IEEE Trans. Dependable Secur. Comput.*, vol. 8, no. 5, pp. 756–769, 2011.

[6] T. Grasser, R. Entner, O. Triebl, H. Enichlmair and R. Minixhofer, "TCAD Modeling of Negative Bias Temperature Instability," *International Conference on Simulation of Semiconductor Processes and Devices*, Monterey, CA, pp. 330-333, 2006.

[7] A. Waksman and S. Sethumadhavan, “Silencing hardware backdoors,” *Proc. - IEEE Symp. Secur. Priv.*, pp. 49–63, 2011.

[8] B. Vaidyanathan, A. S. Oates, Y. Xie, and Y. Wang, “NBTI-aware statistical circuit delay assessment,” *10th Int. Symp. Qual. Electron. Des.*, no. 4, pp. 13–18, 2009.

[9] S. Khan and S. Hamdioui, "Temperature dependence of NBTI induced delay," *IEEE 16th International On-Line Testing Symposium*, Corfu, pp. 15-20, 2010.

[10] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, “Stealthy dopant-level hardware Trojans: Extended version,” *J. Cryptogr. Eng.*, vol. 4, no. 1, pp. 19–31, 2014.

[11] D. Patra *et al.*, “Adaptive accelerated aging for 28 nm HKMG technology,” *Microelectron. Reliab.*, vol. 80, pp. 149–154, 2018.

[12] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, “Trojan detection using IC fingerprinting,” *Proc. - IEEE Symp. Secur. Priv.*, pp. 296–310, 2007.

[13] J. Li and J. Lach, “At-speed delay characterization for IC authentication and Trojan horse detection,” *IEEE Int. Work. Hardware-Oriented Secur. Trust. HOST*, pp. 8–14, 2008.

[14] M. Abramovici and P. Bradley, "*Integrated circuit security: new threats and solutions"*. In Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies (CSIIRW '09). Association for Computing Machinery, New York, NY, USA, Article 55, pp 1–3, 2009.

[15] R.S. Chakraboty, F.Wolff, S.Paul, C. Papachristou, and S. Bhunia, *"MERO: A statistical approach for hardware Trojan detection"*, In Proceedings of the 11th International workshop on cryptographic hardware and embedded systems (CHES'09), Springer-Verlog, Belin, pp 396-410, 2009.

[16] S. Narasimhan and D. Du, “Multiple-parameter side-channel analysis: a non-invasive hardware Trojan detection approach,” *Hardware-Oriented Security and Trust*, pp. 13–18, 2010.

[17] C. Lamech, R. M. Rad, M. Tehranipoor, and J. Plusquellic, “An experimental analysis of power and delay signal-to-noise requirements for detecting trojans and methods for achieving the required detection sensitivities,” *IEEE Trans. Inf. Forensics Secur.*, vol. 6, no. 3 PART 2, pp. 1170–1179, 2011.

[18] Xuehui Zhang and M. Tehranipoor, “RON: An on-chip ring oscillator network for hardware Trojan detection,” *Des. Autom. Test Eur.*, vol. 1, pp. 1–6, 2011.

[19] A. Ferraiuolo, X. Zhang, and M. Tehranipoor, “Experimental analysis of a ring oscillator network for hardware trojan detection in a 90nm ASIC,” *Proc. Int. Conf. Comput. Des. - ICCAD ’12*, p. 37, 2012.

[20] Y. Cao, C. Chang and S. Chen, "Cluster-based distributed active current timer for hardware Trojan detection," *IEEE International Symposium on Circuits and Systems (ISCAS)*, Beijing, pp. 1010-1013, 2013.

[21] O. Söll, T. Korak, M. Muehlberghuber and M. Hutter, "EM-based detection of hardware trojans on FPGAs," *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Arlington, VA, pp. 84-87, 2014.

[22] J. Balasch, B. Gierlichs and I. Verbauwhede, "Electromagnetic circuit fingerprints for Hardware Trojan detection," *IEEE International Symposium on Electromagnetic Compatibility (EMC)*, Dresden, pp. 246-251, 2015.

[23] Ngo, X.T., Najm, Z., Bhasin, S. *et al.* "*Method taking into account process dispersion to detect hardware Trojan Horse by side-channel analysis*". *J Cryptogr Eng* 6, pp 239–247, 2016.

[24] P. Singh, E. Karl, D. Blaauw, and D. Sylvester, “Compact Degradation Sensors for Monitoring NBTI and Oxide Degradation,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 9, pp. 1645–1655, 2012.

[25] Y. Wang, M. Enachescu, S. D. Cotofana, and L. Fang, “Microelectronics Reliability Variation tolerant on-chip degradation sensors for dynamic reliability management systems,” *Microelectron. Reliab.*, vol. 52, no. 9–10, pp. 1787–1791, 2012.

[26] Y. Wang and S. D. Cotofana, “Statistical Reliability Analysis of NBTI Impact on FinFET SRAMs and Mitigation Technique Using Independent-Gate Devices,”

IEEE/ACM Int. Symp. Nanoscale Archit., pp. 109–115, 2012.

[27] J. P. D. Comput, Y. Wang, S. D. Cotofana, and L. Fang, “Analysis of the impact of spatial and temporal variations on the stability of SRAM arrays and the mitigation technique using independent-gate devices,” *J. Parallel Distrib. Comput.*, vol. 74, no. 6, pp. 2521–2529, 2014.

[28] S. V. Kumar, C. H. Kim and S. S. Sapatnekar, "NBTI-Aware Synthesis of Digital Circuits," *44th ACM/IEEE Design Automation Conference*, San Diego, CA, pp. 370-375, 2007.

[29] A. Calimera, E. Macii, and M. Poncino, “Design Techniques for NBTI-Tolerant Power-Gating Architectures,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 59, no. 4, pp. 249–253, 2012.

[30] Z. Abbas, M. Olivieri, U. Khalid, A. Ripp and M. Pronath, "Optimal NBTI degradation and PVT variation resistant device sizing in a full adder cell," *4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, Noida, pp. 1-6, 2015.

[31] I. Chao Lin, S.M. Syu, and T.Y. Ho, *"NBTI tolerance and leakage reduction using gate sizing"*, ACM J. Emerg. Technol. Comput. Syst. 11,1, Article 4, pp 12, 2014.

[32] P. Mangalagiri, S. Bae, R. Krishnan, Y. Xie, and V. Narayanan, “Thermal-aware reliability analysis for platform FPGAs,” *IEEE/ACM Int. Conf. Comput. Des. Dig. Tech. Pap. ICCAD*, pp. 722–727, 2008.

[33] K. Wu, D. Marculescu, M. Lee and S. Chang, "Analysis and mitigation of NBTI-induced performance degradation for power-gated circuits," *IEEE/ACM International Symposium on Low Power Electronics and Design*, Fukuoka, pp. 139-144, 2011.

[34] W. H. Choi, H. Kim, and C. H. Kim, “Circuit Techniques for Mitigating Short-Term Vth Instability Issues in Successive Approximation Register (SAR) ADCs,” *IEEE Cust. Integr. Circuits Conf.*, pp. 1–4, 2015.

[35] S. Kiamehr, M. Ebrahimi, F. Firouzi and M. B. Tahoori, "Extending standard cell library for aging mitigation," in *IET Computers & Digital Techniques*, vol. 9, no. 4, pp. 206-212, 7 2015.

[36] G. Zhang, M. Yi, Y. Miao, D. Xu, and H. Liang, “NBTI-induced Circuit Aging Optimization by Protectability-aware Gate Replacement Technique,” *16th Latin-American Test Symp.*, pp. 1–4, 2015.

[37] S. V. Kumar, K. H. Kim and S. S. Sapatnekar, "Impact of NBTI on SRAM read stability and design for reliability," *7th International Symposium on Quality Electronic Design (ISQED'06)*, San Jose, CA, pp. 6 -218, 2006.

[38] T. H. Kim, R. Persaud, and C. H. Kim, “Silicon odometer: An on-chip reliability monitor for measuring frequency degradation of digital circuits,” *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 874–880, 2008.

[39] E. Saneyoshi, K. Nose, and M. Mizuno, “A Precise-Tracking NBTI-Degradation Monitor Independent of NBTI Recovery Effect,” *IEEE Int. Solid-State Circuits Conf.* -, pp. 192–193, 2010.

[40] X. Zhang, M. Tehranipoor, and S. Member, “Design of On-Chip Lightweight Sensors for Effective Detection of Recycled ICs,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 22, no. 5, pp. 1016–1029, 2014.

[41] C. Dong, “A Multi-Layer Hardware Trojan Protection Framework for IoT Chips,” *IEEE Access*, vol. 7, pp. 23628–23639, 2019.

[42] S. Moein, T. A. Gulliver, and S. Member, “A New Characterization of Hardware Trojans,” *IEEE Access*, vol. 4, pp. 2721–2731, 2016.

[43] S. R. Hasan, “An All-Digital Skew-Adaptive Clock Scheduling Algorithm for Heterogeneous Multiprocessor Systems on Chips (MPSoCs),” *IEEE Int. Symp. Circuits Syst.*, pp. 2501–2504, 2009.

[44] S. R. Hasan, S. F. Mossa, O. Sayed, A. Elkeelany, and F. Awwad, “Tenacious Hardware Trojans Due to High Temperature in Middle Tiers of 3-D ICs,” *IEEE 58th Int. Midwest Symp. Circuits Syst.*, pp. 1–4, 2015.

[45] S. Khan and S. Hamdioui, “Temperature Dependence of NBTI Induced Delay,” *2010 IEEE 16th Int. On-Line Test. Symp.*, pp. 15–20, 2010.

[46] A. P. Shah, N. Yadav, A. Beohar, and S. K. Vishvakarma, “SUBHDIP: process variations tolerant subthreshold Darlington pair-based NBTI sensor circuit,” *IET*

Comput. Digit. Tech., vol. 13, no. 3, pp. 243–249, 2019.

[47] Gupta A. A survey on different CMOS full adder design techniques / A.Gupta, R. Thakur // Int. Journal of Advanced Research in Computer Science and Software Engineering, – Vol. 5, No. 7, pp 1196-1201, 2015.

[48] Joshi D. Design and implementation of 16-bit ripple carry adder for low power in 45mm CMOS technology / D.D. Joshi, J.K. Singh // Int. Journal of Emerging Technology and Advanced Engineering, – Vol. 4, No. 1, pp 216-220, 2014.

[49] A. Amouri, F. Bruguier, S. Kiamehr, P. Benoit, L. Torres, and M. Tahoori, “Aging effects in FPGAs: An experimental analysis,” *Conf. Dig. - 24th Int. Conf. F. Program. Log. Appl. FPL 2014*, pp. 5–8, 2014.

[50] X. Zhang, N. Tuzzio and M. Tehranipoor, "Identification of recovered ICs using fingerprints from a light-weight on-chip sensor," *DAC Design Automation Conference 2012*, San Francisco, CA, pp. 703-708, 2012.

[51] U. Guin, D. Forte and M. Tehranipoor, "Design of Accurate Low-Cost On-Chip Structures for Protecting Integrated Circuits Against Recycling," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1233-1246, April 2016.

[52] W. Wang, Z. Wei, S. Yang and Y. Cao, "An efficient method to identify critical gates under circuit aging," *IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, pp. 735-740, 2007.

[53] G. Wu, G. W. Deptuch, J. R. Hoff, and P. Gui, “Degradations of threshold voltage, mobility, and drain current and the dependence on transistor geometry for stressing at 77 K and 300 K,” *IEEE Trans. Device Mater. Reliab.*, vol. 14, no. 1, pp. 477–483, 2014.

[54] M. Omaña, D. Rossi, N. Bosio and C. Metra, "Low Cost NBTI Degradation Detection and Masking Approaches," in *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 496-509, March 2013.

[55] X. Wang *et al.*, “Aging Adaption in Integrated Circuits Using a Novel Built-In Sensor,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 34, no. 1, pp. 109–121, 2015.

[56] K. A. Bowman *et al.*, "Energy-Efficient and Metastability-Immune Resilient

Circuits for Dynamic Variation Tolerance," in *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 49-63, Jan. 2009.

[57] J. C. Vazquez *et al*., "Predictive error detection by on-line aging monitoring," *2010 IEEE 16th International On-Line Testing Symposium*, Corfu, pp. 9-14, 2010.

[58] E. Mintarno, V. Chandra, D. Pietromonaco, R. Aitken, and R. W. Dutton, “Workload Dependent NBTI and PBTI Analysis for a sub-45nm Commercial Microprocessor,” *2013 IEEE Int. Reliab. Phys. Symp.*, pp. 3A.1.1-3A.1.6, 2013.

[59] Y. Cao *et al*., "Cross-Layer Modelling and Simulation of Circuit Reliability," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 1, pp. 8-23, Jan. 2014.

[60] Khatib *et al*., "Degradation analysis of datapath logic subblocks under NBTI aging in FinFET technology," *Fifteenth International Symposium on Quality Electronic Design*, Santa Clara, CA, pp. 473-479, 2014.

[61] G. Lian, W. Chen, S. Huang, and S. Member, “Cloud-Based Online Ageing Monitoring for IoT Devices,” *IEEE Access*, vol. 7, pp. 135964–135971, 2019.

[62] J. U. N. Tong, J. Yang, J. Xi, and P. O. Ogunbona, “Tuning the Parameters for Precision Matrix Estimation Using Regression Analysis,” *IEEE Access*, vol. 7, pp. 90585–90596, 2019.

