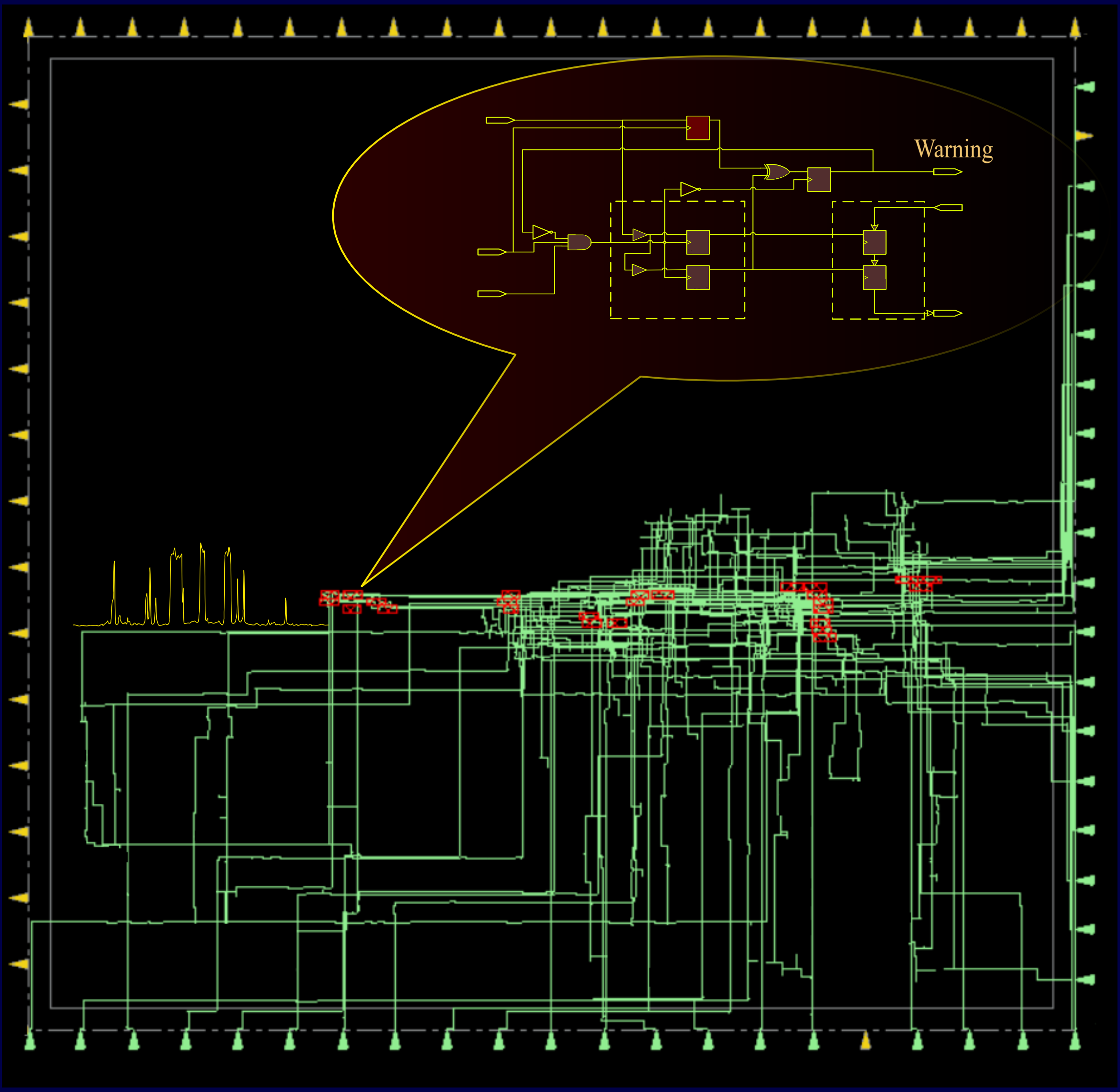
Design and Verification of Embedded Instruments for Detecting Intermittent Resistive Faults in Electronic Systems



Hassan Ebrahimi

DESIGN AND VERIFICATION OF EMBEDDED INSTRUMENTS FOR DETECTING INTERMITTENT RESISTIVE FAULTS IN ELECTRONIC SYSTEMS

Hassan Ebrahimi

DESIGN AND VERIFICATION OF EMBEDDED INSTRUMENTS FOR DETECTING INTERMITTENT RESISTIVE FAULTS IN ELECTRONIC SYSTEMS

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. ir. A. Veldkamp,
on account of the decision of the Doctorate Board,
to be publicly defended
on Friday 8 September 2023 at 16.45 hours

by

Hassan Ebrahimi

born on the 23$^{\text{rd}}$ of July, 1983
in Tehran, Iran

This dissertation has been approved by:

Promotors
Dr. ir. H. G. Kerkhoff
Dr. ir. M. Ottavi

Cover design:	Hassan Ebrahimi
Printed by:	Ipskamp printing
Lay-out:	Hassan Ebrahimi
ISBN (print):	978-90-365-5759-7
ISBN (digital):	978-90-365-5760-3
DOI:	10.3990/1.9789036557603
URL:	https://doi.org/10.3990/1.9789036557603

Abstract

No-fault-founds (NFFs) threaten the dependability of highly-dependable systems in the avionic and car industries. Moreover, they drastically increase the test and maintenance costs. One of the main causes of NFFs is intermittent resistive faults (IRFs). These faults can occur randomly in terms of time, duration and amplitude in every interconnection. The occurrence rate can vary from a few nanoseconds to months, making the evocation and detection of such faults a major challenge.

This thesis tackles IRF detection at the chip and board levels. First, the characteristics of IRFs have been extracted by observing two main causes of IRFs in boards: cold-solder joints and loose connectors, under thermal cycling experiments. Based on these measurements, a representative model of IRFs has been introduced and employed to develop an IRF generator. The proposed IRF generator facilitates the study of IRF effects on digital systems. In order to analyse the effect of IRFs at the transistor level, a software-based IRF generator has been developed. A hardware-based IRF generator has been built to evaluate IRFs at the board level. The measurement results show that cold-solder joints and loose connectors can cause IRFs, and that IRFs can evolve under thermal cycling and eventually become permanent faults.

Two embedded test instruments (ETIs) are proposed to detect IRFs at the chip and board levels. The ETIs are fully-digital, scalable, and compatible with the IJTAG standard. At the chip level, the effectiveness of the ETIs has been validated via simulation-based fault injections. It has been shown that the ETIs can be used to distinguish between intermittent, ageing, and permanent faults.

An algorithm is proposed to find the best locations in a design for IRF monitoring and ETIs insertion into the chip. The objective of this algorithm is to maximise the IRF coverage of a circuit. The algorithm maximises the IRF coverage for the most vulnerable parts of the circuit to IRFs, such as wire interconnections, vias and input pads. Two different selection strategies have been investigated. The first strategy selects nodes from end nodes only. In the second strategy, the node selection is from all nodes, including internal and end nodes. The simulation results reveal that the algorithm achieves a higher IRF coverage when using the second strategy. The efficacy of the algorithm is verified using simulation-based fault injections.

To tackle IRF detection at the board level, an enhanced version of the IEEE 1149.4 standard has been introduced. It has been shown that the proposed method can detect IRFs in the interconnections with mixed-signals on boards. The on-chip embedded test instruments have been reused for IRF detection at the board level. The efficacy of this reuse was evaluated using hardware fault-injections. Moreover, the ability of IRF detection using a combination of periodic testing and the ETIs was verified by evoking actual IRFs caused by cold solder joints under thermal cycling. The experimental results show that our on-chip ETIs can be reused periodically to detect IRFs at the board level.

At the end, IRF evoking at the chip level has been investigated. Evoking IRFs is very crucial during test mode. Without using evoking techniques, IRFs may remain inactive and, as a result, remain undetected. To speed up IRF detection, IRF evoking and monitoring should be employed simultaneously. A thermal cycling technique has been proposed to evoke IRFs in processors. A software procedure using customised workloads has been introduced for creating and performing thermal cycling. Furthermore, the feasibility of creating local hotspots in a chip has been investigated using two processors as case studies: the Plasma and Xentium processors. The simulation results demonstrate that the thermal cycling technique produces local hotspots only with a few degrees centigrade variation between functional units in the processors. Therefore, the thermal cycling technique can be used to evoke IRFs on the entire chip rather than locally. The proposed ETIs can be employed during thermal cycling to detect and localise IRFs.

Samenvatting

No-fault-founds (NFFs) bedreigen de betrouwbaarheid van zeer betrouwbare systemen in de luchtvaart- en auto-industrie. Bovendien verhogen ze in significante mate de test- en onderhoudskosten. Een van de belangrijkste oorzaken van NFFs betreft intermitterende resistieve fouten (IRFs). IRFs kunnen willekeurig voorkomen in tijd, duur en amplitude in elke onderlinge verbinding. De frequentie kan hierbij variëren van enkele nanoseconden tot maanden. Het oproepen en detecteren van dergelijke fouten is dan ook een grote uitdaging.

Dit proefschrift behandelt IRF-detectie op chip- en board niveau. Ten eerste zijn de kenmerken van IRFs geëxtraheerd door twee hoofdoorzaken van IRFs in PCBs te observeren, d.w.z. koud soldeerverbindingen en losse connectoren, tijdens experimenten met thermische cycli. Op basis van deze metingen is een representatief model van IRFs geïntroduceerd. Bovendien is het IRF-model gebruikt om een IRF-generator te ontwikkelen. De voorgestelde IRF-generator vergemakkelijkt de studie van IRF-effecten op digitale systemen. Om het effect van IRFs op transistor niveau te analyseren, is daarnaast een softwarematige IRF-generator ontwikkeld. Ook is er een op hardware gebaseerde IRF-generator gebouwd om IRFs op board niveau te evalueren. De meetresultaten laten zien wat koud soldeerverbindingen en losse connectoren IRFs kunnen veroorzaken. Bovendien kunnen IRFs evolueren onder thermische cycli en uiteindelijk permanente fouten worden.

Er worden twee ingebed testinstrumenten (embedded test instruments, ETIs) voorgesteld om IRFs op chip- en board niveau te detecteren. De ETIs zijn volledig digitaal, schaalbaar en compatibel met de IJTAG-standaard.Op chip niveau is de effectiviteit van de ETIs gevalideerd middels simulatie gebaseerde fout injecties. De resultaten laten zien dat de ETIs kunnen worden gebruikt om onderscheid te maken tussen intermitterende, verouderings- en permanente fouten.

Er wordt een algoritme voorgesteld om de beste locaties in een ontwerp voor IRF-monitoring en ETIs in de chip te vinden. Het doel van dit algoritme is om de IRF-dekking van een circuit te maximaliseren. Het algoritme maximaliseert de IRF-dekking voor de meest kwetsbare delen van het circuit naar de IRFs, zoals draadverbindingen, vias en input pads. Er zijn twee verschillende selectie strategieën onderzocht. De eerste strategie selecteert knooppunten van eind knooppunten. De tweede strategie betreft de selectie van alle knooppunten (zowel interne als eind-knooppunten). De simulatieresultaten laten zien dat het algoritme met de tweede strategie een hogere IRF-dekking behaalt. De efficiëntie van het algoritme is

geverifieerd met behulp van op simulatie gebaseerde fout injecties.

Om IRF-detectie op PCBs aan te pakken, is een verbeterde versie van de IEEE 1149.4-standaard geïntroduceerd. Het is aangetoond dat de voorgestelde methode IRFs kan detecteren in de onderlinge verbindingen met gemengd analoog / digitale signalen op PCBs. De on-chip ETIs zijn hergebruikt voor IRF-detectie op board niveau. Het hergebruik van onze on-chip embedded testinstrumenten voor IRF-detectie op board niveau is geëvalueerd met behulp van hardwarefout-injecties. Bovendien werd het vermogen van IRF-detectie met behulp van een combinatie van periodieke tests en de ETIs geverifieerd door werkelijke IRFs op te roepen die worden veroorzaakt door koude soldeerverbindingen onder thermische cycli. De experimentele resultaten laten zien dat onze on-chip ETIs periodiek kunnen worden hergebruikt om IRFs op board niveau te detecteren.

Tot slot is er onderzoek gedaan naar IRF-oproepen op chip niveau. Het oproepen van IRFs is erg cruciaal tijdens de testmodus. Zonder gebruik te maken van evoking-technieken, kunnen IRFs inactief blijven en als gevolg daarvan onopgemerkt blijven. Om IRF-detectie te versnellen, moeten IRF-opwekking en -monitoring gelijktijdig worden ingezet. Er is daarom een thermische cyclische techniek voorgesteld om IRFs in processors op te wekken. Hierbij is een softwareprocedure geïntroduceerd die gebruikmaakt van aangepaste werkbelastingen voor het maken en uitvoeren van bovengenoemde thermische cycli. Verder is de haalbaarheid onderzocht van het creëren van lokale hotspots in een chip met twee processors als case studies: de Plasma- en Xentium-processors. De simulatieresultaten tonen aan dat de thermische cyclus techniek alleen lokale hotspots produceert met een variatie van enkele graden Celsius tussen functionele eenheden in de processors. Daarom kan de thermische cyclische techniek worden gebruikt om IRFs over de hele chip op te roepen in plaats van enkel lokaal. De voorgestelde ETIs kunnen tijdens thermische cycli worden gebruikt om IRFs te detecteren en te lokaliseren.