6 FPGA HEALTH ESTIMATION USING KERNEL LEARNING APPROACH

High thermal and power stresses coupled with increased switching frequencies result in the accelerated degradation in the timing performance of FPGA primitives, such as look-up tables (LUTs), configuration logic blocks (CLBs), and programmable interconnects. Essentially, this is attributable to the deviation in CMOS transistor parameters from their initial values over the operational lifespan of the device. The resulting signal path delays and timing violations, eventually leading to the accelerated ageing, affect the reliability of an FPGA as well as the sensitive applications running on its fabric. The quantification of an FPGA health, under such an accelerated degradation environment, therefore, becomes vital to support its reliable operation and maintainability. Existing approaches to predict degradation and the overall FPGA health are very limited and inconclusive. Accordingly, an FPGA health estimation method is developed using a unique kernel-based machine-learning approach. This chapter is, therefore, organised to give details of the developed method and various interpretations as per the disposition shown in Figure 6-1.

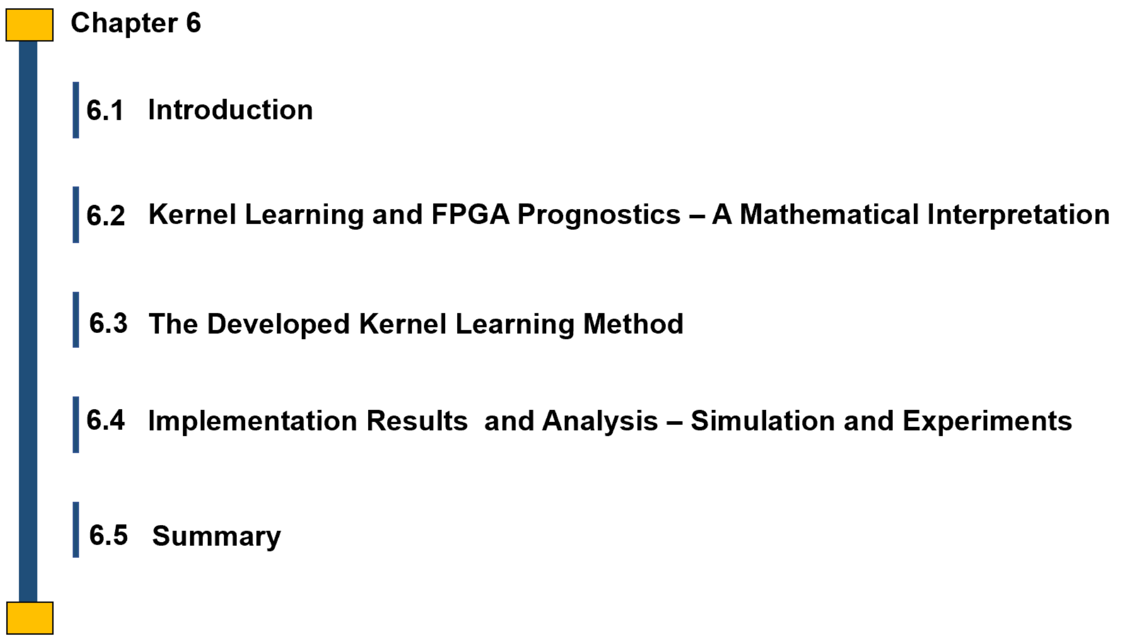

Chapter 6
6.1 Introduction
6.2 Kernel Learning and FPGA Prognostics – A Mathematical Interpretation
6.3 The Developed Kernel Learning Method
6.4 Implementation Results and Analysis – Simulation and Experiments
6.5 Summary

Figure 6-1 The Disposition of Chapter 6.

6.1 Introduction

Health estimation phenomena are not as straightforward in VLSI devices as they are for discrete electronic components. It is because of the complex nature of fault/failure mechanisms, the interdependence of electrical parameters, and varying component tolerances (*due to process variations*) that place challenges on devising realistic and dependable prognostics in VLSI devices - in this case, the FPGAs. These devices composed of primitives such as look-up-tables (LUTs), flip flops and programmable interconnects with underlying circuitry of CMOS transistors exhibit parametric shifts as they degrade over their operational lifetime. The parametric shift is defined as the deviation in the parameters of FPGA primitives due to changing I-V characteristics of CMOS transistors from their initial values and beyond their acceptable tolerance limits [1]. This results in gradual/accelerated (*when infected with stealthy and malicious electronic circuitry - hardware Trojan*) performance degradation in an FPGA, eventually leading to the application as well as the FPGA failure. Predicting such FPGA performance degradation and failures and, hence its health holds the key to maintaining as well as enhancing the reliability and availability of the existing and future system and network environments (SoCs, NoCs, and ACAP), augmented by state-of-the-art FPGAs. It is, therefore, essential to estimate degradation in FPGA health due to parametric shifts (e.g., rise in electric field strength, increased threshold voltage, reduced thermal conductivity, etc.) in its primitives with underlying CMOS transistors. This study focuses on the parametric shifts in CMOS transistors of SRAM Look-up Tables (LUTs) and bistable elements, connected to support logic that could perform predetermined functions (*in this case – the combinatorial and sequential functions*).

6.1.1 Related Work

A limited amount of studies have developed methods to quantify degradation in FPGA health due to parametric shifts in its primitives. Most of the prognostics work has been component-centric using data-driven methods for discrete electronic devices like IGBTs, electrolytic capacitors, lithium batteries etc., For instance, Mahalanobis distance (MD)-based feature transformation has been used by [2] for prognostics as a health indicator (HI). The author in [3] employed Euclidean distance (ED) measure for filter circuit prognostics, which, however, does not take into account the correlation between extracted features. This could result in a false prediction.

On the other hand, [4] and [5] have calculated the health index (HI) as the cosine ($\cos^{-1}$) and sine ($\sin^{-1}$) functions of the distance between the test features and features extracted from the circuit with the no-fault condition. They, however, did not take into account the impact of component tolerances, which in addition to the measurement noise, may affect the accuracy and authenticity of prognosis.

Fractional contributions through Mahalanobis distance measurements, conducted over a specific time window of extracted features, has been demonstrated by [6]. A *k*-nearest neighbor-based prognosis for IGBTs has been proposed by [7], which uses ED measurement between the test data to the centroid of the nearest neighbors. Here, healthy and failure classes are constructed offline.

It is noteworthy that the abovementioned MD and ED measure-based prognostics [2]-[7] are applicable under the condition that allows the healthy and failure classes to be linearly separable in the extracted feature space. On the contrary, [8] and [9] have demonstrated that circuit responses are rarely linearly classifiable (be it the no-fault or faulty condition). Instead, they are more optimally classified with non-linear Kernel-learning methods to identify faults.

In terms of accuracy, an MD-based classifier has been found to achieve 78% classification accuracy, as demonstrated by [10] on the Sallen-Key bandpass filter (BPF). However, for the same circuit and training data, the least-squares support vector machine (SVM) has been found to achieve the classification accuracy of 99% approximately, as demonstrated by [2]. These results show that a non-linear method is more suitable for classifying a healthy circuit from the one with parametric faults.

Based on the above, there are various questions that we have sought to address in this work: Is it computationally feasible to use machine learning methods for health estimation of complex FPGA architecture? How can hyperparameter selection problem be solved for accurate health estimate? Do N/PBTI degradation mechanisms represent a realistic account of FPGA ageing in terms of frequency degradation, threshold voltage and corresponding delay degradation?

Accordingly, we have developed a kernel learning technique to prognosticate FPGA health, taking into account the parametric shifts in its primitives, such as LUT. This involves treating FPGA health as a soft classification entity using a parameterized kernel function. As noise may accompany the extracted features, especially when the component tolerances are taken into account, it is viable to choose a regularization parameter to manage the relationship between training error and the complexity of decision function. This implies that the prognostics accuracy depends on the optimal choice of kernel and regularization parameters, collectively termed as 'hyperparameters.' In order to address this hyperparameter selection problem, we have also developed a stochastic filtering-based optimization method. This method helps fine-tune the Kernel and regularization parameters for a given FPGA.

6.2 Kernel Learning and FPGA Health Estimation/Prognostics – A Mathematical Interpretation

The mapping of data, extending from a Euclidean to a higher-dimensional space, and then fitting linear models into the projected space, forms the basis of Kernel-learning approach [11]. The projection of test data to a higher dimensional space and the calculation of similarity measures between the test data '$\boldsymbol{d_t}$' and the training data $'\{\boldsymbol{d_i}\}_{i=1}^{n}{}'$ for both the healthy and failure FPGA states facilitate the decision on test data.

Precisely, the function $\mathbf{K}(d_i, d_t)$: $\mathbb{R}^{n_d}$ **x** $\mathbb{R}^{n_d} \rightarrow \mathbb{R}$ is used to determine the similarity measure between the test and training features, $\boldsymbol{d_t}$ and $\boldsymbol{d_i}$, respectively, of length $\boldsymbol{n_d}$, along with a parameterized family of kernel functions. For instance, the automatic relevance determinant (ARD) Gaussian kernel function, which is represented as:

$$\boldsymbol{K}(di, dt\,) \; = \; exp\left(- \sum_{j=1}^{n_d} \frac{\| d_{i,j} \; - \; d_{t,j^2} \|}{\sigma_j} \right. \qquad (6-1)$$

is fundamentally parameterized by a kernel vector $\boldsymbol{\sigma} = [\boldsymbol{\sigma_1}, \boldsymbol{\sigma_2}, \ldots \boldsymbol{\sigma_{nd}}]$. This implies that if we consider the intermediate metric z as supporting the decision on the test data $\boldsymbol{d_t}$, it can then be represented mathematically as:

$$\boldsymbol{z} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{\alpha_i} \boldsymbol{K} (d_i , d_t) + b \qquad (\mathbf{6-2})$$

where, [$\alpha_1, \alpha_2, \alpha_3 \ldots.. \alpha_n$ b] are the estimated model parameters obtained by solving the system of linear equations [12]

$$\begin{bmatrix} \Omega + \frac{1}{\gamma} I & 1 \\ 1^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} \\ 0 \end{bmatrix} \qquad (6-3)$$

where, Υ represents the regularization parameter as $\boldsymbol{\alpha} = [\boldsymbol{\alpha_1, \alpha_2, \alpha_3 \ldots.. \alpha_n}]^T$, $\boldsymbol{C} = [c_1, c_2, c_3 \ldots. c_n]^T$ gives the class label related to the training data $'\{\boldsymbol{d_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{n}}{}'$,$\mathbf{1} = [1,1 \ldots.1]^T_{nx1}$, $\boldsymbol{I}$ represents the identity matrix of size $\boldsymbol{n}$ $\boldsymbol{x}$ $\boldsymbol{n}$, and $\boldsymbol{\Omega} = [\boldsymbol{\Omega_{ij}}] = [\boldsymbol{K} (\boldsymbol{d_i}, \boldsymbol{d_j})]$. This implies that the estimation of model parameters in (6-2) depends on $\boldsymbol{\Upsilon}$ and $\boldsymbol{\sigma}$, which are collectively termed as ‘hyperparameters,’ $\boldsymbol{h}$.

The problem of hyperparameter selection can be addressed by optimizing an error measure, for instance, a v-fold cross-validation error, on a hyperparameter value grid [13]. However, a grid search approach is limited and does not provide a wide coverage to the entire hyperparameter space. Moreover, in the case of a large number of features $\boldsymbol{n_d}$, it may get computationally expensive. For instance, the generalization error is estimated for $\mathbf{10}^{\boldsymbol{nd+1}}$ combinations of hyperparameter ($\boldsymbol{n_d}$ kernel parameters and one $\boldsymbol{\Upsilon}$) when a grid search of size **10** is used. This translates the generalization error estimate to $\mathbf{10^7}$ hyperparameter combinations, if a dataset with six features is considered.

The methods like ‘Gradient descent’ and ‘Evolutionary search’ for model selection and estimating hyperparameters have also been reported in the literature [14], [15], [16], [17]. Notwithstanding the effect of local minima problem on gradient descent (in particular) and evolutionary search, it is still desirable to include directional information as provided by these two methods in higher dimensional search spaces. However, an approach with faster convergence through

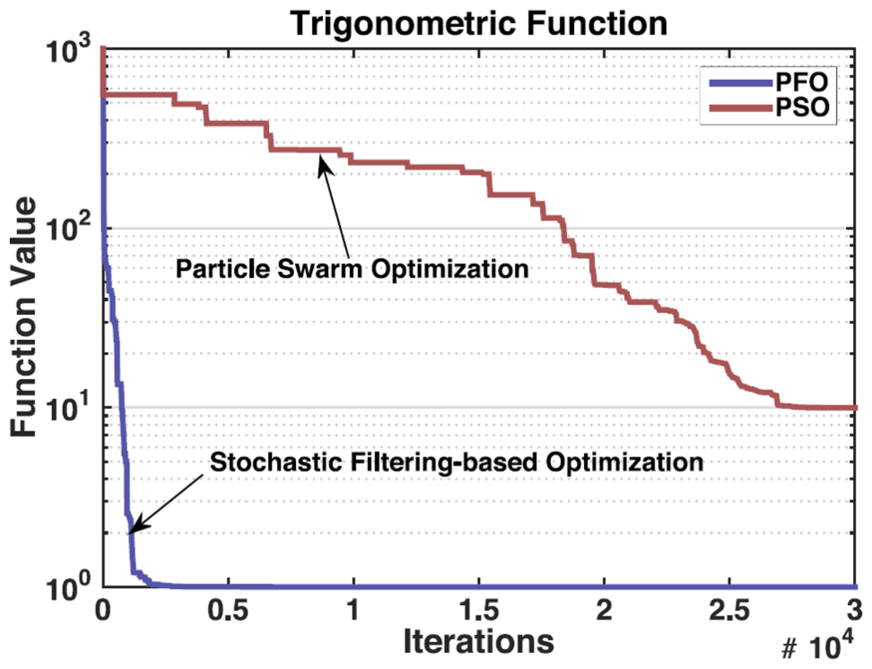


Figure 6-2 A comparison of Optimisation Method based on Stochastic Filtering with Particle Swarm Optimisation using a problem related to benchmark optimisation.

reformulation of the global optimization issue as a stochastic filtering problem can be considered. Using different benchmark optimization problems, the filtering-based optimization approach has been shown to perform more optimally than cross entropy (CE) and simulated annealing (SA) methods. On the other hand, the authors in [18] have weighted CE method higher, in terms of selecting the hyperparameters of an SVM classifier more accurately, over the particle swarm optimization (PSO) and grid search methods. Also, the stochastic filtering optimization method proves more efficient when compared with PSO, as shown in Figure 6-2.

Keeping in perspective the above, we have developed a method for FPGA health estimation that combines the positives of gradient descent with evolutionary search methods and provides an optimum solution to the hyperparameter selection problem as well.

6.3 The Developed Kernel Learning Method

The method developed for FPGA prognostics comprises both the learning and training phases. The learning phase involves building up of a fault dictionary for subsequent training of kernel-based learning algorithm. This is followed by the testing mode, where FPGA health is estimated by the continual comparison of extracted features with those stored in the fault dictionary by applying the kernel algorithm. Accelerated stress test results are used to help identify critical primitives (*holding the logic application under test*) and construct the fault dictionary. In our case, we take a look up table (LUT) as a critical primitive with an underlying CMOS circuitry that may experience parametric shifts and in turn, prevent any associated logic application from executing its intended function.

The characteristic behaviour of electronic circuits is represented in time and frequency responses. It is, therefore, a standard and recommended practice to excite the circuit with a test signal for feature extraction. With the identification of critical FPGA primitive as well as the fault/failure mechanism/mode, the fault dictionary is subsequently constructed on this vital information. The application (*circuit under test -CUT*) built using LUT primitive is then placed under a simulation environment to observe its hypothesized fault conditions by stressing it under a series of stress test conditions to extract features, as detailed in Chapter-4. Here, the fault condition represents a state where CUT fails to perform its intended function due to parametric shifts in its primitive and the underlying transistor beyond their predefined failure range, which in itself, is much higher than the actual tolerance limit. Signal processing techniques (such as wavelet transform) are applied to CUT behavioural responses for feature extraction [19], the collection of which lately forms the fault dictionary.

Mathematically, if we let $\boldsymbol{T} = \{\boldsymbol{d_i}, \boldsymbol{c_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{n}}$ represent the extracted features for training, where $\boldsymbol{n}$ is the total number of training feature vectors, $\boldsymbol{d_i}$ is the $\boldsymbol{i}$**th** feature vector, and $\boldsymbol{c_i}$ represents the label for which $\boldsymbol{c_i} = +1$ (*for* $\boldsymbol{d_i}$ *when the FPGA is healthy*) and $\boldsymbol{c_i} = \mathbf{-1}$ (*for* $\boldsymbol{d_i}$ *when the FPGA is faulty*), then the FPGA health (*application-based*) can be estimated (with prognosis) as a metric $\mathbf{HI} \in \mathbf{[0,1]}$ for a test input $\boldsymbol{d_t}$ given $\boldsymbol{T}$. So, for a given choice of $\boldsymbol{\Upsilon}$ and $\boldsymbol{\sigma}$, the optimal estimation of the model parameters in (6-2) can be made using (6-3) [12]. Similarly, the FPGA health index $\mathbf{HI}_t$, can be estimated at time t, considering the metric $\mathbf{HI}$ as the healthy class conditional probability, which means that d_t is extracted when the LUT is in a healthy state. The conditional probability with a positive label has been demonstrated by [20] to follow a logistic regression function. Based upon this, we can use the posterior class probability function and define the following relation:

$$\widehat{HI}t = P\,(c_t = +1|\, d_t) = g\,(z_t) = \frac{1}{1 + exp\,(Ez_t + V)}$$
$$= \mathcal{P}_t \qquad (6-4)$$

where, $\mathbf{E}$ and $\mathbf{V}$ are parameters estimated by employing Newton's backtracking method. As is evident from the above equation, $\mathbf{HI}$ depends on z , which resultantly depends on h. This further implies that the selection of h for a given T is essential to attain more accurate prognostics.

6.3.1 Employing Likelihood Function for Hyperparameter Selection

An objective function of the form $F+\lambda R$, is developed to solve the hyperparameter selection problem. F is dependent on the empirical loss, and R and λ represent the regularization term and parameter, respectively The authors in [14] have proposed the regularization term to be more of a negative logarithm of posterior probability than selecting priors on hyperparameters. We, therefore, constructed an objective function in this paper that extends posterior class probability function to a negative log-likelihood function.

So, if we Let p represent the health estimate for an LUT (*holding an application*), from which d is extracted, then p_i if $c_i = +1$ and $(1\text{-}p_i)$ if $c_i = \text{-}1$ will be the likelihood function $\mathcal{L}(*)$ for a feature vector d_i. Mathematically:

$$\mathcal{L}(d_i, c_i) = \mathcal{P}i^{\left(\frac{c_i + 1}{2}\right)}\,(1 - \mathcal{P}_i)^{\left(\frac{1 - c_i}{2}\right)} \qquad (6-5)$$

It is worth noting in the above equation, that $\mathcal{P}_i$ is the function of z_i (an intermediate metric) which depends on the model parameters a and b, as shown in (6-2). These model parameters, in turn, depend on h comprising Υ and σ, as shown in (6-3). It can be concluded that the likelihood function is essentially a function of $\boldsymbol{h}$. Also, the objective function is, characteristically, defined over cross-validated datasets that are pulled from the training dataset. This implies the cost function to a negative log-likelihood function over a cross-validation set $\tilde{S} = \{d_l, c_l\}_{l=1}^{L}$.

$$\mathcal{L}_{\tilde{S}}(\gamma, \sigma) = -\sum_{l=1}^{L}\left(\left(\frac{c_l + 1}{2}\right) log\,(\mathcal{P}_l) + \left(\frac{1 - c_l}{2}\right) log\,(1 - \mathcal{P}_l)\right) \quad (6-6)$$

where, $\mathcal{P}_l = \frac{1}{1 + exp\,(Ez_l + V)}$ and $z_l = \sum_{i=1}^{n} \alpha_i\, k\,(d_i, d_l) + b$. We, therefore, focused on minimizing the *v*-fold cross validation log likelihood as follows:

$$\mathcal{L}_S\,(\gamma, \sigma) = (V^{-1}) \sum_{v=1}^{V} \mathcal{L}_{\tilde{S}_v}\,(\gamma\,, \sigma) \qquad (6-7)$$

where, $\boldsymbol{S = \tilde{S}_1\ U\ \tilde{S}_1\ U \ldots\tilde{S}_V}$ represents the partition of the training dataset into $\boldsymbol{V}$ disjoint subsets and $\boldsymbol{\mathcal{L}_{\tilde{S}}(\gamma, \sigma)}$ is the objective function given the holdout set $\boldsymbol{Sv}$.

6.3.2 Optimization Method for Hyperparameter Selection

It is imperative to identify the hyperparameter values to reduce (6-7). In such a case, the optimization problem can be expressed as follows:

$$\boldsymbol{h}^* = arg\,min_{h\,\in\,\mathcal{H}}\mathcal{L}_S\,(\boldsymbol{h}) \qquad (6\text{-}8)$$

where, $\boldsymbol{\mathcal{L}_S}\,(\boldsymbol{h})$ denotes the likelihood function in (6-7) over the dataset $\boldsymbol{S}$ and $\boldsymbol{\mathcal{H}}$ represents the solution space for $\boldsymbol{h}$. We assume that $\boldsymbol{\mathcal{L}_S}\,(\boldsymbol{h})$constitutes a unique global optimal solution $\boldsymbol{h}^*$. Also, as mentioned in section-II, we solve the optimization of hyperparameter by reconstructing it as a stochastic filtering issue. The primary goal of stochastic filtering is to make an accurate estimate of the unobserved condition of a dynamic system by observing a sequential stream of noises that accompanies it. The unobserved condition corresponds to $\boldsymbol{h}$ and as the system evolves toward $\boldsymbol{h}^*$, the conditional distribution of the unobserved condition approaches a delta function concentrated on the optimal solution. Accordingly, the optimal solution is searched through the sequential estimation of the conditional density. Here, it is important to afford some sort of approximation to facilitate stochastic filtering implementation. As particle filter (PF) is a commonly used sequential Monte-Carlo technique that does not impose any constraint on the condition's distribution, it is considered viable to employ PF for global optimization to address the problem of model selection.

An appropriate state-space model is constructed to transform the optimization problem into a filtering problem as:

$$\boldsymbol{h}_k = \boldsymbol{h}_{k-1} - \varepsilon \nabla \mathcal{L}(\boldsymbol{h}_{k-1}), \qquad \mathrm{k} = 1{,}2{,}\ldots\ldots \qquad (6\text{-}9)$$

$$e_{\kappa} = \mathcal{L}(\boldsymbol{h}_k) - \upsilon_k \qquad (6-10)$$

where, $\boldsymbol{h}_k$ represents the unobserved state to be estimated and $\boldsymbol{e_k}$ denotes the observation with noise $\boldsymbol{\upsilon_k}$. $\nabla\mathcal{L}(\boldsymbol{h}_k)$ in (6-9) is the gradient of $\mathcal{L}_{\tilde{S}}(\boldsymbol{h})$ with respect to $\boldsymbol{h}_k$ for a considered holdout set. By virtue of $\mathcal{L}(\boldsymbol{h}_k)$ being a log-likelihood function, it is differentiable with respect to $\boldsymbol{\Upsilon}$ and $\boldsymbol{\sigma}$, as and when the kernel function is differentiated. Therefore, $\nabla\mathcal{L}(\boldsymbol{h}_k)$ can be determined using the following linear equations with ARD kernel function:

$$\frac{\partial \mathcal{L}_S}{\partial \gamma} = \sum_{l=1}^{L} \frac{\partial \mathcal{L}_S}{\partial \mathcal{P}_l} \left[\frac{- Aexp\,[A\psi^T\,(d_l)\,\beta]}{\mathcal{P}_l^2} \right] \psi^T\,(d_l)\,\boldsymbol{\beta} \qquad . \qquad (\mathbf{6-11})$$

$$\frac{\partial \mathcal{L}_S}{\partial \sigma_i} = \sum_{l=1}^{L} \frac{\partial \mathcal{L}_S}{\partial \mathcal{P}_l} \left[\frac{-Aexp[A\psi^T(d_l)\beta]}{\mathcal{P}_l^2} \right]$$

$$x\ \{\, \psi^T\,(d_l)\,\dot{\boldsymbol{\beta}} \; + \; \dot{\psi}^T\,(d_l)\,\boldsymbol{\beta}\,\} \qquad (\mathbf{6-12})$$

where, $\mathcal{P}_l$=$(1+exp[A\psi^T(d_l)\beta \; + \; V])^{-1}, \psi^T\,(d_l)[k\,(\,d_1, d_l)k\,(\,d_2, d_l)\ldots k\,(\,d_n, d_l)\;\;1]$, and $\boldsymbol{\beta} \; = \; [\alpha_1 \;\; \alpha_2 \;\; \ldots\ldots \alpha_n \quad b]^T$. In order to determine $\dot{\boldsymbol{\beta}}$, we can use the relation $\dot{\boldsymbol{\beta}} \; = \; -\boldsymbol{P}^{-1}\,\dot{\boldsymbol{P}}\,\boldsymbol{\beta}$, with P = $\begin{bmatrix} \boldsymbol{\Omega} + \frac{1}{\gamma}\mathbf{I} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}$.

The application of PF for hyperparameter selection is shown in Figure 6-3. Here, $\boldsymbol{\mathcal{H}}$ is assumed to be one dimensional (1-D) for illustration purposes. To begin with, the distribution ***b*** follows $\boldsymbol{\mathcal{H}}$ as illustrated in Figure 6-3(a). Subsequent to this, random sampling of ***b*** is carried out in an independent and identically distributed (**i.i.d**) manner, with N being the sampling size. We updated the hyperparameter vectors based on their gradients $\nabla\mathcal{L}(\boldsymbol{h}_k^j); j = 1{:}N$. Accordingly, the related generalization error $\mathcal{L}_S(\boldsymbol{h}_k^j)$ is calculated (as shown in Figure 6-3(b)). Subsequent to this, the hyperparameters with least generalization error (a.k.a. elite particles) are chosen as **$(1\text{-}\rho)$**-quantile of the complete generalization error (as shown in Figure 6-3(c)). This is followed by the updation of the distribution $\boldsymbol{b}$ in line with the elite particle locations (Figure 6-3(d)). As the distribution $\boldsymbol{b}$ is characterized by particles and associated weights, different shapes for $\boldsymbol{b}$ can be realized without building a parametric model. The aforementioned steps are repeated to a point where the distribution $\boldsymbol{b}$ approaches delta function and the global optimum $\boldsymbol{h}^*$ is identifiable.