Acknowledgements

This thesis was accomplished with the invaluable assistance and support of several individuals, whom I would like to acknowledge. I am deeply grateful, especially to my supervisor, Dr. ir. Hans Kerkhoff, for his exceptional mentorship and unwavering support throughout my PhD journey. His incredible patience and the extensive time he devoted to reviewing my thesis are highly appreciated. Additionally, I extend my thanks to Dr. ir. Marco Ottavi, for taking the time to review my thesis. Their contributions were instrumental in shaping the outcome of this research, and I am truly thankful for their guidance and input.

The research described in this thesis was conducted under the FP7 BASTION project, funded by the European Committee and the Netherlands Enterprise Agency. I would like to extend my heartfelt appreciation to all the members of the BASTION project for their active participation in the fruitful meetings and engaging discussions we had throughout this research.

I also would like to thank my colleagues at the University of Twente: Alireza Rohani, Ahmed Ibrahim, Jinbo Wan, Andreina Zambrano, Yong Zhao, Muhammad Aamir Khan, Ghazanfar Ali, and Jerrin Pathrose for their support and fruitful discussions. I am also thankful to our dedicated secretaries, Marlous Weghorst, Nicole Baveld, and Thelma Nordholt, as well as our technical support team comprising Bert Helthuis and Bert Molenkamp. Furthermore, I extend my appreciation to all other CAES members for fostering a productive and stimulating work environment. Furthermore, I am grateful for the support provided by my master's students, Marc Braklels and Dirk-Jan van de Sanden, in the implementation of the PCB for the IRF generator in this research. Their assistance was essential to successfully carry out this aspect of the study.

In addition, I would like to express my gratitude to all my friends in Enschede, with a special mention to Emad, Masoud, Navid, Alireza, Francesco, Dimitri, Guido, Himanshu, Nicole, Fanny, Michela, and many others whose names I cannot mention for the sake of brevity. Their friendship and unwavering support have meant a great deal to me throughout this journey.

I am truly grateful for the incredible support I received from my dear wife, Martina. Throughout this journey, she has stood by my side, enduring so much, and words cannot fully convey my appreciation for her. Additionally, I would like to extend my thanks to my wife's family, especially Franz and Nelly, for their support during this time.

Additionally, I wish to express my gratitude to my father for his constant support and to my mother, to whom I can never fully convey the credit she deserves. I also extend my thanks to my sister, Hamideh, and my brothers, Hossein and Hamed, who have been my supporters since childhood and remain by my side to this day. Their love and encouragement have meant the world to me.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Abstract– *The dependability of highly-dependable systems in the avionics and automotive industries is endangered by no-fault-founds (NFFs). NFFs can significantly increase test and maintenance costs. Intermittent (resistive) faults are known to be one of the main causes of NFFs. This thesis tackles the modelling, detection, and evoking of intermittent resistive faults at the chip and board levels. In this chapter, the background of the thesis is introduced first, with NFF and the subclass of intermittent faults being defined. Moreover, it introduces the modelling, detection, and evoking of intermittent faults. At the end, the outline, motivation and research questions of this thesis are presented.*

1.1 What are NFFs and Intermittent Faults?

With the continuous decrease of CMOS feature size, more electrical products show anomalous behaviour in the field but function normally during testing. These products are known as NFF (No Fault Found). They are also referred to as *could not duplicate*, *trouble not identified*, *retest OK*, *no trouble found* and so on [Dav05b, Söd07, Dav05a].

Manufacturers spend a lot of effort and money investigating the root cause of NFFs to increase product dependability and lower return costs. This is often called No Defect Found (NDF) or No Trouble Found (NTF) and is strongly associated with escape testing.

Many industries, such as avionics and automotive, have been suffering from NFFs for decades. The NFF rate for avionics systems is reported to be more than 400,000 per year. Over $2 billion was estimated to be spent annually on the NFF in 2013 [Raz16, Söd07]. The NFF rate can exceed 50% in the avionics industry. An example is reported in [Ste02] that shows a 67% NFF rate for F-16 radar systems. Several examples in the automotive industry are given in [Tho02] demonstrating how NFFs cost car manufacturers millions of dollars.

The time, materials, and shipping costs to exchange hardware are enormous concerning the cost of replacing the component. Furthermore, NFF returns can also indicate that customer problems have not been resolved, implying decreased customer satisfaction and brand value.

Intermittent faults are one of the major categories of NFFs. The root cause of intermittent faults might be the incompleteness of test coverage metrics due to missing fault models for certain defects and, consequently, the absence of certain types of deterministic test sets.

Faults in semiconductor devices can be classified as permanent, transient, and intermittent faults. Transient faults are induced by temporary environmental conditions such as neutrons from the atmosphere and energetic particles from packaging material. Hard (or permanent) faults reflect irreversible physical changes, mainly corresponding to manufacturing defects, such as contaminations in silicon devices or wear-out of materials. Transient and intermittent faults manifest very similarly. However, an intermittent fault can be distinguished from a transient one by the following criteria:

First, repairing the faulty circuit eliminates the intermittent fault, unlike transients, which cannot be erased by repair. Second, transient faults affect random locations, whereas intermittent faults frequently occur at the same location. Third, errors caused by intermittent faults are more likely to occur in bursts [Con08].

1.2 Categories of Intermittent Faults

Different categories of intermittent faults can be identified. They are as follows:

- **Intermittent stuck-at faults:**

  An intermittent stuck-at fault causes the value on the faulty signal line to intermittently be stuck at a logic value of *1* or *0*. Memory and register files, for example, are the most sensitive to intermittent stuck-at faults.

- **Intermittent short faults:**

  Intermittent short faults are shorts in wires or shorts in transistors. If an element is intermittently shorted to power or ground, it is equivalent to an intermittent stuck-at fault. An intermittent bridging fault occurs if two signal wires are shorted together.

- **Intermittent open faults:**

  Breaks or imperfections in interconnections, such as wires, contacts, transistors, and so on are known as intermittent open faults. Electro-migration, stress migration and intermittent contacts are the most common causes of these faults.

- **Intermittent timing faults:**

  Intermittent timing faults will result in timing violations and affect the propagation of data when they occur. They usually lead to result in incorrect data being written to storage cells. Intermittent path-delay and intermittent transition faults are two intermittent timing faults. Inductive noises, ageing, cross-talk, and process, voltage, and temperature (PVT) variations are the most common causes of intermittent timing faults.

- **Intermittent resistive faults:**

  Intermittent resistive faults (IRFs) belong to a particular category of NFF, characterised by sporadic low-amplitude resistive occurrences that appear randomly in time and location. However, they are often repairable once detected. IRFs can be considered as a superset of other types of intermittent faults, including intermittent (highly resistive) opens and (low resistive) shorts. Moreover, IRFs behave as intermittent delay faults in data paths, leading to random timing violations. This thesis concentrates on modelling, simulating, detecting, and evoking intermittent resistive faults.

  IRFs have a high occurrence rate ($> 50\%$) and are associated with significant testing costs, making them a considerable concern for future technology nodes [Ste08].

  It is important to note that intermittent faults are highly influenced by external environmental conditions, such as temperature and mechanical stimuli (e.g., vibration) [Sor10]. There are several physical root causes of intermittent resistive faults. As interconnection scaling continues in integrated circuits, there is a likelihood of increased occurrence of intermittent faults. Such faults may originate from factors like electro-migration (EM), soft breakdown (SBD), material residuals, induced voids, and cracks in 3D Through-Via Contacts (TSVs) within chips [Mar10].

  At the board level, several causes of IRFs have been reported [Tho02] including substrate cracks, open bumps, excess leakage currents at high temperatures, connector–ramp mating problems, lead frames unsoldered from housing, shorted lead frames, open leadfoots, cracked solder, cold solder joints, cracked resistors, cracked capacitors, and wire bond lifts.

1.3 Intermittent Fault Behaviour

Understanding the behaviour of intermittent faults and their impact on electrical systems is essential to effectively dealing with them.

Intermittent (resistive) faults are often caused by unstable or marginal interconnections. In advanced integrated circuits and printed circuit boards, there is

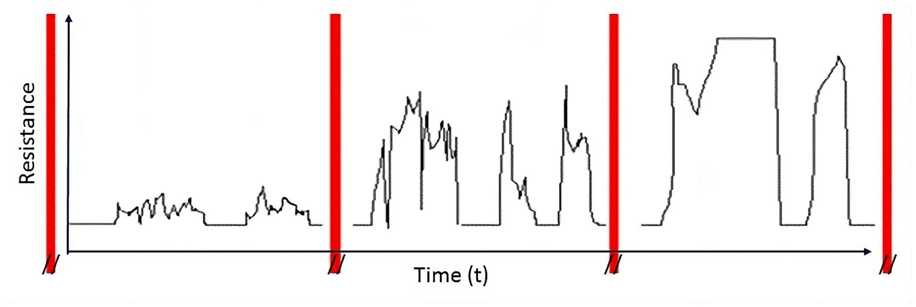


Figure 1.1: Typical appearance of an IRF and its development in time (ageing) [Hof08].

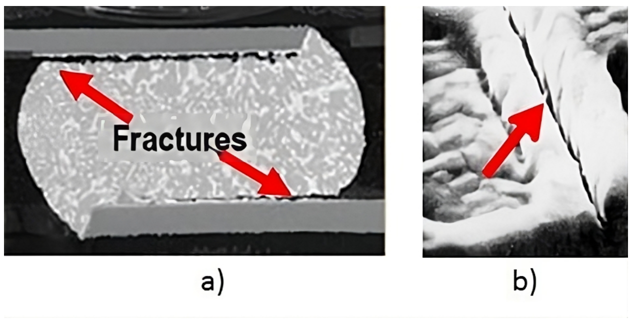


Figure 1.2: a) Cracks in a ball-grid interconnection and b) crack in a chip interconnection causing both IRFs [Hof08].

a significant number of interconnection wires and vias. Over time, these interconnections may experience electro-migration, temperature variations, and mechanical stress, leading to increased instability [And12].

An example of a measured intermittent resistive fault is shown in Figure 1.1 [Hof08]. It demonstrates that resistance values rise over time (ageing) and fault durations lengthen, eventually leading to permanent faults that are easily detectable. The most critical issue is to detect IRFs at the earliest possible stage. Figure 1.2 depicts two unstable interconnection conditions that result in IRFs. Figure 1.2a shows two cracks resulting from different (mechanical) stress factors between the chip package and the board. Figure 1.2b depicts a crack in chip interconnection due to a step-coverage problem. Both are very sensitive to temperature and mechanical stress/vibration.

1.4 Intermittent Fault Detection

Intermittent (resistive) faults have the potential to escalate in severity over the lifetime of a system, eventually transitioning into permanent faults. Consequently, early identification and repair of the defective module or interconnection are crucial to prevent system failures, especially in safety-critical systems.

Detecting intermittent (resistive) faults is a challenging task, mainly due to two significant issues in practice. The first issue pertains to the time of occurrence, which might be any moment in the future.

The second problem is that the duration of the event might be relatively brief. This necessitates the detection of extremely brief events. In the case of a purely digital solution, this necessitates very high sample rates or precise small-delay management. As a result, traditional test methods are unlikely to find these faults in practice.

Periodic testing is one method for detecting intermittent faults[Kra06]. The system is regularly tested with this method. By increasing the number of tests, there is a higher chance of detecting intermittent faults, but this comes at the cost of increased downtime and power consumption.

Another alternative approach is on-line monitoring [Jef05, Kat15]. Online monitoring involves observing a system while it performs its intended function [Nic98]. This method utilises embedded test instruments to monitor the health of a system while it operates. In contrast to offline testing, the advantage of this approach is that the system is monitored in the field under specific environmental conditions (e.g. voltage, temperature, and vibration stresses). This becomes essential because, during laboratory testing, these environmental conditions are unknown and challenging to replicate.

In this thesis, we explore intermittent resistive fault detection using both periodic testing and on-line monitoring approaches.

1.5 Intermittent Fault Evoking

The automotive and avionic sectors experience a substantial number of product returns, mainly attributed to no-fault-founds (NFFs), leading to significant additional test and maintenance expenses.

In the case of NFF, conventional testing methods fail to detect any fault in the returned product [Kha14, Ben02]. This is often due to the inability to evoke and activate the fault(s) during the testing period. Since intermittent faults contribute significantly to NFFs, evoking them becomes critical for reducing test and maintenance costs.

However, evoking these faults during the test is very hard. Because their activation is highly dependent on environmental conditions such as extensive temperature variations and vibrations. The probability that an intermittent (resistive) fault activates without additional stimuli during the test is extremely low. To increase the probability of detecting intermittent (resistive) faults, controlled external stimuli, such as temperature changes, should be employed in the test environment.

Evoking intermittent (resistive) faults can be conducted at board and chip levels. However, compared to intermittent fault evoking in boards, which can be accomplished by applying external stimuli such as temperature and vibration stresses, evoking intermittent faults in chips is a more challenging task. Thermal ovens and infrared approaches are two commonly used methods to elevate the temperature of a chip. However, they suffer from non-controllability. Therefore, a new evoking technique should be developed for activating intermittent (resistive) faults at the chip level. It should be noted that the evoking process should be carried out simultaneously with detection; otherwise, the intermittent fault might remain undetected.

1.6 This Thesis

The significance of IRF detection in the electronic industry as one of the primary sources of NFFs has already been discussed. This section will discuss the motivation, resulting research questions, and original contributions of this thesis regarding IRF detection.

1.6.1 Motivation

The motivation behind this thesis is to detect IRFs at early stages before they grow and cause logic errors in a high-dependable system like a glide computer.

To evaluate our IRF detection design, IRF must first be modelled. Based on this model, an IRF generator can be created to conduct IRF injection experiments.