This method is iterative and employs only a set of N hyperparameter combinations per iteration. Accordingly, the generalization error must be estimated for $N \, x \, I$ hyperparameter combinations, where ***I*** denotes the number of

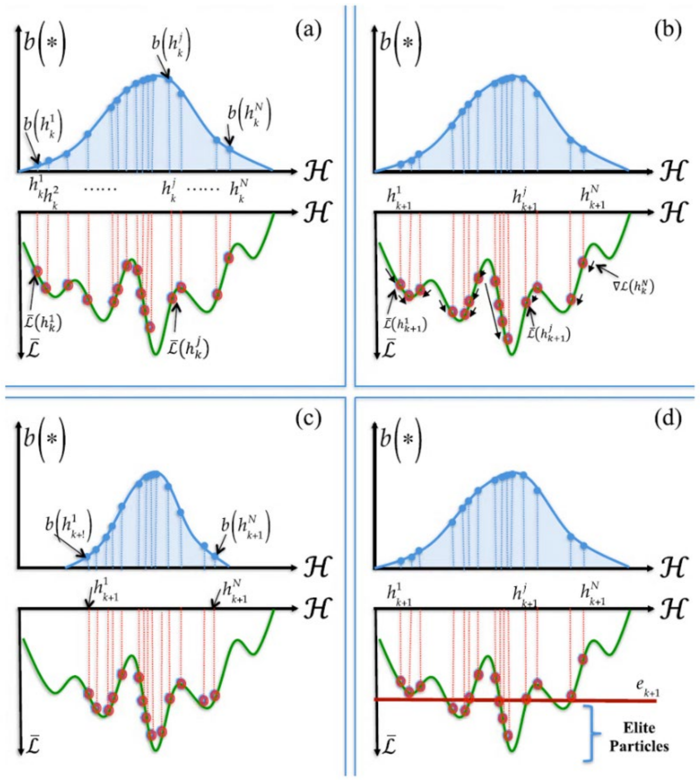

(a)
(b)
(c)
(d)
$b(*)$
$b(h_k^j)$
$b(h_k^1)$
$b(h_k^N)$
$\mathcal{H}$
$h_k^1 h_k^2 \;\ldots\ldots\; h_k^j \;\ldots\ldots\; h_k^N$
$\bar{\mathcal{L}}(h_k^1)$
$\bar{\mathcal{L}}(h_k^j)$
$\bar{\mathcal{L}}$
h_{k+1}^1
h_{k+1}^j
h_{k+1}^N
$\nabla\mathcal{L}(h_k^N)$
$\bar{\mathcal{L}}(h_{k+1}^1)$
$\bar{\mathcal{L}}(h_{k+1}^j)$
$b(h_{k+1}^1)$
$b(h_{k+1}^N)$
e_{k+1}
Elite Particles

Figure 6-3 Optimization of hyperparameters using Particle Filtering Approach

iterations. It shows that the computational complexity of the stochastic filtering method does not scale to the extent as the grid search does, going up to $\mathbf{10}^{nd+1}$. On the contrary, the stochastic filtering method incorporates gradient descent with each hyperparameter combination h^j and resultantly becomes computationally more expensive when compared to the gradient descent.

6.4 Implementation Results and Analysis – Simulation and Experiments

Several experimental tests were conducted on a pair of 28 nm FPGA technology, to demonstrate the viability of this Kernel learning method. Typically, a modern FPGA (including other VLSI circuits) is presumed to operate unhindered for several years without any significant deterioration. However, under the harsh operational environment with highly accelerated temperatures and elevated voltages (in our case, due to the impact of the presence and malicious activity of the hardware Trojans), the ageing process sets in much earlier. This provides a perfect stage for measuring physical degradation in an FPGA and assessing the impact on its primary soft-logic resources. We used a frequency/delay measurement method based on threshold-voltage shifts in PFETs transistors due to NBTI triggered hardware Trojans (Chapter 5). It enables the measurement of frequency/delay degradation in fundamental FPGA primitives such as registers, look-up-tables, and metal interconnects. Moreover, the FPGA application circuitries (CUT) were constructed from typical logic paths, representing real-time circuit designs. These include NAND2 and True Single-Phase Clock (TSPC) based flip-flop circuits integrated with hardware Trojan circuitry (*as NBTI accelerators*) in LUTs. A dual output Ring Oscillator-based sensor segments were wrapped around LUTs to sense and give a measure of frequency/delay degradation, as explained in Chapter-5.

The FPGA was contemplated healthy when all of its primitives and the underlying transistors varied within the designed parametric tolerance ranges of threshold voltage (V_{th}), drain current (I_{dd}), frequency (f), and corresponding delays. Mathematically, $(1- \mathrm{Tr})\, A_n < \mathrm{A} < (1+\mathrm{Tr})\, A_n$, where T represents the tolerance range, A is the actual real value, and A_n is the nominal value of the primitive. We can, therefore, imply that if any primitive parameter varies beyond its tolerance limit, i.e., $\mathbf{A < (1\text{-}Tr)}\, A_n$ **or** $\mathbf{A > (1\text{+}Tr)}\, A_n$, the primitive is considered having some parametric fault. However, it must be noted that the presence of any parametric fault does not mean that the FPGA had failed. The FPGA is evaluated as 'failed' only when the parametric variation beyond the primitive tolerance range leads to the FPGA failing to execute its intended function.

6.4.1 Accelerated Life Tests on FPGA

6.4.1.1 Prologue to Degradation Mechanisms

This section of the chapter is primarily meant to give a concise overview of the degradation mechanisms that affect VLSI devices and how we based the acquisition of the FPGA's health profile on one of the mechanisms - mainly when the FPGA is under the influence of malicious activity, such as the hardware Trojan.

Mainly, four degradation mechanisms impact the reliability and performance of an FPGA and FPGA-based applications. These mechanisms include Hot Carrier Injection (HCI), Electromigration, Time-Dependent Dielectric Breakdown (TDDB), and N/PBTI (Negative/Positive Bias Temperature Instability), as explained in previous chapters. The focus of our work is, however, on the frequency/delay degradation an NBTI mechanism can exert on FPGA primitives and its underlying PFETs when a threshold-voltage triggered hardware Trojan is activated under highly accelerated stress conditions. Accordingly, the accelerated test methodology was devised to acquire critical parametric features (threshold voltage **(V_{th}),** drain current **(I_{dd}),** frequency **(f),** and signal propagation time delays **(t_{pd})** of the FPGA primitives and PFETs to build a comprehensive fault dictionary for the health estimation of an FPGA. The details of the same are given in Chapters -4 and 5.

6.4.1.2 Accelerated Test Methodology

The stress test conditions (STCs) were chosen to achieve results within a practical time-frame and, at the same time, induce degradation consistent with the regular use of the device. High levels of voltage and temperature were applied to attain a combined acceleration in ageing of the FPGA. The voltage dependence is given by the relation $t_f \propto V_{gs}^{-\Upsilon}$, where t_f represents the time for threshold voltage shift V_{th} to reach a nominal failure limit and Υ is the constant that lies between 6-8. We confirmed through experimentation that the target FPGA remains stable under a 1V increase in its core supply voltage. Resultantly, we increased the core power supply from 1.2V to 1.6V.

On the other hand, high-temperature influences the NBTI- best explained by Arrhenius law as $\boldsymbol{t_f \propto \exp\left(\frac{E_a}{kT}\right)}$**.** In our accelerated test, we used the on-chip heaters (*part of the hardware Trojan circuitry implemented in FPGA to induce accelerated device ageing* - shown in Figures 5-9(a) and (b), Chapter -5) to heat the FPGA to varying junction temperatures up to 110 - 125ºC.

6.4.1.3 Stress Test Conditions

The stress tests were conducted on a pair of 28 nm technology FPGA under the following stress test conditions:

- **STC-1**: Stress Temp. - 110ºC, Stress Duration - 27 hours, Stress Mode - AC (1.6V), and Ring Oscillator is always enabled to switch.
- **STC-2**: Stress Temp. - 110ºC, Stress Duration - 27 hours, Stress Mode – DC (1.6V), and Ring Oscillator is always enabled every 15 minutes to record the data. Data sampling is maintained at < 3s.
- **STC-3**: Same as STC-2. However, the Stress Temp is kept at 80ºC - safe operating limit for 28 nm technology FPGA under test.

6.4.2 LUT Based Combinatorial Circuit with Hardware Trojan

The underlying schematic of a combinatorial circuit, NAND2 with hardware Trojan, is shown in Figure 5-6. The magnitude of degradation observed was significant in all stress test conditions when the Trojan PFET and NAND2 P/NFETs were stressed. No hard faults were detected during this stress period, meaning the transistors, P/NFETs (*the critical components*), continued working within the pre-defined tolerance ranges of parameters including threshold voltage shift (**10%**), drain current **(-5%),** and the resulting RO-based sensor segments' frequency **(-5%).** This tolerance range included the worst-case, nominal, and best-case effects of process variations (varying oxide thickness and transistor length). However, we observed the circuit failure as the hardware Trojan gets activated just over 20% of the shift in V_{th} and the corresponding -15% shift in I_{DD}. It was, therefore, essential to define a safe operating limit for this LUT based combinatorial circuit in the presence of a hardware Trojan. Based on several experimentation cycles, we defined the circuit failure condition to be a 10% decrease in the RO-based sensor segments' frequency (set at 25 MHz) as the parameters of critical PFETs move beyond the tolerance ranges mentioned above.

The failure condition, at this point, is assigned to assess and evaluate the performance and efficiency of the developed Kernel health estimation method. We, therefore, started with the off-line learning phase. Several parametric faults were induced within the P/NFETs by varying the afore-mentioned parameters' ranges to determine the failure threshold in line with the defined failure condition of 20% frequency/delay degradation. Table 6-1 shows the critical P/NFETs parametric tolerances and failure threshold for the LUT based combinatorial circuit, NAND2, under the influence of hardware Trojan.

The parametric responses of the combinatorial and the FRED sensor segments are closely evaluated to extract various statistical and frequency features. Accordingly, a digital wavelet packet transform with Haar mother wavelet (*optimal diagnostic accuracy*) is used to refer the frequency features to the energy held in the detail coefficients up to four levels of decomposition of the LUT's response.

Table 6-1 Critical Components of LUT-based Combinatorial Circuit with Nominal, Tolerance and Threshold Values.

Components	Nominal Value			Tolerance (%)	Failure Threshold (%)
	Threshold Voltage (V_{TH})	Drain Current (I_{DD})	Frequency (MHz)	V_{TH} | I_{DD}	V_{TH} | I_{DD}
PFET – MT	0.45 V (450 mV)	25 µA		10 | -5	20 | - 15
PFET – M1	0.43 V (430 mV)	26 µA		10 | -5	20 | - 15
PFET – M2	0.43 V (430 mV)	26 µA		10 | -5	20 | - 15
NFET – M3	0.42 V (420 mV)	27 µA		10 | -5	35 | - 20
NFET – M4	0.42 V (420 mV)	27 µA		10 | -5	35 | - 20
FRED Sensor Segments (RO-based)			25	- 5	- 10

So, if $\boldsymbol{d_c}$ is the detail coefficient, then the energy held within it $\boldsymbol{E_j}$ can be represented at the $\boldsymbol{jth}$ level (in our case, we set $\boldsymbol{J} = \mathbf{4}$) of decomposition as:

$$E_j \;=\; \sum_k |d_{j,k}\;|^2\,, \quad j \;=\; 1 \,:\, J \qquad (6-13)$$

The statistical features, on the other hand, comprised the entropy and kurtosis elements of the LUT based circuit responses. We simulated 400 no-fault and 400 fault cases during the off-line testing of the FPGA primitives (*LUT in our case*). The combinatorial circuit, along with the ring oscillator in LUT, was subjected to stress tests, and features were extracted accordingly. The kernel-based health estimator was then trained using the extracted features and their class labels in

Kernel Algorithm: Hyperparameter Selection Using Particle Filtering	
Input:	Set of 'Training Features' from Fault Dictionary
	$\boldsymbol{T} = \{d_i, c_i\}_{i=1}^{n}$
Output:	Optimised Hyperparameters: $\boldsymbol{h}^* \in \boldsymbol{\mathcal{H}}$
1.	**Initialisation Phase**: Specify $\rho \in [0,1]$ and an initial probability density function b_0, defined on $\mathcal{H}$. Sample $\{h_1^j\}_{j=1}^{N}$ i.i.d. from b_0 and set $k = 1$
2.	**Observation Formulation Phase:** Let e_k be the sample $(1 - \rho)$-quantile of $\{\boldsymbol{\mathcal{L}}(\boldsymbol{h}_k^j)\}_{j=1}^{N}$. *If* $\boldsymbol{k} > 1$ and $e_k < e_{k-1}$, then set $e_k = e_{k-1}$.
3.	**State-Update Phase:** Update the particle locations in the hyperparameter space in accordance with the system dynamic model. $\boldsymbol{h}_k \;=\; \boldsymbol{h}_{k-1} \;-\; \varepsilon\, \nabla\mathcal{L}\,(\boldsymbol{h}_{k-1}), where\ k \;=\; 1,2,3\ldots..$
4.	**Bayesian Update Phase:** $b_k\,(\boldsymbol{h}_k) = \sum_{j=1}^{N} w_k^j\;\; \delta(\boldsymbol{h}_k \;-\; \boldsymbol{h}_k^j)$ then calculate weights according to: $w_k^j \;\propto\; \varphi\,(\mathcal{L}(\boldsymbol{h}_k^j) \;-\; e_k)$ and normalise.
5.	**Resampling Phase:** Formulate a continuous approximation from $b_k\,(\boldsymbol{h}_k)$ and undertake i.i.d sampling to achieve $\{(\boldsymbol{h}_{k+1}^j)\}_{j=1}^{N}$
Termination Criterion	In case the standard deviation $b_k\,(\boldsymbol{h}_k) < w$, then terminate; else, $k \;\longleftarrow\; k \;+ 1$ and proceed to Phase -2.

Figure 6-4 Kernel-based FPGA Health Estimation Algorithm.

accordance with the Algorithm given in Figure 6-4. With the particle size of the hyperparameter set at $N = 40$, the search was executed in a $log(h)$ plane from [-20, +20] (*defined failure threshold*) with the elements of hyperparameter taking the values from 10^{-4} to 10^{+4} ($J=4$).

6.4.2.1 Health Estimation Metrics

The P/NFETs and RO-based Sensor Segments' frequency degradation trends obtained from highly accelerated thermal and power stress tests were assessed for four degradation pathways to evaluate the FPGA health estimation method. In the process, the following metrics were defined:

- T_{af} : Actual circuit failure time.
- T_f : Failure time estimated from HI_t (i.e., the time when Ideal Health, $HI_t < 0.05$).
- T_{pf} : Time at which parametric fault alarm was raised (i.e., the time when $HI_t < 0.95$).
- F_s : Fault severity at T_f.
- F_{pf} : Fault severity at T_{pf}.