Furthermore, because the occurrence of IRFs is highly dependent on the environmental conditions under which the system operates, online monitoring techniques should be developed to monitor the system in the field. As a result, special embedded test instruments (ETIs) for IRF detection and online monitoring should be developed. Moreover, an insertion algorithm should be designed to find the best location inside the chip for inserting the ETIs.

In addition, fault localisation is essential regarding IRFs. Because physical defects cause IRFs, identifying the location of the fault is crucial to repair the system. The IEEE IJTAG standard [IEE14] can be used to facilitate fault localisation.

To reduce hardware costs for IRF detection at the board level, on-chip ETIs can be reused for detecting IRFs in PCBs. Furthermore, since IRFs occur randomly over time, periodic testing can increase the likelihood of IRF detection by re-testing the system regularly.

Conventional periodic tests are not fast enough to detect short occurrences of IRFs since they require several time-consuming phases, such as loading test patterns, fetching the test results, and comparing them with expected ones. Therefore, to increase the likelihood of IRF detection, a new testing method that provides fast, online, and at-speed testing without requiring test patterns should be developed.

One common problem with testing a system with IRFs is that these faults may remain inactive and undetected if no evoking technique is used. Therefore, to increase the probability of IRF detection, it is crucial to develop techniques to evoke and activate IRFs. Moreover, IRF evoking and monitoring should be used concurrently to speed up IRF detection.

1.6.2 Problem Statement and Research Questions

There is still no effective method for detecting IRFs at early stages. This brings us to the main problem statement addressed by this thesis: *How to detect IRFs in the early stages before they cause errors in a system?* Which can be further specified by the following research questions:

Q1. What is the physical origin of IRF, and how do they manifest themselves in practice?

Q2. How can IRFs be modelled at the board and chip levels? How to emulate (or simulate) IRFs to test IRF detection techniques? How to measure the sensitivity of a design against IRFs?

Q3. How to detect IRFs at the chip level? Is it possible to design an embedded instrument for IRF detection that is fully digital, scalable, and compatible with the IJTAG standard?

Q4. How can embedded instruments be located for efficient IRF detection within a design? How many embedded instruments are required, and where should these instruments be located?

Q5. How may IRFs be detected at the board level? Is it possible to use on-chip embedded instruments for IRF detection in PCBs?

Q6. Is it possible to accelerate IRF detection while testing? How can one evolve and activate IRFs at the chip level?

The aim of this thesis is to answer the above research questions.

1.6.3 Outline of the Thesis

This thesis is organised as follows:

- **Chapter 1 - Introduction:**

  No-Fault-Found (NFF) and the subclass of intermittent faults are introduced in this chapter. It provides preliminary introductions regarding modelling, detection and evoking of intermittent faults. More detailed introductions on intermittent resistive faults are presented in subsequent chapters.

- **Chapter 2 - Background and Related Work:**

  This chapter briefly provides the required general background and related work for this thesis. In addition to this background chapter, subject-specific background information and related work are included in the subsequent chapters.

- **Chapter 3 - IRF Measurements, Modelling, Simulation and Hardware Emulation:**

  The features of IRFs were extracted in this chapter by monitoring two major sources of IRFs in boards, namely cold-solder joints and loose connections, during thermal cycling experiments. Based on these observations, a representative model of IRFs has been formulated and presented. In addition, the IRF model has been used to design IRF generators for both transistor and board level fault injections. A software-based IRF generator has been developed to analyse the effect of IRFs at the transistor level. To evaluate the effects of IRFs at the board level, a hardware-based IRF generator has been built.

- **Chapter 4 - IRF Detection at Chip Level:**

  This chapter proposes two new embedded test instruments (ETIs) for IRF detection at the chip level. The ETIs are entirely digital, scalable, and IJTAG compatible. The effectiveness of ETIs has been investigated by simulation-based fault injections. It is shown that ETIs can differentiate between intermittent, ageing, and permanent faults.

  The second section of the chapter proposes an algorithm for locating the ideal spots in a design for IRF monitoring and ETI insertion on the chip. The algorithm maximises the IRF coverage for the most vulnerable parts of the circuit to IRFs, such as wire interconnections, vias and input pads.

  At the end of this chapter, a new periodic testing method for IRF detection at the chip level has been introduced. A Network-on-Chip infrastructure with a mesh topology was utilised as a case study.

- **Chapter 5 - IRF Detection at Board Level:**

  This chapter introduces an enhanced version of the IEEE 1149.4 standard to tackle IRF detection at the board level. It has been shown that the proposed method can detect IRFs in the interconnections with mixed-signals on boards. The on-chip ETIs have been reused for IRF detection at the board level. The reuse of our on-chip embedded test instruments for IRF detection at the board level was evaluated using hardware fault-injections. Moreover, the ability of IRF detection using a combination of periodic testing and the ETIs was verified by evoking actual IRFs caused by cold solder joints under thermal cycling.

- **Chapter 6 - Evoking IRFs at Chip Level:**

  This chapter proposes a thermal cycling technique to evoke IRFs in processors. A software procedure has been implemented to generate and conduct thermal cycling using customised workloads. Furthermore, the possibility of generating localised hotspots within a chip has been explored through an investigation involving two processors as case studies: the Plasma and Xentium processors.

- **Chapter 7 - Conclusions and Future Work:**

  In this chapter, general conclusions are given, then thesis contributions are summarised, and future work is discussed.

Chapter 2
Background and Related Work

Abstract– *This chapter provides the required background and related work for this thesis. The chapter is divided into three sections. First, studies on intermittent fault modelling and simulation are presented. Following that, research on intermittent fault detection is addressed. Finally, background on intermittent fault evoking is given.*

2.1 Intermittent Fault Modelling and Simulation

Field collected data and failure analysis results from avionic instruments [Kir04, Ste05, Jam03], and commercial electrical systems [Dav05a, Con08, Con07b, Con07a], clearly show that intermittent faults are a major cause for field returns in modern integrated circuits and boards.

There have been a few publications that have studied the effects of intermittent faults on micro-controllers [Gil12] and microprocessors [Pan12, Gar13, Gar14]. In [Gil12], the authors created fault models for intermittent faults at logic and RTL abstraction levels. Then, they injected intermittent faults in the VHDL model of a microcontroller to investigate their impact on the system behaviour. In their experiments, the so-called fault injection technique was used, and the buses were the focus of the investigation. The results showed that the intermittent faults had a significant impact on the system behaviour and that their fault models could capture the effects of these faults. The authors concluded that the fault injection technique is useful for studying intermittent fault effects on digital systems. The results of their experiment indicate that when multiple intermittent faults are injected, the percentage of failures ranges from 60 to 80 percent, depending on the workload.

In [Gar14], the authors studied the impact of intermittent faults on the behaviour of a reduced instruction set computing (RISC) microprocessor. They used VHDL-based fault injection to investigate the impact of intermittent faults. The authors in [Pan12] proposed a metric referred to as the intermittent vulnerability factor and computed these factors for register files and buffers considering three intermittent fault models: the intermittent stuck-at-1 and stuck-at-0 fault model, the intermittent open and short fault model, and the intermittent timing fault model. The intermittent vulnerability factor quantifies the likelihood that a fault will cause a system error. It is calculated by considering the number of faults that can occur in a given time period, the number of detected faults, and the number of corrected faults. The authors used this metric to compare the susceptibility of register files and buffers to intermittent faults. They discovered that register files are more susceptible to intermittent faults than buffers and that the intermittent stuck-at-1 and stuck-at-0 fault models are the most likely to result in errors. Overall, these studies demonstrate that intermittent faults can have a significant impact on the behaviour of microprocessors.

The above studies focused on modelling and simulating intermittent stuck-at-0, stuck-at-1, and open and short faults. Nevertheless, they did not present a model for intermittent resistive faults, which are among the primary causes of NFFs issues. In this thesis, we aim to fill this gap by introducing a model for intermittent resistive faults. Subsequently, we will develop a fault generator based on this model to assess the impact of IRFs in various designs. Furthermore, we will propose several design methodologies to detect the effects of intermittent resistive faults in circuits.

2.2 Intermittent Fault Detection at Chip Level

Intermittent (resistive) faults in a chip can increase path delays and, as a result, induce timing violations. Since we have not found previous work for intermittent fault detection at the chip level, the related work for path-delay fault detection have been provided in this section.

There are several approaches toward measuring path delay by using dedicated measurement circuits such as tunable replica circuits (TRC), ring oscillators (RO) and time-to-digital converters (TDC). They will be briefly discussed subsequently.

A TRC is a method for detecting (timing) errors due to voltage and delay degradation. Process variations and ageing similarly affect replicas in the critical path. However, the random component of process variations will result in differences between the TRC and the actual critical path [Bow09].

The ring-oscillator-based architecture can be used more efficiently for path delay measurements by making the path-under-test a part of the oscillator. The ring oscillator frequency can be monitored continuously, and any degradation in frequency with time can be attributed to device ageing effects [Nan08]. Test structures based on ring oscillators can be used to measure the ageing degradation effects in digital circuits [Kim15, Hsi15].

A TDC measures the time interval between two transitional signals. It is typically constructed from registers and delay elements and, like ROs, can be used to measure the effectiveness of the process, temperature and voltage variation and ageing degradation. However, as the TDC is an on-chip circuit, it suffers from process variation and ageing effects. Hence, it needs self-calibration [Kat15].

However, none of the aforementioned methods can be used for IRF detection. As IRFs occur randomly in time, an embedded test instrument should continuously monitor path activities to detect IRF occurrences. TRC is not applicable for IRF detection as an IRF can occur in every interconnection wire independently. As a result, an IRF in a replica does not mean an IRF in the actual path. Similarly, an RO needs a dedicated path, which makes it unsuitable for IRF detection. The usage of TDC for IRF detection is not cost-effective in terms of area and power consumption.

However, another approach to measure the timing violations is to use an embedded monitor accompanied by a guard-band. Most of the ageing faults increase the delay of transistors and connections. In a synchronous system, a delay increment can result in a timing failure. Timing failures occur when the delayed data does not meet the flip-flop setup requirement and has a late transition near the clock edge. Hence, on-line delay (or slack) monitoring in an integrated circuit is a promising method for measuring the ageing of synchronous circuits [Bow09, Rah14].

The timing slack or guard-band is defined as the delay between the data arrival time and the active edge of the clock minus the flip-flop setup time. Generally, the guard-band assignment is part of the pre-silicon design phase. A guard-band for a monitor is determined based on the slack of the target path and the clock frequency of the system. This chapter does not focus on the time-windows (guard-band) generation.

According to recent studies [Lai14, Sad15], slack monitoring techniques have proven effective in detecting timing errors, including those arising from ageing, process variations, and voltage fluctuations. In [Rah14], a timing slack monitoring methodology of inserting monitors at both the end-node and intermediate-node of the data path is presented. In [Sad15], the authors presented a digital slack monitor which can measure the slack of a selected path. However, their sensor can detect either rise or fall transitions, requiring re-initialisation for each slack monitoring procedure.

The methods mentioned above cannot be used for intermittent fault detection because they need pre-initialisation and calibration; moreover, they do not provide continuous on-line monitoring of the transmission line.

Numerous studies have investigated the end-nodes selection approach for on-line ageing and process variation detection. In [Gom18], a critical-paths selection algorithm has been proposed for dynamic frequency scaling under BTI-ageing and process variations effects. A path selection flow for on-line ageing monitoring in reconfigurable architecture has been presented in [Ebr16c]. Their algorithm first selects the ageing-prone path based on path delay, temperature, duty cycle, and switching activity. Then, the selected paths get pruned based on fan-out and the physical location of the path end-nodes.

Several studies have focused on ageing-monitor insertion at internal nodes of the circuit [Lai14, Liu15, Sad18]. The underlying rationale behind these efforts is to detect timing violations at the earliest possible stage, before they become masked or lead to a failure. The proposed method in [Lai14] inserts slack monitors like probes at a selected set of nodes, including internal nodes of critical paths.

In [Sad18], the authors have proposed a procedure for selecting the monitoring nodes only among the internal nodes of the circuit. This increases the accuracy of age-monitoring mechanisms and decreases the number of nodes, resulting in low hardware overhead.

The techniques mentioned above are based on detecting delay violations caused by ageing or process variation based on monitoring the critical path(s), near-critical path(s) or a representative of them. The measurement of path delay, temperature and switching activity plays the main role in the ageing estimation. Since the critical and near-critical paths are more prone to ageing effects, they are good candidates for ageing monitoring. Unlike the ageing effects, IRF can happen randomly in any circuit path and may result in timing violations at the path end-node. Therefore, the critical and near-critical paths are not necessarily the best candidates for IRF detection.

In this section, we have discussed the related work for intermittent fault detection at the chip level. The existing approaches for path-delay fault detection, such as tunable replica circuits (TRC), ring oscillators (RO) and time-to-digital converters (TDC), were discussed. However, it was discussed that these approaches are unsuitable for intermittent fault detection. In this thesis, we have presented our embedded test instruments for online monitoring and detection of IRFs. Moreover, an algorithm is proposed to find the best locations in a design for IRF monitoring and ETI insertion in the chip. The objective of this algorithm is to maximise the IRF coverage of a circuit.

2.3 Intermittent Fault Detection at Board Level

Unstable and defected interconnections are the primary source of IRFs at the board level. Several methods have been used traditionally for testing interconnections at the board level, such as DC resistance measurements [Pan14], radio-frequency

(RF) impedance analysis [Kwo11], [Loe12], Built-in Self-Test [Hof10] and analogue neural network technology [Ste08]. DC resistance measurements [Pan14] is a valuable method to detect stuck-at-short, stuck-at-open and bridge faults in interconnections at test mode. However, for detecting small impedance changes in interconnections, RF analysis is more effective [Kwo11].

RF analysis has been widely used for detecting discontinuities in wire and cable connections. This measurement technique can be used in both the time and frequency domains. Time-domain reflectometry transmits an RF signal through a wire and observes the reflections. If there is a discontinuity in the wire, an impedance mismatch will be observed. Another reflectometry technique is frequency-domain reflectometry (FDR). FDR can detect a fault location in a transmission line by measuring frequency, amplitude and phase changes. FDR has been utilised in [Loe12] to monitor connector degradation.

Another technique to test interconnection reliability at the board level is based on BIST. A method based on BIST, referred to as SJ BIST, was proposed in [Hof10]. It can detect faults induced by solder-joint fractures in the input/output pins of field-programmable gate array (FPGA) devices.

The aforementioned techniques are powerful methods for testing discontinuity or impedance degradations in interconnections and wires in test mode. Because of the nondeterministic behaviour of intermittent faults, the probability that an IRF is activated while in test mode is very low. Therefore, the conventional test methods are implausible in detecting these faults. Furthermore, even if one can evoke an IRF during the test phase, very accurate equipment at high frequencies such as data logger[Pan14], vector network analyser [Kwo11, Loe12], and analogue neural network [Ste08] are needed to detect brief IRFs. Therefore, since IRF may remain undetected during a test phase, the in-situ on-line monitoring technique should be used to detect such faults when they become active while a system is in operational mode.

In-situ on-line monitoring is widely used at the chip level for timing-fault detection [Das09, Lai14, Sad15]. Timing violations along a data path can occur due to process, voltage, and temperature variations [Uns06]. In addition, ageing and IRF can cause timing violations in the data path. In-situ on-line monitoring techniques have been used for timing error detection and correction at the critical paths [Das09], timing slack measurement [Sad15], adaptive voltage scaling [Sha17] and ageing detection [Sal15]. This thesis shows that with some modifications, the in-situ on-line monitoring techniques can be used for IRF detection at the chip level as well.

The main difference is that IRFs can induce random timing violations. The IRF embedded test instrument should be capable of detecting such unpredictable behavior. Another difference is that they can occur randomly at any location, necessitating a fault localization mechanism for effective IRF detection. In addition, RFs may reappear at the same location as previously observed. To manage this aspect, a fault-management software is essential to monitor the occurrence rate and behavior of faults. In conclusion, IRFs are a major source of reliability issues at the board level. Conventional test methods are inadequate for detecting these faults. However, in-situ on-line monitoring techniques offer a viable solution for detecting IRFs while a system is operational. This thesis has proposed two IRF-ETIs and a fault-management framework with fault localisation and classification capabilities, to effectively detect and manage IRFs. The proposed techniques have been evaluated using fault injection techniques, and the results show that the proposed techniques are successful in detecting and managing IRFs.

2.4 Intermittent Fault Evoking

A product is labelled as NFF if it has been reported to have faulty behaviours in the field. However, when the product is returned to the laboratory and tested, no fault can be found in the product [Kha14, Ben02]. One reason is that the fault(s) can not be revoked during the test time. Knowing that intermittent (resistive) faults are one of the leading causes of NFFs, evoking these faults is critical for reducing test and maintenance costs induced by NFFs.

Intermittent faults should be evoked in order to be detected during the testing period. Otherwise, they remain dormant and undetected while testing, but become active when the product with intermittent fault is returned to the field. The probability that an intermittent fault becomes activated without extra stimuli while testing is very low. Therefore, external stimuli, such as temperature changes, should be used in a controlled environment to increase the likelihood of intermittent fault detection.

A popular method of accelerating and identifying early-life failures is the burn-in test [Sem03]. Burn-in should be completed at a minimal cost and reasonable amount of time. The typical approach for carrying out the burn-in test is using an oven. However, in [Agh15], an algorithm is proposed to schedule a structural test to increase temperature locally and create a temperature gradient in a chip to enhance the burn-in procedure. Alternatively, in this thesis, we investigate the possibility of employing functional workloads to control temperature changes in processors. The proposed approach uses a custom workload to induce thermal cycling in the processor. The workload will be designed to create a temperature gradient in the processor, which will be used to evoke intermittent faults.

2.5 Summary

In this chapter, a brief discussion and background on intermittent (resistive) faults, modelling, simulation, detection, and evoking are presented. In addition to this background chapter, subject-specific background information and related work are included in the subsequent chapters.

There is some research on the modelling and simulation of intermittent stuck-at-0, stuck-at-1, and open and short faults. However, there is no model for intermittent resistive faults. Therefore, developing a model for these faults is crucial to evaluate design vulnerability against IRFs.

Previous work and background on resistive fault detection in PCBs have been presented. Conventional testing methods have been discussed to be ineffective in detecting IRFs. As a result, a new testing method for detecting IRFs in PBCs should be developed.

IRFs can cause path delays and induce timing violations in chips. Although there is some previous work regarding delay-fault detection, these methods could not detect IRFs. Therefore, new testing methods and embedded test instruments should be designed for IRF detection.

Conventional periodic tests are not fast enough to detect short occurrences of IRFs since they require several time-consuming phases. As a result, a new testing method should be developed that provides fast, online and at-speed testing without requiring test patterns to increase the probability of IRF detection.

The importance of evoking IRF at the chip level has been discussed. However, there is a lack of an efficient method to invoke IRF in chips. This thesis addresses this issue as well.

Chapter 3
Intermittent Resistive Fault Measurements, Modelling, Simulation and Hardware Emulation

Abstract– *In order to be able to tackle intermittent resistive faults (IRFs), it is necessary to know the behaviour of these faults and their effects on electrical systems. In this chapter, first, the characteristics of IRFs have been extracted by observing cold-solder joints and loose connectors under thermal cycling experiments. Then, based on these measurements, a representative model of IRFs has been introduced. Moreover, the IRF model has been employed to develop an IRF generator. The proposed IRF generator allows the study of IRF effects on digital systems. In order to analyse the effect of IRFs at the transistor level, a simulation-based IRF generator has been developed. A hardware-based IRF generator has been designed to evaluate IRFs at the board level.*

The simulation-based IRF generator has been utilised for fault injection at the transistor level. These initial simulations have been used as a first evaluation and justification of IRF influence on analogue as well as digital circuits. Then, IRFs have been physically generated and injected into transmission lines at the board level using a hardware-based IRF generator. In this way, the effects of IRFs on actual circuits have been measured at the board level.

In addition, the effectiveness of the proposed IRF generator has been validated by experimental results. The IRF generator allows the effectiveness of newly developed IRF detection methods to be evaluated. Besides,

Parts of this chapter were published in:

- [Ebr16a] H. Ebrahimi and H. G. Kerkhoff, "Testing for resistive faults in CMOS integrated systems," in IEEE Euromicro Conference on Digital System Design (DSD), pp. 703–707, 2016.
- [Ebr18] H. Ebrahimi and H. G. Kerkhoff, "Intermittent resistive fault detection at board level," in IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), pp. 135–140, 2018.

designers can utilise it to evaluate the sensitivity of new designs with respect to IRFs.

The comparison of the simulation-based and hardware-based IRF injection reveals that the IRF injection procedure can be performed about seven orders of magnitude faster using a hardware-based IRF generator.

The measurement results show that cold-solder joints and loose connectors can cause IRFs. Moreover, IRFs can evolve under thermal cycling and eventually become permanent faults. The experimental results show that IRFs can lead to intermittent delay faults if they occur in transmission lines or data paths. For example, in the case of the UART protocol, they can cause single, multiple and frame errors in transmitted data.

3.1 Intermittent Resistive Fault Measurements

At the board level, intermittent resistive faults are mainly due to interconnecting malfunctions such as solder joints, traces, component leads, and connectors [Qi08, Tho02]. Mechanisms such as corrosion [Par08], fatigue [Lee14, Pan11], and mechanical stress [Sam18] threaten the reliability of interconnections in boards.

Mechanical and temperature stress are the most common mechanisms that researchers have used to study the fatigue life of interconnections at the board level [Sur20, Jia19, Loe12]. The temperature stress has been utilised to analyse the reliability of solder joints [Pan01]. Mechanical stress experiments have been carried out to estimate the remaining life of solder joints [Kwo15, Sam18]. In [Liu16, Jia19], authors investigated the combined effects of electromigration and thermal stress on the degradation of solder joints. Moreover, the failure of solder joints has been experimented with using combined mechanical and temperature stresses in [Zha15].

Although the fatigue life of interconnections in boards has been studied widely, there is still a lack of information about the electrical characteristics of intermittent resistive faults at the board level. To address this issue, we have measured the resistive behaviour of two main physical causes of intermittent faults in boards, i.e. cold solder joints and loose connectors [Qi08, And14].

In this chapter, the relationship between thermal cycling and intermittent resistive faults has been investigated. Thermal cycling has been carried out to evoke the intermittent behaviour of these interconnections. During thermal cycling, the ambient temperature increases and decreases periodically [Pan14]. The thermal cycling exerts temperature stress on the interconnections under test. Several cold solder joints and loose connectors have been implemented for the experiments, and their behaviour under thermal cycling has been investigated.

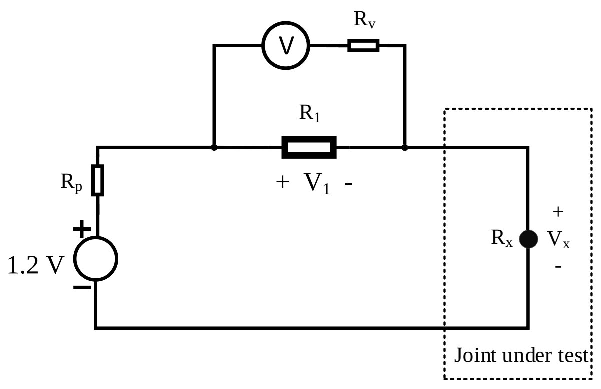


Figure 3.1: Setup to measure a cold solder or a loose connector joint anomaly under thermal cycling.

Two alternative approaches can be used to evaluate the connectivity of an interconnection and measure its resistance changes according to environmental stress. One method is direct current (DC) resistance measurements [Pan14], and another is radio-frequency (RF) impedance analysis [Kwo11, Loe12]. The DC measurement is straightforward. It can be carried out with a DC voltage source and a digital voltmeter. Whereas the RF impedance analysis requires expensive signal generators and data loggers running at high frequencies [Kwo11].

In this chapter, the DC measurement method has been used to monitor and measure resistance changes of solder joints and loose connectors [And14] under thermal cycling. Thermal cycling has been applied to evoke the intermittent behaviour of cold solder joints. To prevent the voltmeter from being damaged by high temperatures, the solder joints were measured indirectly by measuring the voltage of an accurate constant resistor R_1 connected in series with the connection under test. To be able to measure resistance changes with small magnitudes, the constant resistor R_1 was selected to have a small resistance magnitude. For this purpose, a resistance of $1 \pm 0.01\ \Omega$ was used.

Our measurement setup is shown in Figure 3.1. In this circuit, a resistor with a constant resistance R_1 is in series with the joint under test. The circuit is connected to a constant DC supply voltage of 1.2 Volt and a small internal resistance of R_{pow}. A digital voltmeter (with input resistance $R_v = 1M\Omega$) is connected in parallel with R_1. By measuring V_1 using a digital multimeter and applying Ohm's law, the value of the resistance for the cold solder joint can be calculated from the following Equation:

$$R_x = \frac{1.2 * R_1}{V_1} - R_{pow} - R_1 \tag{3.1}$$

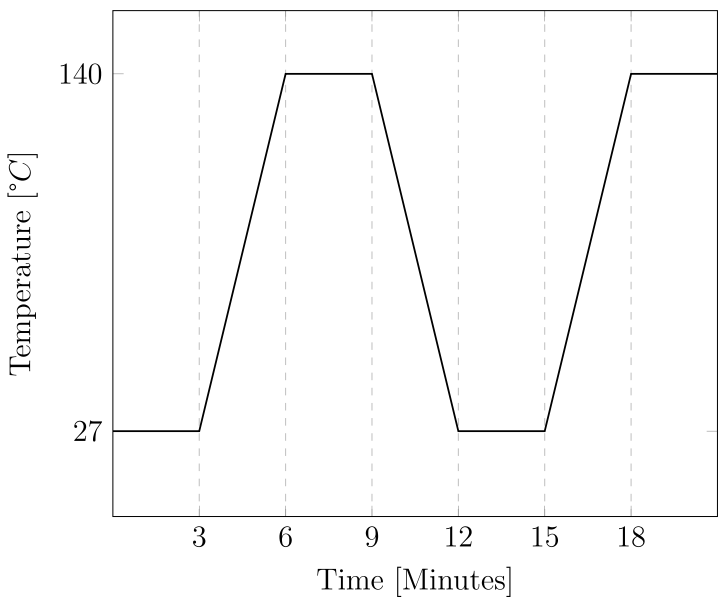


Figure 3.2: Thermal cycling profile with 3 minutes dwelling time.

During the measurement, the value of V_1 was continuously measured using a signal digitiser with the speed of 100 mega samples per second. This sample rate could detect short changes as small as 10 ns. The calculation and analysis of the IRFs were continuously executed using a Python script.

In total, 50 solder joints were made for the experiment. From 50 solder joints, 40 were made to be cold-solder joints. To differentiate IRF effects from noise, 10 solder joints were accurately created with a sufficiently high soldering temperature.