Table 6-2 and Figure 6-5 give details of the health estimates obtained from the developed kernel method. They are indicated in blue. The figures also include a comparison with MD and ED-based methods, as given in [20] and [21], respectively. Moreover, HI^a_t the ideal health of the LUT based circuit is also

Table 6-2 Performance Analytics of the Developed Health Estimation/Prognostics Method for FPGA – LUT Primitive (NAND2 Application)

Component	Tolerance (%) V_{TH} | I_{DD}	Failure Threshold (%) V_{TH} | I_{DD}	T_f (hrs)	T_{pf} (hrs)	F_{pf} (%)	F_s (%)	T_{af} (hrs)
PFET – MT	10 | -5	20 | - 15	20	13	7.5	20.45	40
PFET – M1	10 | -5	20 | - 15	23	15	8.25	21.50	35
PFET – M2	10 | -5	20 | - 15	23	18	8.50	20.75	40
NFET – M3	10 | -5	35 | - 20	25	20	10.57	32.20	45
NFET – M4	10 | -5	35 | - 20	25	19	9.78	35.67	40
FRED Sensor Segments (RO-based)	- 5	- 10	22	15	8.79	19.65	40

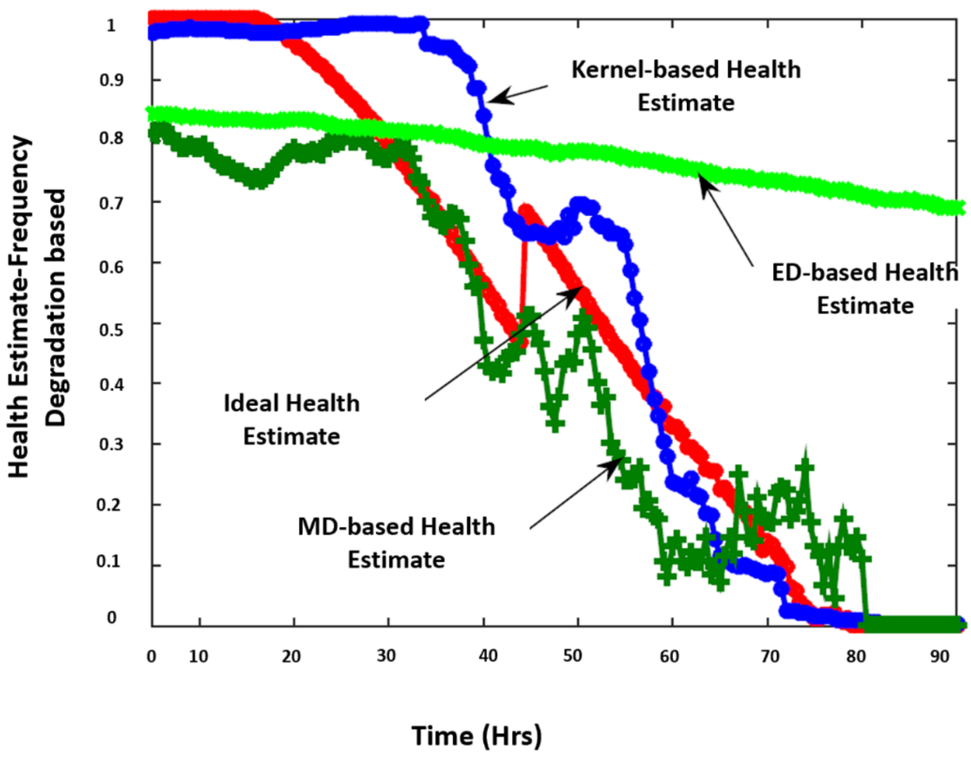


Figure 6-5 Results of FPGA Health Estimation for Parametric Deviation (Frequency Degradation of RO-based Sensor Segments) in Combinatorial Circuit (NAND2)

shown to verify the capability of the kernel method to indicate the increase in the fault intensity. HI^{a}_{t}, the ideal health, reflects the parameters of FPGA primitives and circuit components within the safe operating limits or, in other words, within the designed tolerances.

6.4.2.2 Analysis

It is evident from Figure 6-5 and Table 6-2 that the kernel method is efficient in identifying degradation in the FPGA health (*based on its primitives and underlying transistors parametric behaviour*) as the intensity of faults in its critical components increases. On the other hand, the MD-based method tracks the degradation in LUT's NAND2 circuitry; however, it does not follow degradation in the FRED sensor segments wrapped around the LUT. In other words, this method does not generate health estimates that follow the ideal health trend. This is probably due to the similarities in the frequency degradation of the LUT's transfer function with faults in the RO-based sensor segments when it is compared to the healthy LUT.

The ED-based method, as shown in Figure 6-5, gives a good account of degradation in FPGA health with the slowing down of the faulty PFETs. However, it does not compare well with kernel and MD-based methods when it comes to indicating the intensity of health degradation. This is because the ED-based estimator is not capable of taking into account the P/NFETs and RO-based sensor segments' tolerances. As a result, the ED estimator assumes the healthy and failure classes as linearly separable in the extracted feature space. In other words, it implies that the ED method would give a perfect degradation curve if the circuit components are kept fixed to their nominal values, whereas the faulty component is changed gradually. Comparatively, the kernel-based estimator does not assume the healthy and failure classes as linearly separable in extracted features' space.

6.4.3 LUT Based Sequential Circuit with Hardware Trojan

The sequential circuit comprises true single-phase clock (TSPC) based flip flop, connected with a single PFET (MT) as a hardware Trojan. Figure 5-8 gives the

Table 6-3 Critical Components of LUT-based Sequential Circuit with Nominal, Tolerance and Threshold Values.

Components	Nominal Value			Tolerance (%)	Failure Threshold (%)
	Threshold Voltage (V_{TH})	Drain Current (I_{DD})	Frequency (MHz)	V_{TH} | I_{DD}	V_{TH} | I_{DD}
PFET – MT	0.45 V (450 mV)	25 µA		10 | -5	20 | - 15
PFET – M1	0.43 V (430 mV)	26 µA		10 | -5	20 | - 15
PFET – M2	0.43 V (430 mV)	26 µA		10 | -5	20 | - 15
PFET – M4	0.45 V (420 mV)	25 µA		10 | -5	20 | - 15
PFET – M7	0.42 V (420 mV)	27 µA		10 | -5	20 | - 15
PFET – M10	0.46 V (460 mV)	24.8 µA		10 | -5	20 | - 15
NFET – M3	0.45 V (450 mV)	25 µA		10 | -5	35 | - 20
NFET – M5	0.44 V (440 mV)	25.5 µA		10 | -5	35 | - 20
NFET – M6	0.46 V (460 mV)	24.8 µA		10 | -5	35 | - 20
NFET – M8	0.46 V (460 mV)	24.8 µA		10 | -5	35 | - 20
NFET – M9	0.46 V (460 mV)	24.8 µA		10 | -5	35 | - 20
NFET – M11	0.46 V (460 mV)	24.8 µA		10 | -5	35 | - 20
FRED Sensor Segments (RO-based)			25	- 5	- 10

schematic of the circuit, designed and implemented over a chain of 5 LUTs. In addition, it also consists of a ring oscillator wrapped around the LUT chain. The criticial components of this LUT based sequential circuit are the PFETs in the hardware Trojan circuit, NFETs in the flip flop circuit, and the ring oscillator with a maximum operating frequency of 25 MHz. Any shift in the threshold voltage of P/NFETs results in the frequency degradation of the ring oscillator (FRED Sensor Segments) and we observe the corresponding delay degradation in the LUT based sequential circuit. Here, we use the percent shift in the threshold voltage *'%ΔV_{th}'* of P/NFETs as the precursor parameter to predict the circuit failure. Table 6-3 shows the critical P/NFETs parametric tolerances and failure threshold for the LUT based combinatorial circuit, NAND2, under the influence of hardware Trojan.

Threshold voltage meter (Chapter 5 – Section 5.4.3) was used to capture the increase in threshold voltage under the application of accelerated life tests. The results obtained are given in Table 6-4.

Similar to the combinatorial circuit, the threshold voltage, frequency, and statistical features were extracted from the sequential circuit response by employing the digital wavelet packet transform with Haar mother wavelet. Both

Table 6-4 Performance Analytics of the Developed Health Estimation/Prognostics Method for FPGA – LUT Primitive (TSPC-FF Application)

Component	Tolerance (%) V_{TH} | I_{DD}	Failure Threshold (%) V_{TH} | I_{DD}	T_f (hrs)	T_{pf} (hrs)	F_{pf} (%)	F_s (%)	T_{af} (hrs)
PFET – MT	10 | -5	20 | - 15	20	13	7.5	20.45	34
PFET – M1	10 | -5	20 | - 15	23	17	8.25	21.50	35
PFET – M2	10 | -5	20 | - 15	25	16	8.50	20.75	40
PFET – M4	10 | -5	20 | - 15	25	20	10.57	22.20	38
PFET – M7	10 | -5	20 | - 15	26	19	9.78	20.67	35
PFET – M10	10 | -5	20 | - 15	23	17	8.5	21.45	33
NFET – M3	10 | -5	35 | - 20	20	15	11.25	36.50	45
NFET – M5	10 | -5	35 | - 20	23	16	10.50	32.75	39
NFET – M6	10 | -5	35 | - 20	22	15	10.23	33.10	40
NFET – M8	10 | -5	35 | - 20	25	17	9.18	31.67	39
NFET – M9	10 | -5	35 | - 20	27	16	11.55	32.20	38
NFET – M11	10 | -5	35 | - 20	26	17	10.78	36.67	40
FRED Sensor Segments (RO-based)	-5	- 10	21	15	10.79	11.65	35

the threshold voltage and frequency features comprised the energy held in detail as well as approximate coefficients up to four levels of decomposition with discrete wavelet transformation. Statistical features, on the other hand, included the entropy and kurtosis of the LUT based circuit response to different stress test conditions. In all, 400 no-fault and hardware Trojan induced fault cases were simulated with four different degradation trends (***increase in threshold voltage, decrease in drain current, slowing down of Sensor Segments' frequency, and increase in delays***).

6.4.3.1 Analysis

The resulting LUT circuit health estimation using the kernel method for one of the cases (PFET-MT) is shown in Figure 6-6. As is evident from the figure, the ideal health $'HI_t^A'$ degradation curve (Brown) variation implies a gradual degradation in the I_{DD} of PFET -MT (Trojan Transistor) in LUT and does not approach the failure threshold of -15% till it reaches 15 hours (approx.) of testing. Whereas, the kernel method showed some variations in FPGA health estimation (*based on LUT*

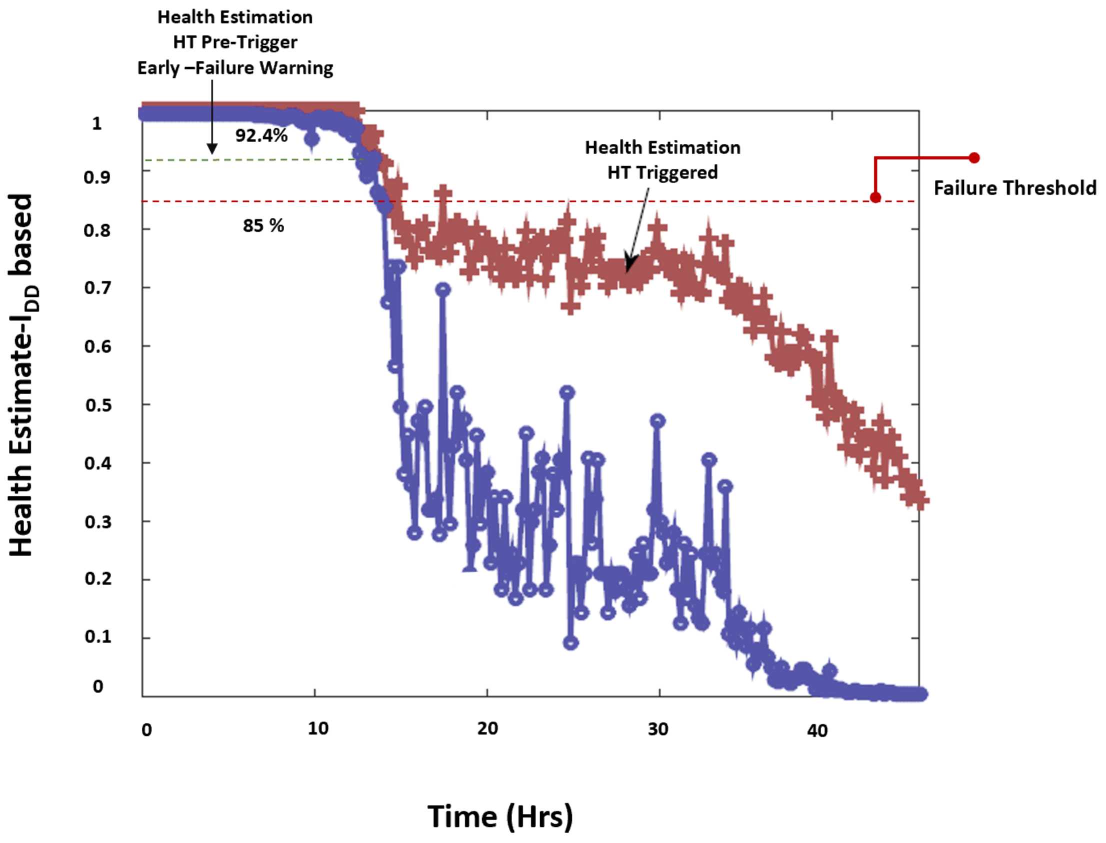


Figure 6-6 FPGA Health Estimate with Early-Warning Indication using Kernel Method in a Sequential Circuit configuration

primitive) with detection of health degradation (with HT-triggered) earlier than exhibited by the ideal health trend. In addition to this, estimates of the failure time of the circuit were provided by the kernel-based method 02 hrs before (at the health estimate of 92.4%) the critical component reached its failure threshold with health estimate at 85%. We investigated it and concluded that such variation could have very likely resulted from the un-stressed PFET transistors in the flip flop that operated within their tolerance ranges.