One of the most challenging aspects of these experiments was learning to make a cold solder joint. Cold solder junctions are undesirable in industry. They are caused by applying an insufficient (too low) temperature to melt the solder during soldering [Sun19]. In order to make a cold solder joint in a laboratory, the soldering temperature should be slightly less than the melting point of the solder. If the temperature is too low, the connection will not be created, and no solder joint will form. On the other hand, if the temperature is too high, it will lead to a proper solder joint, which is not the purpose of this experiment. The solder that has been chosen for our experiments is one of the most common solders used in the industry, Sn60Pb40 (60 % tin and 40 % lead), with a melting point of about 180 °C.

Figure 3.2 shows the thermal cycling profile, which was applied to the circuit under test. The profile shows that the circuit was heated up from 27 $°C$ to 140 $°C$ and repeatedly cooled down to 27 $°C$ during each cycle. It should be noted that the highest temperature (140$°C$) was selected to be lower than the melting point of the solder (180 $°C$). Otherwise, it is possible that a cold solder becomes a proper solder

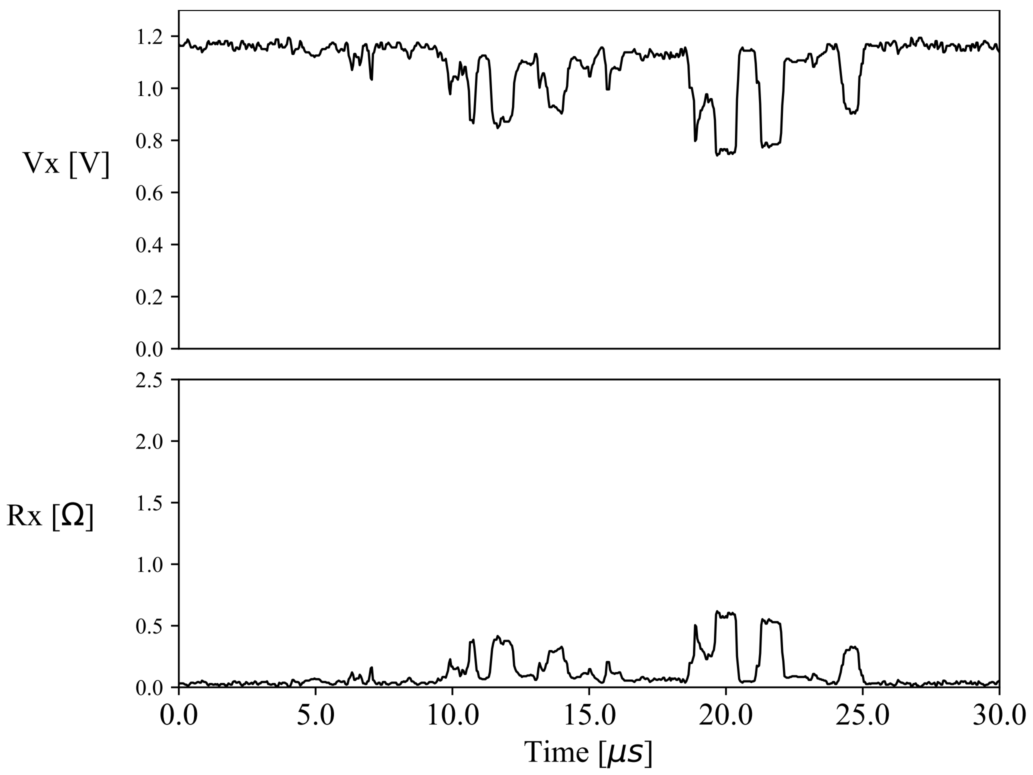


Figure 3.3: An example of the voltage and the resistance changes for a cold solder after 20 cycles of temperature cycling.

joint and hence hinders the observation of the intermittent behaviour of the cold solder joint.

No changes were detected in the cold solder joints at the beginning of the temperature cycling procedure. However, the voltage digitiser detected some voltage changes after several temperature cycles. The voltage changes revealed resistance changes in the cold solder joints. This means that the temperature changes can be used to evoke the intermittent behaviour of a cold solder. An intermittent resistive fault was induced in the circuit when a cold solder was triggered. Figure 3.3 shows the voltage and the resistance of a cold solder joint at the early stage of the temperature cycling (after 20 thermal cycles). The voltage and resistance values are calculated by applying Ohm's law to the measured voltage (V_1) of the constant resistor (R_1). As can be seen, the resistance can be as small as a few Ohms with an occurrence duration of a few nanoseconds. In some cases, the duration increases to microseconds. The behaviour of the same cold solder after several hours of temperature cycling (after 45 thermal cycles) is shown in Figure 3.4. The comparison between Figures 3.3 and 3.4 indicates that the intermittent resistive fault caused by the cold solder becomes more severe in the presence of temperature cycling. The resistance value of the cold solder increases almost by a factor of two, with an active time of a few microseconds.

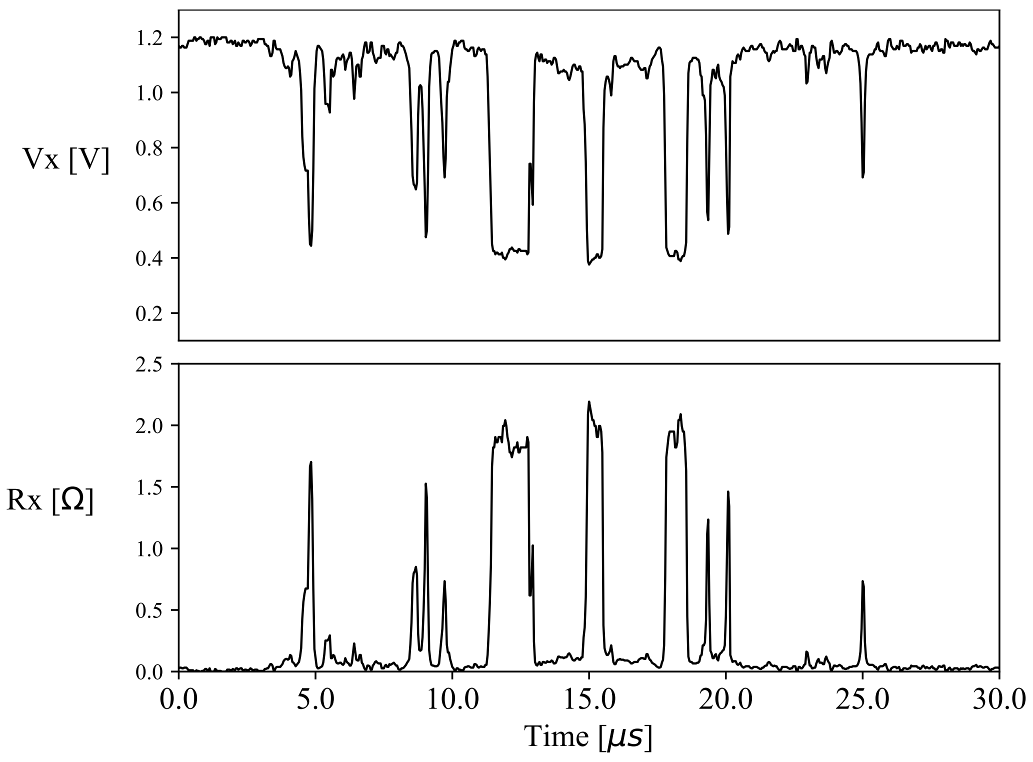


Figure 3.4: An example of the voltage and resulting resistance for a cold solder joint after 45 cycles of temperature cycling.

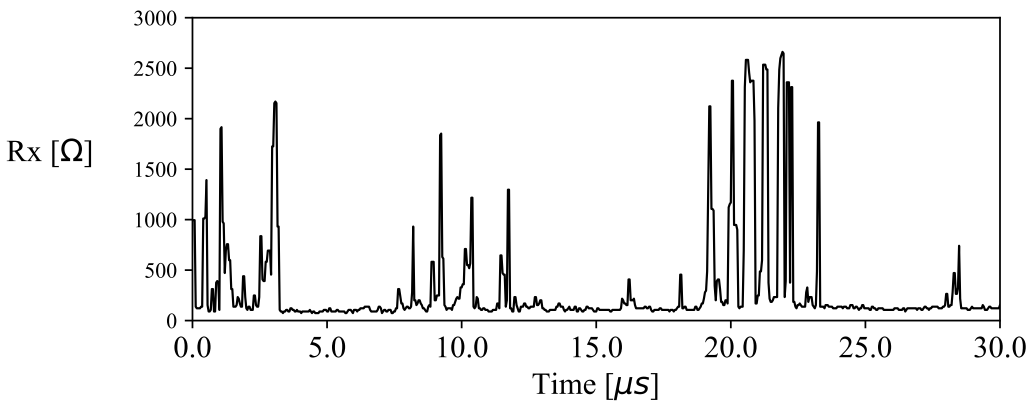


Figure 3.5: An example of measured resistance changes for a loose connector after 49 cycles of temperature cycling.

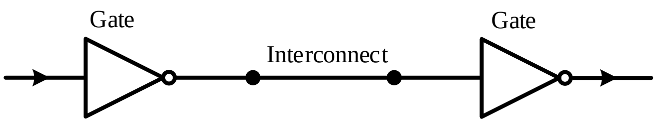
Further measurements on cold solder joints show that the existing intermittent resistive faults can either increase or decrease in intensity when the temperature cycling stops, but they do not disappear completely. Besides, if the temperature surpasses the melting point of the solder joint, in that case, the cold solder joint might turn into a proper solder joint, and subsequently, the intermittent resistive fault would disappear.

In total, 100 thermal cycles have been carried out on cold solder joints. Out of 40 cold solder joints, 11 have been diagnosed with IRF behaviour during the thermal cycles. Out of these 11 solder joints, one became an open circuit (permanent fault) after 85 thermal cycles.

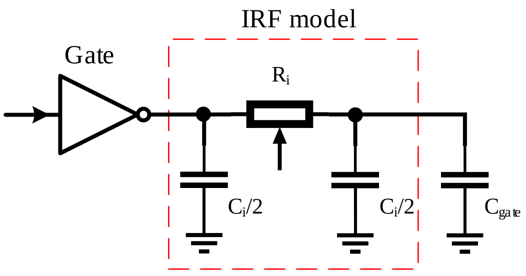
A similar thermal-cycling procedure was also applied for loose connectors. A loose connector is a male-female terminal connector that is mechanically connected, but the connection is weak and loose. The experiments have been carried out on 20 pairs of male-female terminal connectors. The experiments showed that loose connectors induce intermittent resistive faults in the circuit under thermal stress. An example of measured resistance change for a loose connector after 49 cycles of temperature cycling is shown in Figure 3.5. This figure shows that loose connectors can create more significant resistance changes than a cold-solder joint.

Since loose connectors are mechanically loose, they are sensitive to vibrations. Some preliminary experiments have been carried out to investigate the effect of vibration on loose connectors. The result revealed that, like thermal cycling, the vibration can be utilised to evoke IRFs in loose connectors.

The experimental results have shown that thermal cycling can be employed to evoke and activate IRFs at the board level. IRFs occur shortly with random low-level resistances. Moreover, IRFs can become more severe, i.e. larger occurrence times and bigger resistance values, by being longer exposed to thermal stress. Although



(a) A circuit representing an interconnection between logic gates.



(b) Substitution of the interconnection with its IRF model.

Figure 3.6: Representation of an interconnection with its equivalent IRF model.

the occurrence rate of IRFs was measured as low, around 1 %, yet it was noticed that they could cause a failure. Summarising, it was observed that the cold solder joints and the loose connectors have the main characteristics of intermittent resistive faults. They occur randomly in time. If they occurred at a location, they may pop up again in that exact location later. Moreover, eventually, they may evolve into a permanent open fault.

3.2 Intermittent Resistive Fault Generation

A simple model was developed based on the above experiments to represent an IRF in an interconnection. The IRF model is shown in Figure 3.6. Figure 3.6(a) depicts a selected interconnection in a data path. In this model, an interconnection is represented with its π model [Nek92]. Figure 3.6(b) shows how a π model can be employed as an IRF model using a random value R_i resistor instead of a fixed one. The C_{gate} represents the input capacitance of the next gate. In this model, the capacitance of the interconnection (C_i) can also be variable. However, it is assumed that the capacitance has a fixed value for simplicity.

Based on our IRF model, a procedure has been developed to generate IRFs at the transistor and board levels. Note that no measurements of IRFs have been carried out regarding the transistor level. Hence, it has been assumed that an IRF behaves

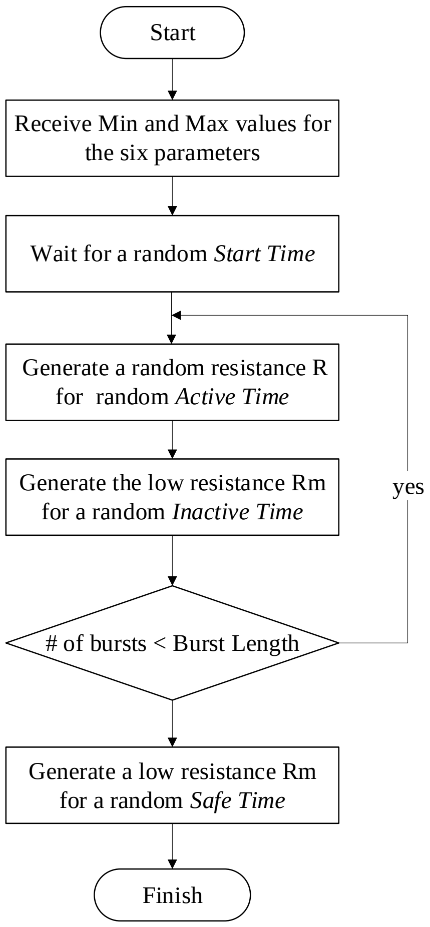

Start
Receive Min and Max values for the six parameters
Wait for a random *Start Time*
Generate a random resistance R for random *Active Time*
Generate the low resistance Rm for a random *Inactive Time*
yes
# of bursts < Burst Length
Generate a low resistance Rm for a random *Safe Time*
Finish

Figure 3.7: Flow chart of our intermittent resistive faults generator.

similarly at the chip and board levels.

A *simulation-based* IRF generator has been developed to study IRF effects at the transistor level. In contrast, a *hardware-based* one has been designed and implemented for the board level. This so-called IRF generator allows designers to study the vulnerability of their designs with respect to IRFs. The flow chart of our IRF generation procedure is presented in Figure 3.7.

The IRF generator can be customised by assigning six parameters. These parameters can be set by determining their minimum and maximum possible values. Besides, any (random) distribution, such as a uniform or Gaussian distribution, can be chosen according to specific requirements.

The IRF generator provides the ability to generate a burst (sequence) of resistance changes for the fault-injection procedure. In a burst, each resistive pulse has a random value *R*, which is active during a random activation time (*Active Time*). After each pulse, an inactivation time (*Inactive Time*) is randomly generated in which a fault-free situation exists *Rm*. In the case that the burst length is designed to be larger than 1, the same procedure will be followed again until the required number of pulses is equal to the initialised burst length being produced. The IRF generation procedure is completed by generating a fault-free situation at the end. In the following, this generator has been used to generate IRFs for simulation-based and hardware-based fault injection at the transistor as well as board level.

Fault injection is a valuable method to study the behaviour of a system under the influence of faults. Fault injection can be carried out at several levels. In this chapter, fault injection techniques have been used to study the behaviour of a given system regarding IRFs at the transistor as well as board levels.

The technique of the so-called *saboteur* [Gil15] has been used for fault injection. It means a module referred to as *saboteur* is placed in every interconnection which was selected for IRF injection. The saboteur is inactive during fault-free operation, but it injects a random IRF in the selected interconnection when it becomes active.

The IRF injection at the transistor level and subsequently at the board level are presented in the following.

3.3 IRF Evaluation at Transistor Level

Interconnections on-chip are susceptible to resistive faults caused by corrosion [Kod05], electromigration [Che15], and manufacturing defects in 2D [Mon02, Li05] as well as 3D chips [Yan10, Jun12]. Since these faults have physical roots, they can become activated or deactivated by variations in frequency, voltage, and operating temperature [Con08]. Due to this intermittent behaviour, these faults are referred to as intermittent resistive faults (IRFs). IRFs can affect digital as well as analogue electrical chip systems. The effects of IRFs on analogue systems have been studied in [Wan14].

3.3.1 The Effect of IRFs on Circuit Timing

In this chapter, the influence of IRFs on the timing characteristics of digital circuits has been studied. For this purpose, an accurate transistor-level simulation-based fault injection has been used. Simulations have been carried out with the Spectre simulator within the Cadence Virtuoso platform [Cad14].

A basic full-adder from the Nangate 45 nm CMOS library [Nan08] has been used

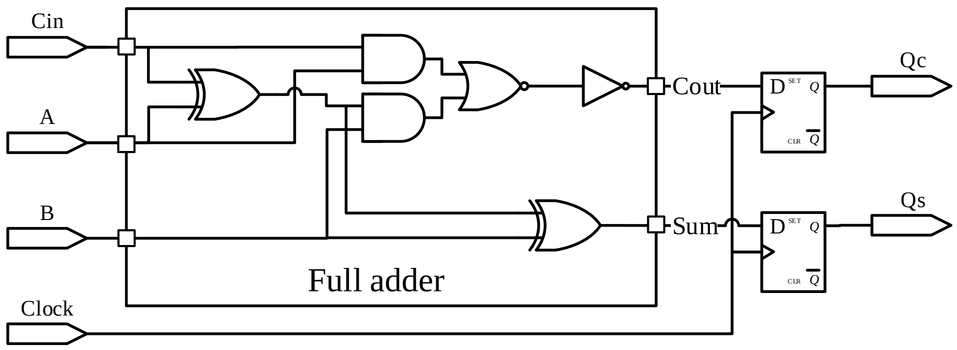


Figure 3.8: Logic scheme of a full adder using the Nangate 45 nm CMOS technology library [Nan08].

for demonstration purposes. The outputs of the full-adder have been latched by two D-flip-flops to capture the effect of IRFs on the end-point of a data path. The logic scheme is depicted in Figure 3.8. A *transistor-level* implementation of the circuit has been used for simulation. The IRF injection has been applied to the data inputs and other selected interconnections.

The result of injecting an IRF into one of the inputs of the full adder, e.g., the carry input *Cin*, is shown in Figure 3.9.

In this figure, the shown signals are explained as follows. The clock of the flip-flops is represented by the signal *Clock*. The inputs are *A, B, Cin* and the outputs *Sum, Qs, Cout*, and *Qc*, respectively. The *Cin** signal is the signal *Cin* distorted by the injected IRF. The injected IRF signal is shown at the bottom of Figure 3.9. It depicts an *IRF* signal with different resistance values of 51 kΩ, 75 kΩ, 64 kΩ and 36 kΩ at different times that have been injected in Cin during the simulation.

As can be seen in Figure 3.9, the input *Cin* has been disturbed by the injected IRF (see *Cin**). The injected IRF has affected the rise and fall times of the signal *Cin**. The IRF causes the input signal *Cin** to rise and fall slowly due to the increased RC delay. The rise and fall times of the input signal are proportional to the injected resistances of the IRF. However, only in one case, if the IRF has 75 kΩ resistance, the distortion has also caused an incorrect logic output *Qs* being captured in the flip-flops at the end of the *Sum* output (shown by a red cross). From this simulation and many similar ones, it can be concluded that IRFs can cause intermittent delay faults in data paths. In some cases, they can result in logic errors.

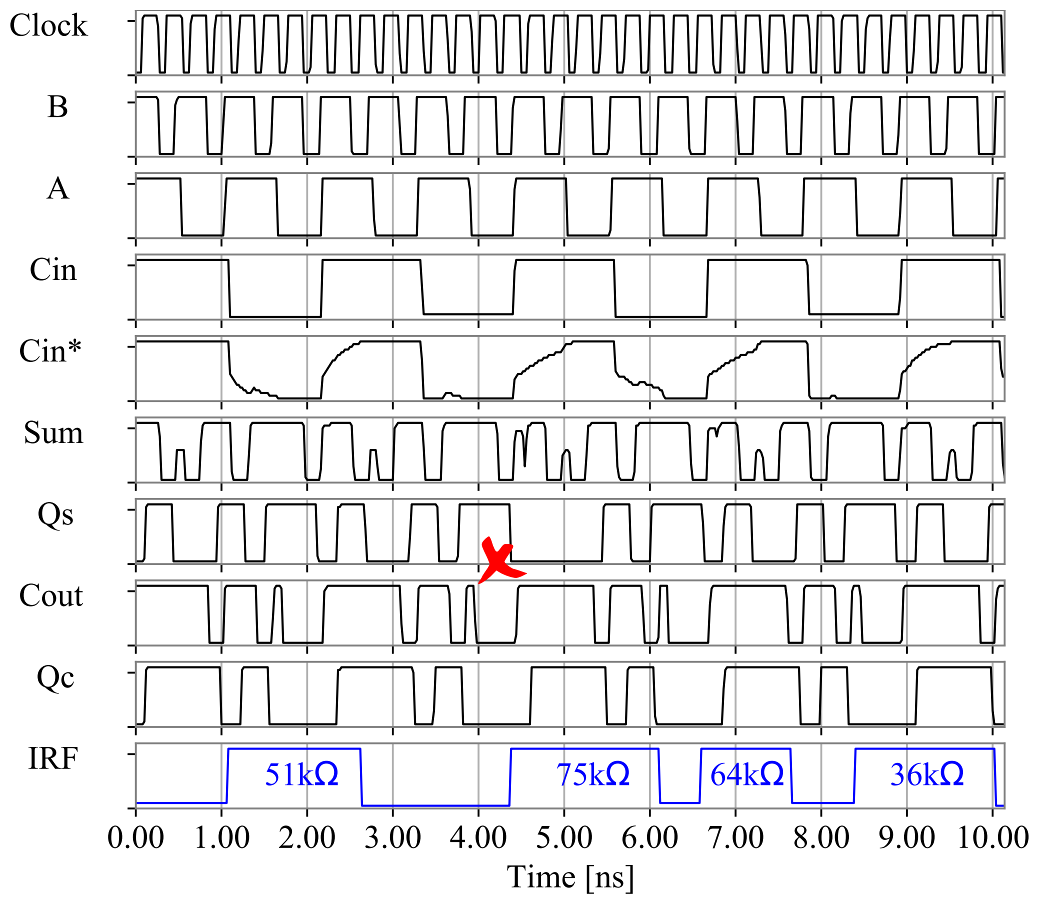


Figure 3.9: Simulation-based IRF injection in the *Cin* input of the full adder. The red cross indicates a logic error captured by the flip-flop at the Sum output *Qs*.

3.3.2 The Effect of IRFs on the Dynamic Supply Current

The dynamic supply current (I_{DDT}) measurement has been used in the past to detect open and delay faults in circuits [Ish01]. I_{DDT} is a transient power supply current that flows into a circuit via the V_{DD} pin in the transient state of the circuit. In this section, the effects of IRFs on I_{DDT} have been investigated. As depicted in Figure 3.10, IRFs can also cause disturbances in I_{DDT}. The dashed lines indicate the expected voltage and current for the signal in the fault-free case. The shaded area indicates the difference between the I_{DDT} in the fault-free (I_{DDT}) and faulty case (I_{DDT}*). It shows the average of I_{DDT} has increased by about 220 μA in the case of an IRF occurrence. However, the maximum and the minimum of the I_{DDT} have remained unchanged. Therefore, the average of I_{DDT} per clock cycle can be utilised to investigate the relationship between an I_{DDT} disturbance and a logic error.

The average I_{DDT} per clock cycle has been calculated from simulations for fault-free and faulty situations. Changes in I_{DDT} values due to an IRF applied to *Cin* are shown in Figure 3.11. It shows a burst of 18 resistance changes ranging from 1 $k\Omega$

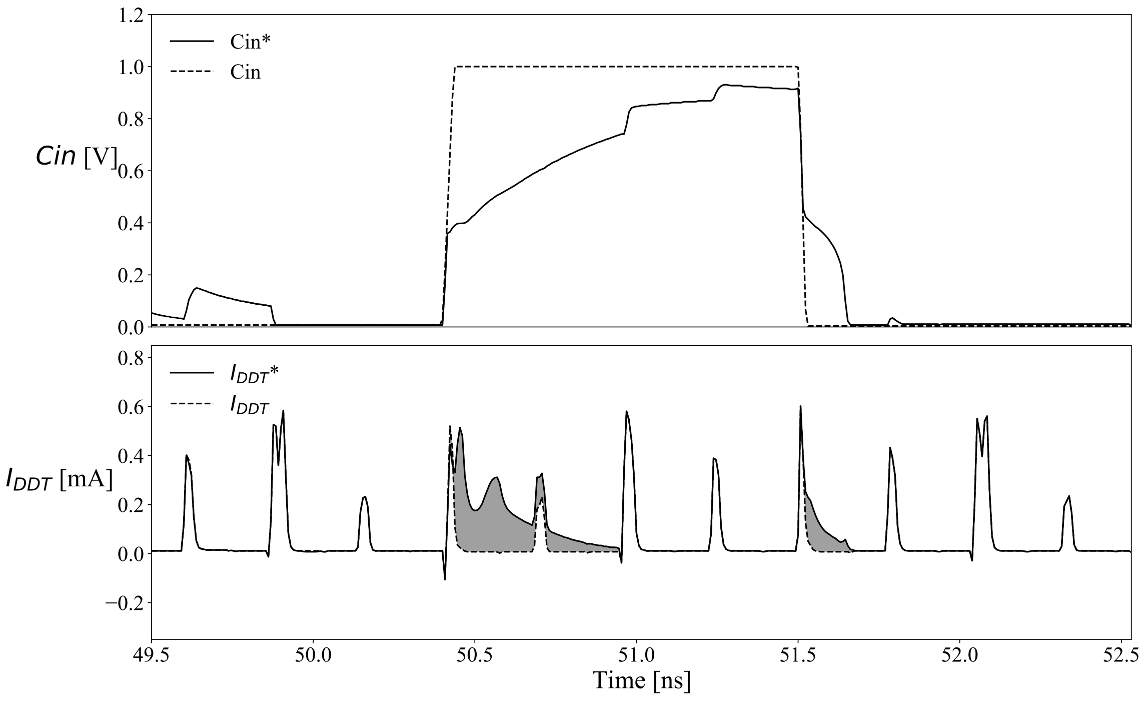


Figure 3.10: Detail of the simulation of a full adder under the influence of an IRF in the input Cin.