In this case of the hardware Trojan circuit, as the threshold voltage increases with high negative bias and elevated temperature in PFETs, the drain current I_{dd} decreases causing slowing down of the flip flop circuit. The threshold voltage response generated by PFETs under different stress conditions was directly fed to the health estimator. Table 6-4 gives a detailed account of the performance results of validation on all the three circuits of the sequential circuit in LUT.

As is evident from Table 6-4, the kernel health estimator was able to detect and identify the instant at which the sequential circuit developed a parametric fault (***PFET - MT*** ΔV_{th} *< 20% and* ΔI_{dd} *< -15%).* This was also true for the hardware Trojan part of the sequential circuit in LUT. The kernel estimator was equally proficient in detecting the actual failure time for the RO-based Sensor circuit. More importantly, in the case of the flip flop circuit of the sequential logic, when $T_f < T_{af}$, the health estimator raised early failure warning before the circuit failure. This is a desirable feature not only from the health prediction viewpoint but from the security perspective as well. Primarily when FPGA primitives are implemented with malicious circuits like hardware Trojans, the efficient detection and classification is the most sought-after property of an effective algorithm.

6.4.4 Viability of Kernel-Based Health Estimation for FPGAs

This kernel-based method for estimating FPGA health can be applied in both the offline and online settings to accrue its maximum benefits and optimize its usability for other VLSI devices as well. The offline implementation of this method is relatively straightforward. A PC system integrated with automated test equipment (ATE) is used to hold and process the kernel-based algorithm along with the training data. The on-chip or in-circuit measurements are made using different EDA tools like Vivado, Cadence Virtuoso, and HSpice simulation tools. The online health estimation involves an offline training of the kernel-based algorithm, whereby its results, including the estimated hyperparameters (Υ and σ) and the model parameters (α and b), are stored in the processor to conduct health estimation in real-time.

The computation time, an important performance parameter, was also investigated for the combinatorial and sequential circuits to ascertain its applicability for in-field operations. These two circuit configurations within LUT primitive provided a reasonably large number of fault classes as well as the size of the training data. In a nutshell, all the processing steps including signal denoising, feature extraction, and the FPGA health estimation using kernel algorithm, took precisely 1.5 and 2.2 ms, respectively, in MATLAB 2019b environment running on a 2.86 GHz Intel Core i7 processor with 32-Gb DDR4

RAM. It took the whole onboard computation to complete within 3.8 ms approximately. This implies that if we have more critical circuits/applications with high loading in an FPGA, the developed kernel method would still be computationally efficient for health estimation.

6.4.4.1 Classification/Health Estimation/Prognostics Accuracy

As mentioned above in Sections 4.4.2 and 4.4.3, a total of 400 cases each for no-fault FPGA and hardware-Trojan infected FPGA, with a combinatorial and sequential circuits, were simulated and experimented under the stress test conditions to create a fault dictionary, an excerpt of the same is shown in Table 6-5. The Kernel-based classifier was able to classify the two categories with accuracies as mentioned below:

- Combinatorial circuit – non-faulty FPGA: 375/400 – **93.75%**
- Combinatorial circuit – faulty FPGA: 387/400 – **96.75%**
- Sequential circuit – non-faulty: 395/400 – **98.75 %**
- Sequential circuit – faulty FPGA: 390/400 – **97.5%**

6.4.4.2 FPGA Time to Failure/RUL

Based upon the instantaneous health estimate, for instance in the case of LUT with Combinatorial circuit application and hardware Trojan implemented, the

Table 6-5 A Simplistic Excerpt of FPGA Fault Dictionary – (LUT- TSPC FF)

Fault Code	Operational Mode	Fault Code Description
F0	Normal	$V_{TH0} = 0.45V$ | Tj = 24ºC | AC = 1.0V
PFET – MT	$V_{THt}\uparrow$ | $I_{DDt}\downarrow$	**AC and DC** 1.2V ~ 1.6 V **Junction Temp. Tj** 60ºC ~ 110ºC **Duration** 0 hrs ~ 50 hrs
PFET – M1	$V_{TH1}\uparrow$ | $I_{DD1}\downarrow$	
PFET – M2	$V_{TH2}\uparrow$ | $I_{DD2}\downarrow$	
PFET – M4	$V_{TH3}\uparrow$ | $I_{DD3}\downarrow$	
PFET – M7	$V_{TH4}\uparrow$ | $I_{DD4}\downarrow$	
PFET – M10	$V_{TH5}\uparrow$ | $I_{DD5}\downarrow$	
NFET – M3	$V_{TH6}\uparrow$ | $I_{DD6}\downarrow$	
NFET – M5	$V_{TH7}\uparrow$ | $I_{DD7}\downarrow$	
NFET – M6	$V_{TH8}\uparrow$ | $I_{DD8}\downarrow$	
NFET – M8	$V_{TH9}\uparrow$ | $I_{DD9}\downarrow$	
NFET – M9	$V_{TH10}\uparrow$ | $I_{DD10}\downarrow$	
NFET – M11	$V_{TH11}\uparrow$ | $I_{DD11}\downarrow$	

Time to Failure (TtF)/Remaining Useful Life (RUL) can be calculated by subtracting the time at 'prediction made' from the time when the value of threshold voltage/frequency shift equals the defined failure threshold.

6.5 Summary

The reliability of FPGAs is an issue of growing importance as process scaling approaches its extremity and FPGAs find their way into state-of-the-art SoCs, NoCs, and ACAPs. A firm understanding is needed for various changes that FPGAs experience as they age, the factors that influence them, and methods to predict their health accurately. Subjecting them to highly accelerated life conditions results in parametric deviations in MOSFETs, that make up the FPGA primitives (LUTs, CLBS, and registers). This chapter has focused on the degradation mechanisms of NBTI, their exploitation by hardware Trojans to degrade FPGA reliability by inflicting power and timing closures in FPGA based applications and developing an efficient health estimation method based on kernel learning method.

This method exploits the features extracted from an FPGA primitive holding a logic circuit and its underlying P/NFETs. Additionally, the development of the Stochastic Filtering Optimization method helps to achieve better health estimation accuracy by addressing the hyperparameter selection problem.

The results indicate an effective detection capability of this method, whereby it enables the capturing of actual degradation trends in critical and faulty components with enhanced accuracy (97%) as compared to the Euclidean and Mahalanobis based methods. The most important and useful attribute of this health estimation method is its ability to provide an early failure warning before the actual application circuit failure. In most cases, the estimated failure time T_f was found to be less than the actual failure time T_{af}. While in some cases, the error was observed in health estimation. We were able to identify two main reasons for it. Firstly, the magnitude of the failure class features in the projected space is significantly larger than the magnitude of the healthy class features. This resulted in biasing the conditional probability toward the faulty class. Secondly, both the healthy and failure classes are spaced well-apart in the higher dimensional space. This makes their classification very easy; however, we resolved it by choosing the failure threshold value near to the theoretical failure limit. Nevertheless, in the higher dimensional space, it is imperative to have a method for tighter control on the distribution of faulty features.

We also observed some variability in the FPGA circuit health estimate of the sequential logic, despite the inclusion of the regularisation parameter to counter the impact of component tolerances. Therefore, in order to control the spread of healthy class features in the projected space, the addition of an application-specific constraint in the hyperparameter selection algorithm could prove useful.

REFERENCES

[1] M. Pecht and R. Jaai, “A prognostics and health management roadmap for information and electronics-rich systems,” *Microelectron. Reliab.*, vol. 50, no. 3, pp. 317–323, Mar. 2010.

[2] Z. Liu, T. Liu, J. Han, S. Bu, X. Tang, and M. Pecht, “Signal Model-Based Fault Coding for Diagnostics and Prognostics of Analog Electronic Circuits,” *IEEE Trans. Ind. Electron.*, vol. 64, no. 1, pp. 605–614, 2017.

[3] M. Li, W. Xian, B. Long, and H. Wang, “Prognostics of Analog Filters Based on Particle Filters Using Frequency Features,” *J Electron Test* 29, pp. 567–584, 2013.

[4] C. Zhang, Y. He, and L. Yuan, “A Novel Approach for Analog Circuit Fault Prognostics Based on Improved RVM,” *J Electron Test* 30, pp. 343–356, 2014.

[5] Zhou, Jingyu et al. “A novel prediction method about single components of analog circuits based on complex field modeling.” *The Scientific World Journal* vol. 2014, 2014.

[6] S. Kumar, N. M. Vichare, E. Dolev, and M. Pecht, “Microelectronics Reliability A health indicator method for degradation detection of electronic products,” *Microelectron. Reliab.*, vol. 52, no. 2, pp. 439–445, 2012.

[7] E. Sutrisno, “Fault detection and prognostics of IGBT using k-nearest neighbor classification algorithm,” M.S. thesis, Dept. Mech. Eng., Univ. Maryland, College Park, MD, USA, 2013.

[8] Z. Zhang, Z. Duan, Y. Long, and L. Yuan, “A new swarm-SVM-based fault diagnosis approach for switched current circuit by using kurtosis and entropy as a preprocessor,” *Analog Integr. Circuits Syst.*, vol. 81, no. 1, pp. 289–297, 2014.

[9] B. Long, S. Tian, and H. Wang, “Diagnostics of Filtered Analog Circuits with Tolerance Based on LS-SVM Using Frequency Features,” *J. Electron. Test.*,vol. 28, pp. 291–300, 2012.

[10] S. Menon, X. Jin, T. W. S. Chow, and M. Pecht, “Evaluating covariance in prognostic and system health management applications,” *Mech. Syst. Signal Process.*, vol. 58–59, pp. 206–217, 2015.

[11] B. T. Hofmann, B. Sch, and A. J. Smola, “KERNEL METHODS IN MACHINE LEARNING 1 ¨ lkopf By Thomas Hofmann, Bernhard Sch o and Alexander J. Smola,” vol. 36, no. 3, pp. 1171–1220, 2008.

[12] J. Suykens and J. Vandewalle, “Least squares support vector machines classifiers,” *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 2000.

[13] O. Chapelle and V.Vapnik, “Model selection for support vector machines,” in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 230–236,1999.

[14] T. Glasmachers and C. Igel, “Maximum likelihood model selection for 1-norm soft margin SVMs with multiple parameters,” *IEEE Trans. Pattern Anal.*, vol. 32, no. 8, pp. 1522–1528, Aug. 2010.

[15] S. S. Keerthi, “Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms,” *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1225–1229, 2002.

[16] F. Friedrichs and C. I. Ã, “Evolutionary tuning of multiple SVM parameters,” *Neurocomputing*, vol. 64, pp. 107–117, 2005.

[17] S. L. Ã and M. Tan, “Neurocomputing Tuning SVM parameters by using a hybrid CLPSO – BFGS algorithm,” *Neurocomputing*, vol. 73, no. 10–12, pp. 2089–2096, 2010.

[18] A. Boubezoul, “Application of global optimization methods to model and feature selection,” *Pattern Recog.*, vol. 45, pp. 3676–3686, 2012.

[19] B. Long, M. Li, H. Wang, and S. Tian, “Diagnostics of Analog Circuits Based on LS-SVM Using Time-Domain Features,” *J. Electron. Test.*, pp. 2683–

2706, 2013.

[20] J. Platt, “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods,” in *Advances in Large Margin Classifiers*. Cambridge, MA, USA: MIT, pp. 61–74,1999.

7 DISCUSSION AND CONCLUSIONS

The core aim of this research was to develop an integrated ‘Design for Prognostics and Security’ in FPGAs, taking into consideration the aggravated impact of the continual miniaturisation of technology node and the rising hardware threats, hardware Trojans in particular. The design provides a viable capability to both the industry and the researchers for prognosticating and managing the health of modern FPGAs using an integrated approach that combines the realms of FPGA reliability and security and gives an optimised solution in the form of the FPGA security scheme and a Kernel-based health estimation/prognostics method. This research is, therefore, a substantial effort to strengthen the realms of reliability and security that govern the health dynamics of Field Programmable Gate Arrays (FPGAs) for innumerable mission-critical applications.

In this chapter, we present a brief discussion to highlight the salient findings and contributions to these realms. Section 7.1 gives an account of how the research aim and objectives have been achieved. This is followed by the presentation of key contributions to the existing VLSI reliability and security domains in Section 7.2. Two key avenues for the future work are recommended in the final Section 7.3.

7.1 Addressing the Research Aim and Objectives

The overall scientific research aim was to develop an integrated ‘Design for Prognostics and Security in Field Programmable Gate Arrays (FPGAs) that facilitates their reliability and security enhancement, enables NBTI-based hardware Trojan detection and mitigation within their reconfigurable fabric, and helps estimate their health. This aim has been achieved with the fulfilment of the following objectives:

Objective 1: Develop an Integrated FPGA Health Management (IFHM) Framework.

Evidence 1: This objective was accomplished with the development of a high-level IFHM framework, which provides a guideline for the VLSI design and manufacturing community (including researchers and expert end-users) to develop highly optimised FPGA security and prognostics schemes by adopting integrated approach. This framework was conceived on the pretext that the existing individualistic approach toward FPGA health management does not consider the essential elements of reliability, prognostics and security collectively. As a result, the fragmented solutions are developed which, however, do not reflect the true state of the operational condition of an FPGA. This framework negates the fragmented approach and provides a guidance for the FPGA researchers, design and manufacturing engineers, and expert end-users in establishing the relationship between 'degradation/failure mechanism' and 'hardware threat/attack', determining 'failure precursor', constructing and optimizing the experimental set-up, defining test conditions and estimating the health of an FPGA in a composite manner. The subsequent design and development of the FRED sensor (Chapter 4), the FPGA Security Scheme (Chapter 5), and the Kernel-based health estimation method (Chapter 6) are the products of this framework and validate its concept.