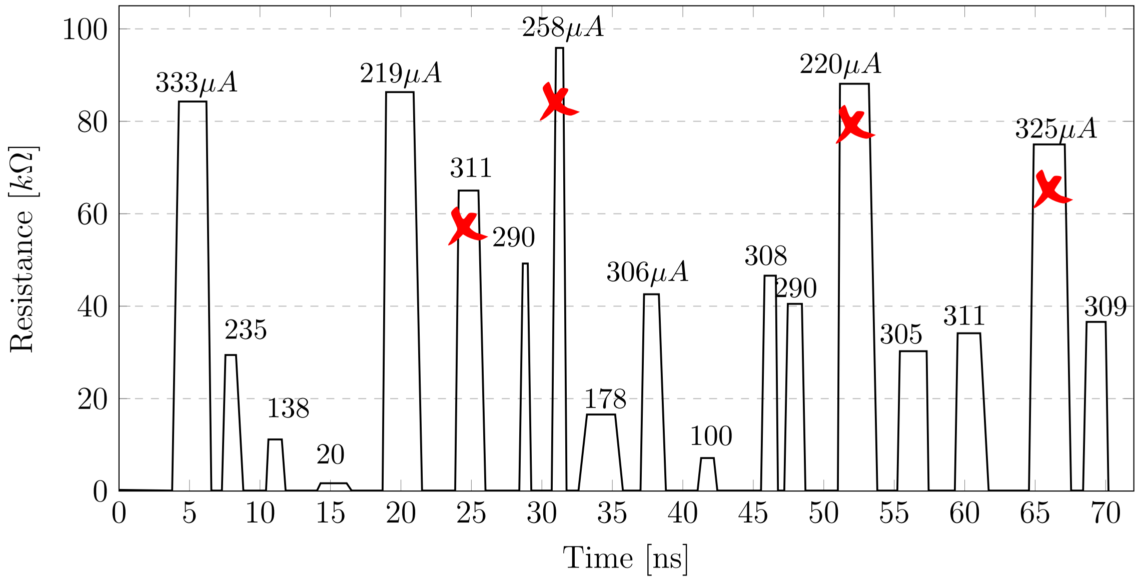


Figure 3.11: Relation between the injected IRF in the input Cin and the increment of the average amount of I_{DDT} current over a clock cycle.

to 100 $k\Omega$ during 70 ns.

The red crosses indicate if a resistive fault has caused a logic error at the output

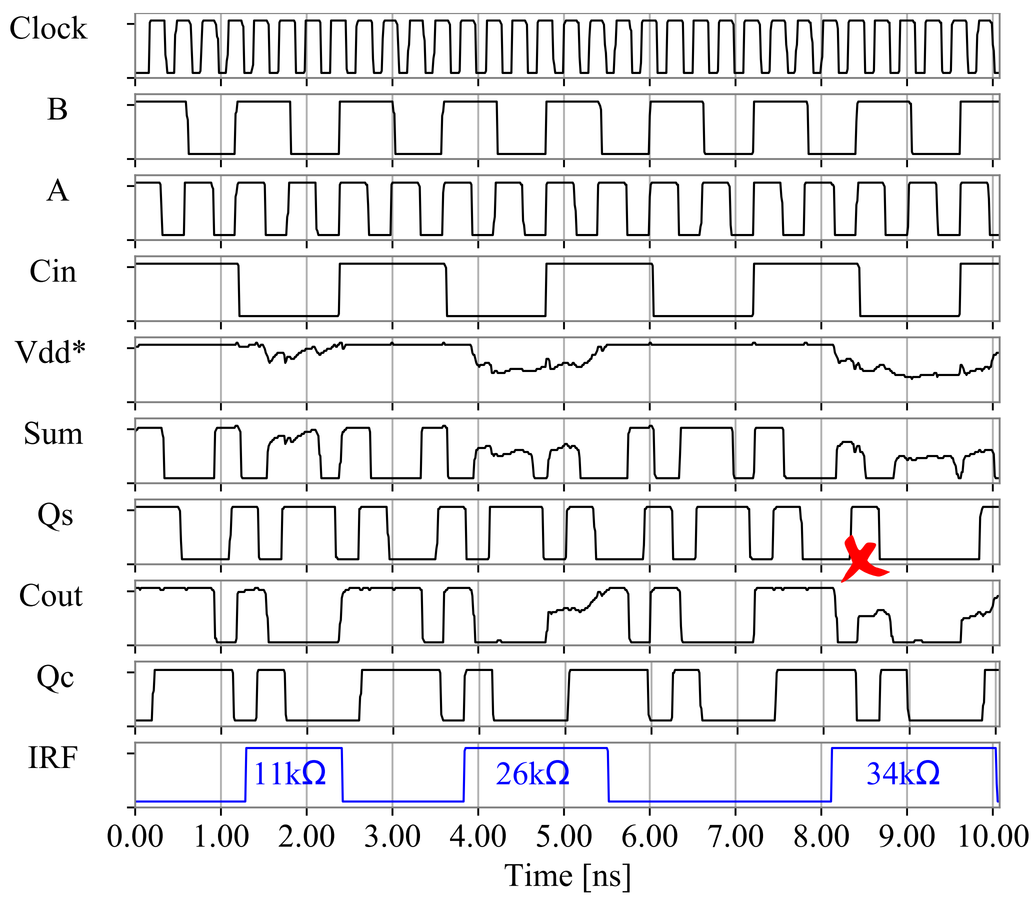


Figure 3.12: The result of IRF injection in the V_{DD} line of the full adder.

Qs. The value on top of each resistance change is the average amount of I_{DDT} increment induced by the injected IRF over a clock cycle. As can be seen, there is no direct correlation between the I_{DDT} increment amount and the logic error in the circuit. For example, from 7.11 to 12.15 ns, the average value of I_{DDT} is 333 μA on average, which is relatively high compared to the other values. However, the circuit has not failed during this period. On the contrary, logic errors at the output *Qs* have occurred at times 25, 33, 53 and 65 ns, and the average amount of I_{DDT} current increment is 211, 258, 220, and 325 μA, respectively.

It has been observed that IRFs affect the I_{DDT} of a circuit in a very short time. However, in the long term, the effect of IRFs on I_{DDT} is negligible. Moreover, no correlation has been observed between the amount of I_{DDT} disturbance and a logic error in the circuit.

3.3.3 The Effect of IRFs on Power Lines

In another experiment, an IRF was injected into the V_{DD} line. The resulting outputs are shown in Figure 3.12. It shows that an IRF with 34 kΩ resistance has resulted in a logic error at Qs. More simulations show that IRFs with resistances higher than 31 kΩ in the V_{DD} line may cause logic errors.

According to the results, the injected IRF can disrupt the voltage level of the output signals, resulting in a wrong value being captured in the D-flip-flops. It can be concluded from the above simulation results that the impact of an IRF at the V_{DD} line (similarly the Ground line) is quite significant. However, since power lines have thicker metal layers and a limited number of vias [Che18b], the occurrence rate of IRFs in them is statistically low.

From these results, it can be concluded that IRFs can induce intermittent delay faults in data paths, cause random timing violations, and sometimes lead to logic errors in digital circuits.

3.4 IRF Evaluation at Board Level

In this section, the effect of IRFs has been investigated at the board level. For this purpose, one of the most common data transmission protocols at the board level, the Universal Asynchronous Receiver Transmitter (UART) [Fan11], has been selected as an example for IRF evaluation. First, a simulation-based IRF injection was carried out. Then, a hardware-based IRF generator was proposed and utilised to evaluate the UART protocol against IRFs in actual hardware.

3.4.1 Simulation-based IRF Injection

The IRF model and generator mentioned previously have been used for fault simulations. A basic block diagram of a UART is shown in Figure 3.13. Essential parts are a baud-rate generator, consisting of a clock generator and a clock divider. Furthermore, it includes a transmitter and a receiver.

The UART receiver and transmitter have been implemented employing the VHDL language and synthesised by our Synopsis tool using the Nangate 45 nm CMOS library [Nan08]. The synthesis result was imported into the Cadence Virtuoso tool, which was used to perform simulation-based IRF injection at the transistor level. A power supply V_{DD} of 1.1 V was used, and the baud-rate was 921 kHz. Several IRFs were applied to the transmission line between the UART transmitter and receiver by the simulation-based IRF generator.

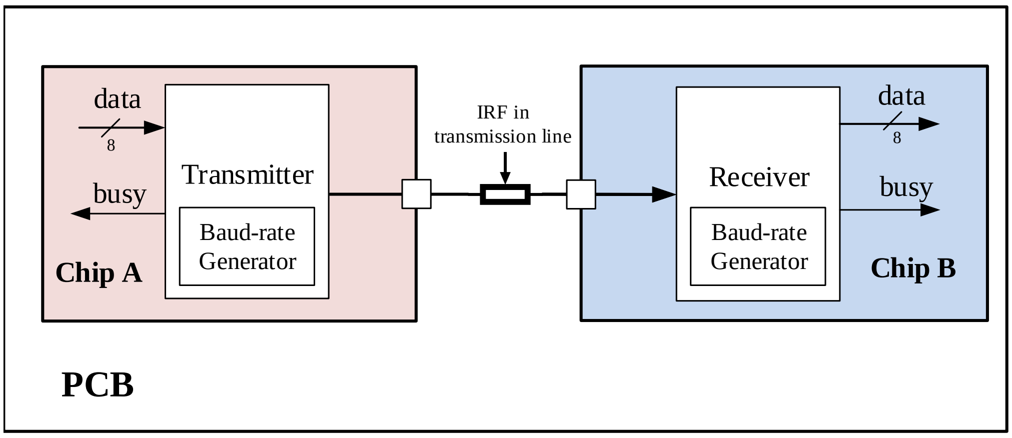


Figure 3.13: The basic block diagram of a UART transmission line in a PCB between two chips. The IRF injection has been carried out in the transmission line between chips A and B.

Table 3.1: The parameters used for simulation-based IRF injection in the transmission line.

Parameter	Minimum	Maximum
Start time	0.1 μs	1 μs
Resistance	1 Ω	80 $k\Omega$
Active Time	0.6 μs	3 μs
Inactive Time	0.3 μs	1 μs
Burst length	1	5
Safe time	1 μs	30 μs

Every frame of data consists of a start bit and, subsequently, the less significant bit of an 8-bit data string. At the end of the frame, there is one stop bit.

Figure 3.14 shows one example of the simulation result for IRF injection in a transmission line with the UART protocol. In Table 3.1, all relevant parameters of the IRF generator are given. For simplicity, uniform distributions were used in all cases, but they can be changed if required.

In a serial data transmission unit, like a UART, an IRF can result in a *single bit*, *multiple-bit* error or a *framing* error. In our case, a frame consists of one start bit, an 8-bit data string and one or two stop bits, respectively. If the receiver gets more or fewer bits than the size of a frame, a framing error will occur.

Moreover, fault simulations have been carried out in the V_{DD} line of the receiver. The simulation results showed that an IRF can result in a framing error; hence, the received data was invalid. The simulation results also revealed that IRF occurrences with resistances higher than about 40 $k\Omega$ result in logic errors in the transmitted data. It should be noted that increasing the baud-rate will reduce the timing margins,

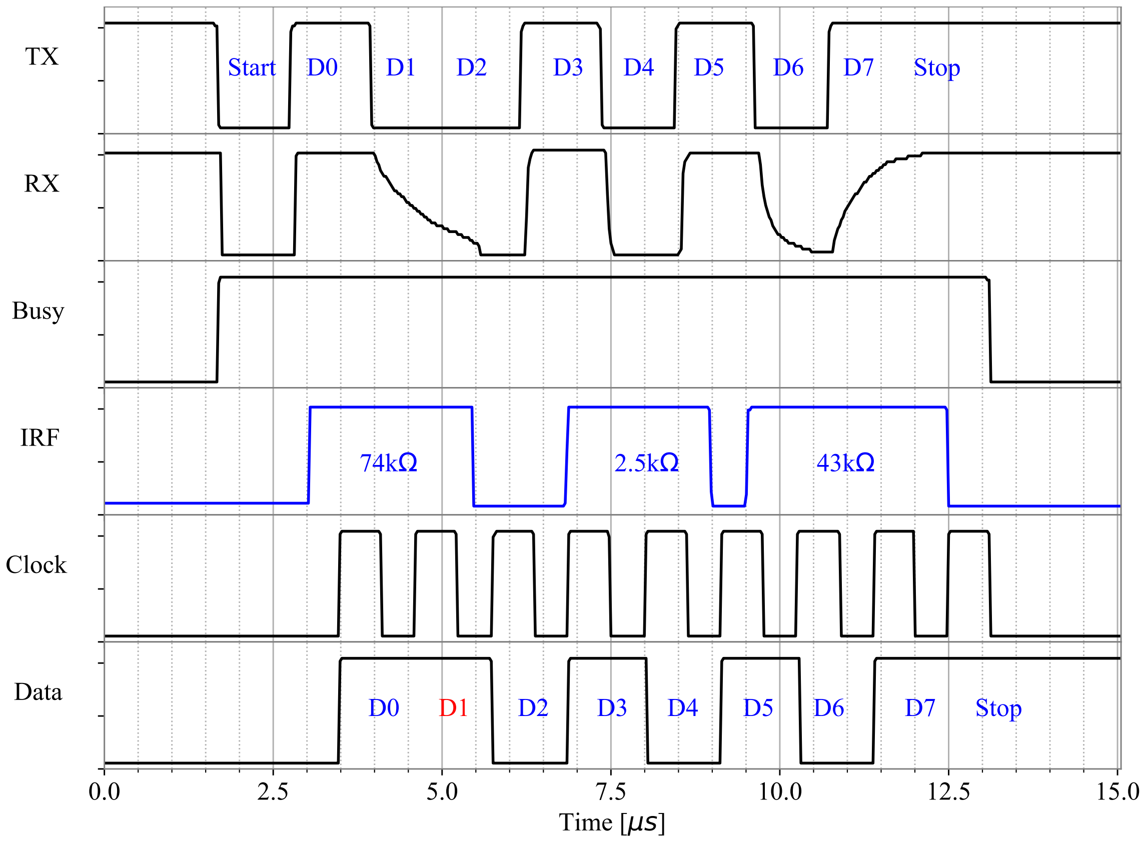


Figure 3.14: Simulation results of an IRF injection in a transmission line with the UART protocol. The injected IRF caused timing distortion on the signal *RX* and a bit-flip on *D1* (shown in red).