The subsequent automation of this framework and integration with electronic design automation (EDA) tools would be highly useful.

Objective 2: Design and implement a small footprint and highly sensitive on-chip sensor for the detection of frequency and subsequent delay degradation in modern FPGAs due to aggravated BTI mechanism.

Evidence 2: This objective was achieved by the design and implementation of a FREquency Degradation (FRED) detection sensor (Chapter 4). The sensor detects and provides a measure of decrement in the frequency of its uniquely designed dual delay-line based segments with the ageing of FPGA primitives due to the BTI degradation mechanisms. The fixed sensor segment (FSS) of FRED is used as a reference with near-zero stress and the dynamic sensor segment (DSS) is built to experience high temperature and voltage stresses. Configured with variant gate length and types, the sensor outputs an accurate measure of the frequency difference between the FSS and DSS segments. Due to near-zero stresses on FSS, it is equally good for the calibration of the sensor, which helps maintain the measurement accuracy. The simulation and real-time experiments under normal and accelerated temperature and voltage conditions validated the effectiveness of the sensor in detecting and measuring small to large delay variability by observing changes in the frequency difference between the FSS and DSS segments.

Based on its capability to capture FPGA ageing in terms of the frequency and delay degradation across the whole FPGA surface, FRED can be considered a good candidate for the detection of malicious circuits, called hardware Trojans, that are based on the parametric variations (such as threshold voltage) in transistors.

Objective 3: Design and implementation of an FPGA Security Scheme capable of detecting a hardware Trojan and providing effective mitigation.

Evidence 3: This was the most important objective and can be termed as the backbone of this research work. We achieved it by designing and implementing a unique FPGA Security Scheme comprising the design and implementation of a novel threshold voltage shift-based hardware Trojan sub-scheme, hardware Trojan detection sub-scheme comprising an improved version of FRED sensor, and a hardware Trojan mitigation sub-scheme based on the online transistor dynamic scaling (OTDS) (Chapter 5).

This FPGA Security Scheme would prove to be a trustable and highly effective capability for a wider industrial community in bolstering their products' reliability and performance. It would augment their capability to defend and deter against hardware Trojans.

Objective 4: Prognosticate the health of an FPGA under the influence of hardware Trojan

Evidence 4: We accomplished this key objective by developing a Kernel-based machine learning method to prognosticate the health of an FPGA under the influence of a hardware Trojan (Chapter 6). This method exploits the features extracted from an FPGA primitive holding a logic circuit and its underlying PFETs. Additionally, it is augmented with a stochastic filtering optimization method that helps to achieve better health estimation accuracy by addressing the hyperparameter selection problem. The most important and useful attribute of this health estimation method is its ability to provide an early failure warning before the actual application circuit failure. This method provides a classification accuracy of 94% on average (calculated taking into consideration different circuit implementations on FPGA).

7.2 Contribution to Knowledge

Any research work that attempts to understand the intricacies of security in the wake of evolving cyber-attacks is, in itself, a decent contribution to the field of hardware security, in particular and cybersecurity, in general. We have delved much deeper into finding novel and practicable solution to FPGA reliability and security issues. In the process, we have made some significant contributions to the VLSI stream of micro and nanoelectronics. These include:

- **The integrated FPGA health management (IFHM) framework**: is a highly useful tool for deriving valuable and relevant research data on FPGA security and reliability, especially for the semiconductors' research community. It provides an all-encompassing and well-directed approach to understand the health contours of FPGA and develop efficient schemes accordingly. When automated, it would be a vital addition to the design rule check (DRC) feature/library of FPGA/ASIC design tools.
- **Lightweight On-chip FREquency Degradation (FRED) Detection Sensor**: The designing and implementation process of FRED provides a cogent understanding of the varying parametric behaviour of CMOS transistors and FPGA primitives due to their interactions and exposure to various stochastic and systematic variations. The study and investigation of the changing threshold voltage and drain current, and their impact on frequency and circuit delays provide a useful knowledge for the sensor designer to optimise the sensor design. In addition, FRED with a high sensitivity and low area and power consumption along with improved quality factor can be incorporated in FPGA designs by the manufacturers not only for frequency and delay measurements but for accurate temperature readings as well, with some

minor modifications. Most importantly, it has been verified and validated to be an efficient hardware Trojan detector (for threshold-voltage shift-based hardware Trojans).

- **Development of a Threat Model**: We have developed a unique hardware Trojan threat model based on a high-end defence asset - a naval warship, fitted with an 'Integrated Self-Protection System' (ISPS) (Chapter 5, Section 5.2). It helps understand the implications of a hardware Trojan-infected FPGA on the system/sub-system/module it has been fitted with. It could range from SoCs and NoCs to LRUs, complex computation modules, radar transceiver systems, and sensitive cipher-decipher assemblies. This model can be utilised to build different FPGA security schemes (based on the multiple types of hardware Trojans) to defy micro-architectural level hardware attacks and ensure the confidentiality, reliability, and the operational availability of computational systems.
- **Design and Implementation of a Novel Hardware Trojan:** Leveraging the NBTI degradation mechanism's growing intensity with the continual downscaling of FPGA technology node, we designed and developed a novel lightweight and stealthy hardware Trojan in a 28nm device (Chapter 5, Section 5.3). This Trojan gets triggered with minor shifts in threshold voltage and disrupts the operation and function of any combinatorial and sequential circuit implemented in the FPGA. This design can be exploited by researchers to develop a threshold voltage triggered hardware Trojan based on PBTI degradation mechanism as well and accordingly develop various mitigation and prevention schemes.
- **FPGA Security Scheme**: In consonance with the IFHM framework, we devised an FPGA/ASIC - implementable Security Scheme to counter the detrimental impact of hardware Trojans. This scheme involves: 1) Ingress of a stealthy threshold voltage-triggered hardware Trojan-(**HT Infection Scheme**), 2) Detection of hardware Trojan using lightweight Threshold Voltage - aware sensor ($\boldsymbol{S_{Vth}}$)-(**HT Detection Scheme**), and 3) Mitigating the impact of hardware Trojan using online transistor dynamic scaling (***OTDS***)-(**HT Mitigation Scheme**). The complete description is given in Chapter 5. This

scheme can be capitalised on by the UK Research and Innovation's 'Industrial Strategy Challenge aiming at radically updating the foundation of the UK's insecure digital computing infrastructure.

- **Kernel-based FPGA Health Estimation/Prognosis**: The culmination of our research work is the development of a Kernel-based method to prognosticate FPGA health under the influence of a threshold-voltage triggered hardware Trojan (Chapter 6). This alongwith the above contributions is also unique in the sense that it is a completely new approach towards FPGA health estimation/prognosis. To the best of our knowledge, Machine Learning (ML) has not been used for an FPGA before. Previously, such techniques/methods have been limited to discrete electronic components.

  This method is the finality of the concept of IFHM framework that provides a composite picture of the health of an FPGA. This method can be further matured with extensive testing with a view to enhancing its classification accuracy to 100%. This method could be a most viable as well as cost and time efficient choice for the FPGA manufacturers to enhance their post-manufacturing reliability monitoring and testing programs.

7.3 Future Work

While conducting this research work and writing down the thesis, we have been intrigued by a number of concerns. For instance, there is a relentless need to secure the semiconductor manufacturing supply chain completely. This includes trusted design kits, reliable tool flows, full-spectrum (encompassing reliability, prognosis, and security) design libraries, manufacturing, packaging, and assembly. The industry still seems reluctant or incapacitated to understand the varying nature of malicious hardware Trojans. How can we analyse them? Is there any anti-virus program for hardware abstraction level devices? Should it be developed on signatures or behaviour/anomaly? Would a completely-trusted delivery be possible in the future autonomous systems? How effective is Tamper resistance? What can be adjudged as the best set of practices to detect, mitigate, and prevent the threat of malicious hardware? What can we learn from the (failed?) efforts to detect subversion in VLSI devices, particularly FPGA based

SoCs? What can be a good defence, dependability, and trust strategy for the existing and future hardware development?

Nevertheless, keeping our research work in perspective, we recommend the following as significantly important future works.

7.3.1 Automation of Integrated FPGA Health Management (IFHM) Framework

The proposed IFHM framework is a manual guidance on gathering requisite information and data related to FPGA reliability, security, and prognostics. Based on the analysis of the collated data, the FPGA security, reliability, and prognostics are designed and developed in a composite manner. This approach is efficient in time, cost, and research effort expended in developing an optimised FPGA health scheme. However, the automation of this framework would quadruple its efficiency and help build more robust and all-encompassing designs. This can then be integrated with the EDA tool kits (developed by Xilinx, Altera, Synopsys, Cadence etc.,) as a Design Rule Check (DRC) for FPGA/ASIC/SoC design flows.

7.3.2 Digital Security Hardening by Design – A Bio-inspired Approach

Traditionally, the performance evaluation of embedded devices (microprocessor, ASIC, and FPGA) and systems (SoC, NoC, and ACAP) is the function of their parametric analysis against designed specifications of a known-good or golden device. The electrical and physical parameters of the embedded device/system under test/evaluation are compared with the standard parameters of the golden device. Any deviation so observed is simultaneously recorded as a peculiar behavioural response (symptom) of the device and categorised into specific hardware threat (hardware Trojan, side-channel attack, counterfeit). However, in the absence of a known-good or a golden device, the performance evaluation becomes a challenging task – time consuming, resource-intensive, and un-reliable. Moreover, unavailability of tools to check the security of design, needs for security upgrade when breaches are detected after fabrication and release of the products, needs for hardware roots of trust, and lack of security model in component/devices libraries, currently used in EDA tools such as cadence and

Mentor Graphics, provide impetus for defining an all-inclusive bio-inspired intervention.

In view of the above, a bio-inspired vaccination approach may be adopted in which the embedded devices/systems are treated as artificial beings infested with malicious circuitry/software and other bugs that remain dormant unless triggered internally or by any external source. Such an approach eliminates the requirement of a golden device and helps build performance profile of the device in a composite manner. Taking lead from the 'Vaccination Mechanism', embedded devices/systems could be made immune at the microarchitectural level against hardware threats - hardware Trojans in specific. The devices/systems can then be exposed to a large set of known threat models to acquire corresponding symptoms, learn from them, and then construct advanced security solutions comprising early threat detection and mitigation schemes for unknown threats (while mimicking engineering solutions from biological counterparts). This would entail:

- Development of hardware Trojan models and integrating them with VLSI design along with the controllability mechanism to defuse them at the end of manufacturing tests.
- Enhancing the existing design for testability (DFT) techniques with novel approaches that allow construction and integration of advanced reconfigurable sensors with the device, called Primitive Security Elements or simply security primitives.
- Mechanisms to test the device/system through DFT, enabling fast and efficient data collection while the intended Trojans are activated or deactivated.
- Data analysis and signal processing to determine symptoms associated with known Trojans.
- Development of ML algorithms to extend the symptoms from known threats to unknown threats.
- Performance evaluation based on the symptoms' analysis and categorisation.

- Building components and devices libraries for various silicon technologies and foundries using threat models and associated symptoms as objects.

7.3.3 Alleviation of Si-H Bond-Breaking Impact on CMOS Transistors

The Si-H (Silicon-Hydrate) bonds' breakage leads to the initiation of the degradation mechanisms of BTI and HCI. As a result, these phenomena generate performance issues for CMOS transistors and this trend seems to get more pronounced despite the miniaturisation of VLSI technologies and usage of heavily nitrided oxides. A number of materials other than amorphous silicon can be the viable candidates for the alleviation of this undesired phenomenon. It is, therefore, considered prudent to investigate new materials and their practicality for state-of-the-art semiconductor devices so as to suppress performance degradation mechanisms such as N/PBTI and help increase their reliability.

7.3.4 Probability of Detection of Hardware Trojans Through Surface Effects' Monitoring

Despite the advanced thermal management/system monitors implemented within FPGAs/SoCs, detection of hotspots within FPGA fabric is not precise and accurate. As a result, any malicious circuit designed to trigger with high temperatures, will not be detected. We observed this during the accelerated thermal and power cycling experiments. The existing system monitors provide an overall junction temperature and all countermeasures/safeties are developed/activated accordingly. The hotspots' generation point has a much higher temperature, sufficient enough to trigger NBTI with corresponding increase in threshold voltage, thereby triggering the Trojan without leaving/showing any noticeable trace /impact. There is, therefore, a critical need to study the possibility of detection of hardware Trojans and develop advanced and more robust techniques to enable their detection by swift identification/capturing of hotspots as well as monitoring and evaluating the device surface effects.

7.3.5 Impact of Run-time Transistor Width Scaling on the Usable Area of an FPGA Device

The researchers/designers may have to make certain viable trade-offs between security implications (at any specific instant of time) and the FPGA optimal performance. As a result, compromises on area, power, and timing may take place. Keeping these factors in perspective, we have proposed this whole FPGA security scheme. We have kept the scheme viable in such a way that it should give improved area, power, and timing (system performance) figures, and at the same time never allowing hardware Trojan trigger by scaling transistor gates as and when thresholds are likely to be violated. However, despite this, the implementation and then the subsequent operation of OTDS (Online Transistor Dynamic Scaling) scheme during the device runtime, may impose some restrictions on the useable area/resources of FPGA and its dynamic reconfiguration. This could be the increase in the capacitive load of the circuit, causing a slight increase in the propagation delay of the monitored signal. In certain cases, there may be a considerable increase in the operating frequency. The question here is whether affecting transistor width will not have a massive effect on the FPGA device as a whole. We would, therefore, encourage the future researchers to critically evaluate the proposed OTDS scheme further and bring forth some highly useful performance evaluations.

It is envisaged that the above-mentioned future work recommendations would help enhance the realms of VLSI reliability and security a step further our research work and provide a robust set of tools to tackle the most damaging hardware and cyber threats.

APPENDICES

Appendix A Improving the Hardware Trojan Detectability

Keeping the patterns of false prediction in perspective, we consider improving the proposed sensor's detection sensitivity by adding two additional pairs of ROs to each of the sensor segment (*Fixed and Dynamic*). The frequencies of all these pairs of RO segments are measured consecutively, during different thermal cycles. Subsequently, the average of FSS *(Fixed Sensor Segment)* and DSS *(Dynamic Sensor Segment)* frequencies is calculated to determine the presence of malicious hardware Trojan.

A.1 Spread Reduction by Averaging Method

Assuming there is *n* number of ROs in the fixed and dynamic sensor segments, their respective frequencies can then be considered as random variables and denoted by a_1, a_2, …., a_n and b_1, b_2,....., b_n, respectively. as the distribution of $\boldsymbol{g_0(f_{fd})}$ depends upon the frequency differences of both the fixed and dynamic sensor segments; we can derive the following equation:

$$X_i = A_i - B_i \qquad (5\text{-}5)$$

In this equation, ***X_is*** is Gaussian, as both the ***A_is*** and ***B_is*** are Gaussian. We further assume the variables ***A*** and ***B*** to have the same mean and variance, as all the RO segments undergo the same process variations. The aim is to determine the mean and variance of a newly formed random variable ***Z_n***. Mathematically, this can be represented as follows:

$$Z_n = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Ai} - \frac{1}{n}\sum_{i=1}^{n} \mathrm{Bi} \qquad (5-6)$$

$$= \frac{1}{n}\sum_{i=1}^{n} (\mathrm{Ai} - \mathrm{Bi}) = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad (5-7)$$

The resultant random variable $\boldsymbol{Z_n}$ will be, therefore, Gaussian as all the ***Xis*** are Gaussian. Based on this, the mean and variance are expressed in the following mathematical form:

$$E[Z_n\,] = E[\frac{1}{n}\sum_{i=1}^{n} X_i\,] \; = \frac{1}{n}\left(\mathrm{E}\left[\sum_{i\,=\,1}^{n} X_i\right.\right.$$

$$= \frac{\mathrm{n}\; x\; \mu}{n} = \mu \qquad (5-8)$$

$$var(Z_n\,) = var\left(\frac{1}{n}\sum_{i\,=\,1}^{n} X_i\right.$$

$$= var\left(\sum_{i=1}^{n}\frac{X_i}{n}\right)$$

$$= \frac{1}{n^2}\sum_{i\,=\,1}^{n} var(\,X_i) \;+\; \frac{1}{n^2}\sum_{i\,\neq\,j} cov\;(X_i, X_j) \qquad (5-9)$$

In the above equations, $\boldsymbol{E[Z_n\,]}$ is the expected value of the random variable $\boldsymbol{Z_n}$- equal to the mean of a Gaussian random variable. Whereas $\boldsymbol{var(Z_n\,)}$ represents the variance of the random variable $\boldsymbol{Z_n}$ and $\boldsymbol{cov\,(X_i, X_j)}$ is the covariance between the random variables $\boldsymbol{X_i}$ *and* $\boldsymbol{X_j}$.In this mathematical model, we assume the frequencies of all the RO segments to be independent so that the random variables $\boldsymbol{X_1, X_2\,, \ldots\ldots X_n}$ also become independent. It, therefore, results in all the covariances in (9) becoming zero.

$$var(Z_n\,) \;\; = \frac{1}{n^2}\sum_{i\,=\,1}^{n} var(\,X_i) = \frac{n\;x\;\sigma^2}{n^2} \;\; = \frac{\sigma^2}{n} \qquad (5-10)$$

Keeping the above equation (9) in view, the mean **(μ)** and the standard deviation **(σ)** of $\boldsymbol{Z_n}$ can be derived as follows:

$$\mu Z_n \;\; = \; \mu \qquad (5\text{-}11)$$

$$\sigma Z_n \;\; = \; \frac{\sigma}{\sqrt{n}} \qquad (5\text{-}12)$$

As can be seen in (5-8) and (5-11), the mean of the average difference $\boldsymbol{Z_n}$ remains unchanged when compared with each $\boldsymbol{X_i}$. On the other hand, the variance of $\boldsymbol{Z_n}$ is dependent on $\sqrt{\boldsymbol{n}}$. A similar derivation is carried out to estimate the resultant mean and variance for the distribution at time t, $\boldsymbol{g_t(f_{FD})}$. We, therefore, infer that the overlapping area between the two distributions can be reduced to an almost negligible amount by adding additional RO pairs to both the fixed and dynamic segments, as is evident from Figure 5- 18(b).

Appendix B Determining Maximum Frequency Degradation

An accurate and precise capturing of frequency degradation in ring oscillators is key to the correct and authentic assessment of hardware Trojan's triggering, its impact, and a reliable measure of the sensor's sensitivity. We, therefore, experimented to determine the maximum frequency degradation experienced by DSS RO pairs when negative bias and elevated temperatures are applied as per the hardware Trojan insertion scheme described in Section 5.3.

and coarse as well as fine stretching operations *(stress-time)* used in [61] to minimise measurement errors. We observe how the frequency degradation (with subsequent delays and ageing), *δf,* changes with the percentage frequency differences at varying threshold voltages. A total of 10K samples were taken at each thermal (60, 90, and 125ºC) and negative bias (-1.2V, -1.4V, and -2.0V) points. The scatter plot of frequency degradation *δf* against frequency difference $\partial f_{t\ DSS}$ at time *t* is shown in Fig. 20, where $\partial f_{t\ DSS}$ = ($ft_{(-2.0V)}$ - $ft_{(-1.4V)}$ - $ft_{(-1.2V)}$) / $ft_{(-1.2V)}$. As is evident, $ft_{(-2.0V)}$, $ft_{(-1.4V)}$, and $ft_{(-1.2V)}$ are the frequencies of DSS RO pairs that are exposed to negative bias and increasing temperature stresses. A positive correlation ($\boldsymbol{\rho}$)for frequency degradation and normalised frequency differences is observed that indicates the ageing and delay degradation in this specific threshold voltage triggered hardware Trojan environment. Based on this experimental observation, we undertook mathematical analysis to determine the relationship that could enhance sensor accuracy defined by the interdependence of temperature, threshold voltage, oscillation count/frequency, and ageing/delays variability.

As the DSS RO pairs are subjected to temperature and threshold voltage variations at time *t*, the oscillation count/frequency $f_{t\ DSS}$ begins to fall. It becomes lower than the frequency $f_{0\ DSS}$ at time *0*. This frequency degradation $\boldsymbol{\delta f}$ can, then, be given as:

$$\delta f = f_{0\ DSS} - f_{t\ DSS} \qquad (5-13)$$

With the application of negative bias at three different values in time *0*, the percentage frequency difference is resultantly calculated as:

$$\partial f_{0\,DSS} = \frac{f_{0\,DSS,V_{DD1}} - f_{0\,DSS,V_{DD2}} - f_{0\,DSS,V_{DD3}}}{f_{0\,DSS,V_{DD3}}} \quad (5\text{-}14)$$

where, $\boldsymbol{V_{DD1}} > \boldsymbol{V_{DD2}} > \boldsymbol{V_{DD3}}$. As there exists a positive correlation between $\boldsymbol{\delta f}$ and $\boldsymbol{\partial f_{0\,DSS}}$, we aim at identifying DSS RO pair that experiences a maximum frequency degradation relative to percentage frequency differences at the afore-mentioned negative bias and temperature stress values, mathematically:

$$\delta f \overset{\rho}{\leftarrow} \partial f_{0\,DSS} \quad (5\text{-}15)$$

Then, the frequency degradation for the sensor can be expressed as follows:

$$\delta f = \Delta f_t - \Delta f_0 \quad (5\text{-}16)$$

where,

$$\Delta f_t = f_{t\,FSS} - f_{t\,DSS} \quad (5\text{-}17)$$

We also consider the impact process variations (PVs) could have on frequency (delay/ageing) degradation $\boldsymbol{\delta f}$ and the percentage frequency difference $\boldsymbol{\partial f_{DSS}}$. With the positive correlation between the two, it is possible to have an optimal estimate $\boldsymbol{\delta^{\Lambda} f}$ for $\boldsymbol{\delta f}$. Minimum mean-square error (MMSE) estimator, for instance, provides versatility to achieve reduced mean square error and make more realistic estimates [62]. The DSS RO degradation is, therefore, expressed using the minimum mean-square error (MMSE) estimator, as follows:

$$\delta \hat{f}_{DSS} = \rho \frac{\sigma_{\delta f_{DSS}}}{\sigma_{\partial f_{DSS}}} \left(\partial f_{DSS} - \mu\partial f_{DSS}\right) + \mu\delta f_{DSS} \quad (5\text{-}18)$$

where, $\boldsymbol{\rho}$ defines the correlation between frequency degradation in dynamic sensor segment ($\boldsymbol{\delta f_{DSS}}$) and percentage frequency difference ($\boldsymbol{\partial f_{DSS}}$); $\boldsymbol{\sigma_{\delta f_{DSS}}}$ and $\boldsymbol{\sigma_{\partial f_{DSS}}}$ connotate the standard deviations for $\boldsymbol{\delta f_{DSS}}$ and $\boldsymbol{\partial f_{DSS}}$ respectively. Whereas, $\boldsymbol{\mu\delta f_{DSS}}$ and $\boldsymbol{\mu\partial f_{DSS}}$ represent the mean for $\boldsymbol{\delta f_{DSS}}$ and $\boldsymbol{\partial f_{DSS}}$ respectively.

The MMSE estimator for the overall sensor degradation ($\boldsymbol{\delta f}$), as opposed to a particular sensor segment, can now be expressed as follows:

$$\delta\hat{f}_s = \Delta\hat{f}_t - \Delta\hat{f}_0 = \left(\hat{f}_{t\,FSS} - \hat{f}_{t\,DSS}\right) - \left(\hat{f}_{0\,FSS} - \hat{f}_{0\,DSS}\right)$$

$$= -\left(\hat{f}_{0\,FSS} - \hat{f}_{t\,FSS}\right) + \left(\hat{f}_{0\,DSS} - \hat{f}_{t\,DSS}\right) \qquad (5\text{-}19)$$

Since, the frequency degradation is assumed to be negligible in case of fixed sensor segment RO pairs, $\boldsymbol{\hat{f}_{0\,FSS} = \hat{f}_{t\,FSS}}$, the above equation can be written as:

$$= \left(\hat{f}_{0\,DSS} - \hat{f}_{t\,DSS}\right)$$

$$= \delta\hat{f}_{DSS} = \rho\frac{\sigma_{\delta f_{DSS}}}{\sigma_{\partial f_{DSS}}}\left(\partial f_{DSS} - \mu\partial f_{DSS}\right) + \mu\delta f_{DSS} \qquad (5\text{-}20)$$

The above relation implies that with $\boldsymbol{\rho}$ being positive, the higher percentage frequency difference between the FSS and DSS RO pairs will, in turn, maximise the sensor frequency *(and subsequent delay/ageing)* degradation. It is represented by the separation between two distributions at ***t = 0*** and ***t = t***. This further implies that in the sensor with more RO pairs to select from, the one with the maximum percentage frequency difference within DSS RO pairs at ***t = 0*** must be selected. This results in maximising the distance between the two distributions of frequency difference and minimising the probability of false prediction, as shown in Figure 5-18(a).

Keeping in view the above mathematical derivations and ‘selection strategy’(*as delineated in process flow – Figure 5-20*), the detectability of hardware Trojan by the sensor is set for optimisation. Accordingly, we define the process variations based on transistor length (L) and oxide thickness (Tox), as given in Table 5-5 and choose ***‘PVc’*** class of process variations as an extreme (worst) case to determine the pre-trigger value of frequency degradation, relative to percentage shift in the threshold voltage. Also, the two sensor segments (FSS and DSS) are implemented close to each other to eliminate the impact of undefined environmental variations upon measurements and the accuracy of detection.

The process flow (Figure 5-20) targets the selection of the best (with maximum frequency degradation) FSS and DSS RO-pair by, initially, selecting all the six

RO-pairs and then capturing their frequencies. These frequencies are stored by two vectors, defined as$\vec{f}_{FSS}$ = $[f_{FSS1}, f_{FSS2}, f_{FSS3}]$ and $\vec{f}_{DSS} = [f_{DSS1}, f_{DSS2}, f_{DSS3}]$ and all the frequency differences are stored in a matrix defined as, $\Delta f = [\Delta f_{ij}]nxn$, where $\Delta f_{ij} = \vec{f}_{FSS}(i) - \vec{f}_{DSS}(j), \forall(i,j)$. If Δf_{ij} is positive, the fixed and dynamic RO-pair with minimum Δf_{ij} is selected. Otherwise, only negative Δf_{ij} values are taken to update Δf. In such a condition, the resulting distribution $g_0^{'}(.)$ presents a significantly reduced spread, as is evident in Figure 5-18(c).

However, at time *t,* the distribution $g_t(.)$ must be shifted to the right to increase δf even further. In such a condition, DSS RO is selected with maximum $\overrightarrow{\partial f}dss(j)$

$$= \frac{\boldsymbol{f_{0\,DSS,V_{DD1}}}(j) - \boldsymbol{f_{0\,DSS,V_{DD2}}}(j) - \boldsymbol{f_{0\,DSS,V_{DD3}}}(j)}{\boldsymbol{f_{0\,DSS,V_{DD3}}}(j)} \tag{5-21}$$

whereas, the corresponding FSS RO with maximum Δf_{ij} is selected to minimise the spread of both distributions, $g_0^{'}(.)$ and $g_t(.)$. Once the optimal RO pair is selected, the frequency difference Δf_{ij} is then stored to form the distribution $g_0^{'}(.)$. The threshold frequency f_{th} is finally calculated, to be referred to for the detection of hardware Trojan by comparing it with the frequency differences of FSS and DSS RO segments implemented in FPGA under authentication.