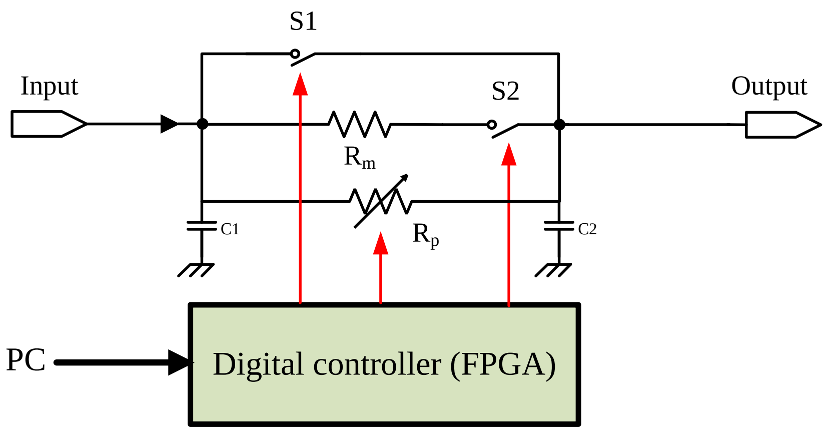


Figure 3.15: Scheme of our hardware IRF generator.

and IRFs with lower resistances can already cause logic errors. It can be concluded that the effect of IRFs on the V_{DD} line is more severe than the transmission line. However, compared to other interconnections, power lines are usually thicker, wider and with a limited number of vias, making them less susceptible to IRFs.

A 64-bit Linux machine with 8 GB RAM running at 3.4 GHz has been used to perform the simulations. The simulation time required for every fault injection was about 5 minutes.

3.4.2 Hardware-based IRF Injection

In the previous section, a simulation-based intermittent resistive fault injection was applied to evaluate the effect of IRFs in board data transmissions. However, simulation-based fault injection is highly time-consuming, particularly in the case of IRFs with a complex model compared to other fault models, such as stuck-at faults. As mentioned previously, every IRF injection can take several minutes. Hardware-based fault injection is an alternative solution for accelerating fault injections. This technique also allows for studying the real-time behaviour of a circuit.

3.4.2.1 The Hardware IRF Generator

In Figure 3.15, a straightforward implementation of our hardware IRF generator is shown. It can be connected as a programmable resistive wire between any two nodes. There are, of course, some constraints with regard to timing and the allowed branch voltages and currents and switch positions. The basic concept is to use a simple network of a fixed resistance R_m (1 Ω) and programmable variable resistances

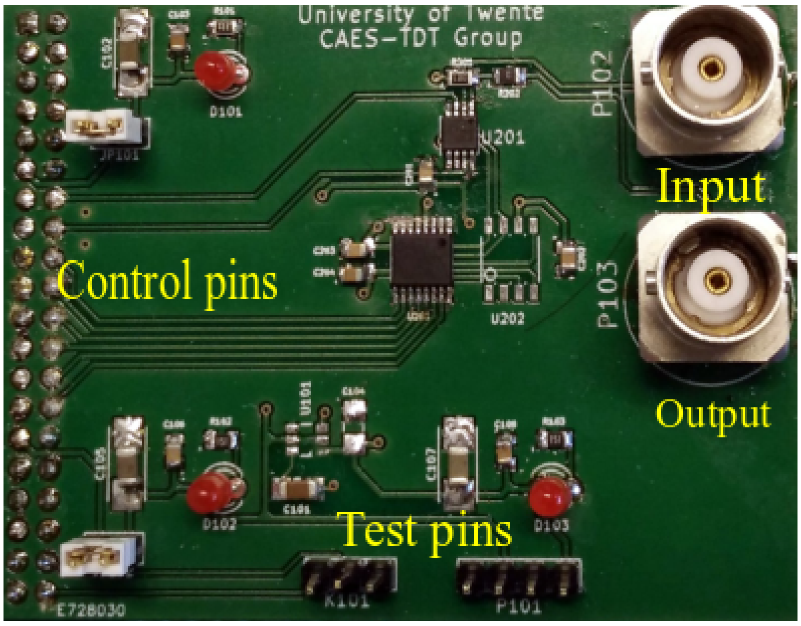


Figure 3.16: Photograph of the implemented hardware IRF generator.

R_p. Several small capacitances C_1 and C_2 with values of 100 nF have been used for noise reduction.

The main characteristic of the design is as follows. The total resistance range from 1 Ω to 2.7 kΩ. The input and output capacitances are 63 pF and 54 pF, respectively. Under a load of 10 Ω and 100 pF, the time and resistance resolutions of the IRF generator were measured to be 120 ns and 33 Ω, respectively. A digital controller (in our case an FPGA) generates the desired resistance by providing the required control signals for the resistances and the digital switches S_1 and S_2 on the board.

The most obvious disadvantage of this architecture is that the RC combinations limit its speed behaviour (see Figure 3.15), of which the intrinsic parasitic capacitances are the most difficult to circumvent.

However, it provides a time resolution of 120 ns, which is acceptable for IRF generation in terms of speed. A photograph of the implementation of the hardware design is shown in Figure 3.16. In this figure, from left to right, one can see: the control pins, which can be connected to an FPGA or a microcontroller to control the switches and the digital potentiometer, test pins, input and output connections for connecting the line under test to the IRF generator.

A personal computer executes a Python script that generates a burst of random resistance values based on our IRF model and the generation flow as depicted earlier in Figure 3.7. The generated burst sequence can be transferred to an FPGA board by a serial communication link. The FPGA is responsible for synchronizing and

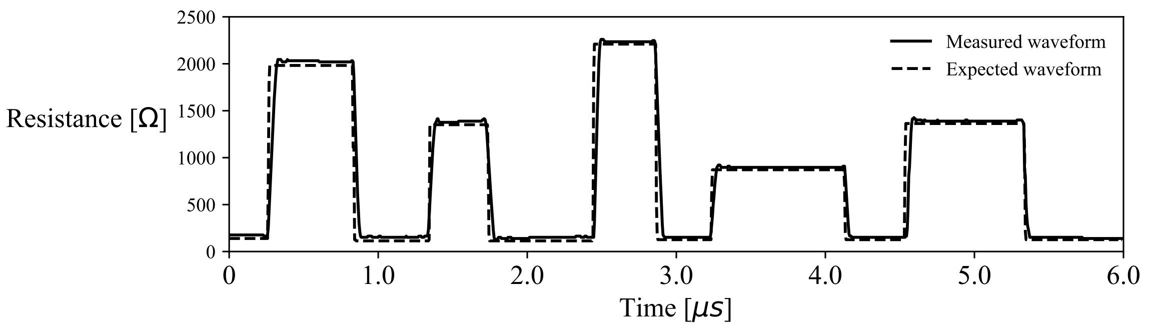


Figure 3.17: An example of a measured output of the IRF generator.

Table 3.2: IRF generator parameters in the case of real measurements.

Parameter	Minimum	Maximum
Start Time	0.1 μs	1 μs
Resistance	2.5 Ω	2.5 kΩ
Active Time	0.6 μs	3 μs
Inactive Time	0.3 μs	1 μs
Burst Length	1	2
Safe Time	1 μs	30 μs

generating the actual control signals for the on-board programmable switches and potentiometers in the IRF generator.

An example of a generated IRF signal by our generator is shown in Figure 3.17. A load with 10 Ω resistance and 100 pF capacitance has been used for this measurement. There are two waveforms in the figure; one is a resistance sequence measured from the IRF board, and another (shown with dashed lines) is the expected sequence generated by the Python script. As can be seen, the measured resistance sequence approximately follows the expected sequence. There is a small RC error because of the parasitic capacitances of the existing switches and potentiometers.

3.4.2.2 Measurement Results

The hardware IRF generator has been used to evaluate IRFs in a UART, implemented on an FPGA. The implemented UART design is the same as the previous simulations used. Figure 3.19 depicts the block diagram of the IRF generation and the measurement set-up of the IRF injection in the UART transmission line. A photograph of the overall test set-up is shown in Figure 3.18.

The baud-rate in these measurements is similar to the one in the previous simulations. The parameters of the IRF generation are as provided in Table 3.2. The major difference is in the range of resistances. In our case, they are below 3 kΩ,

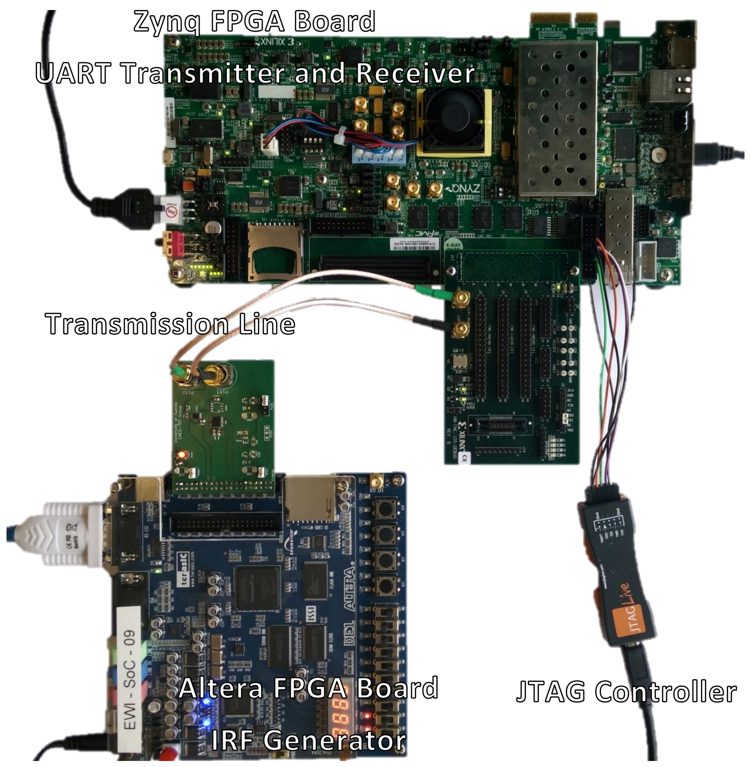


Figure 3.18: Measurement set-up of the IRF generator and the UART for detecting output failures.

which is relatively low and hence less likely to cause logic errors.

The UART first receives the start bit and subsequently the less significant bit of an 8-bit data stream. In Figure 3.20, the data stream is '01101010' ($D_7 ... D_0$).

The signal *IRF* represents the occurrence of the injected IRF in the transmission line. The last line is the signal *busy*. The transmitter activates this signal during data transmission. In this example, the data stream received by the receiver was 011010$\underline{\mathbf{0}}$0. It means the IRF caused one bit-flip in the bit-stream (D_1).

Figure 3.21 shows another case of an IRF injection. In this example, an IRF with two pulses of 2.7 $k\Omega$ and 2.3 $k\Omega$ resistances has been injected into the transmission line. The experiment showed that the injected IRF caused two bit-flips in the transmitted data stream (D_4 and D_5).

Figure 3.22 shows another example of IRF injection in a UART transmission. The stop bit became zero due to an IRF with a pulse of 1.8 $k\Omega$. In this case, the injected IRF resulted in a frame error.

Our experiments revealed that an IRF might cause no error, single-bit, multiple-bit, or framing errors based on the range of resistance changes, activation time and

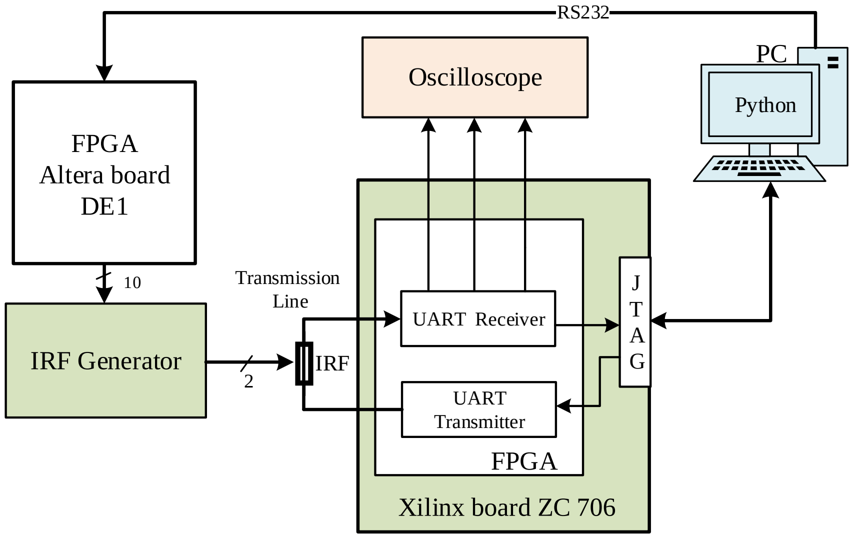


Figure 3.19: Block diagram of the IRF generation and the measurement set-up for IRF injection in the UART transmission line.

the burst length of the IRF.

The comparison of the simulation-based and hardware-based fault injection shows a significant speed enhancement in IRF injection. In the previous section, the fault simulations were carried out using a 64-bit Linux machine with 8 GB RAM and running at 3.4 GHz. The simulation time was about 5 minutes. Whereas the hardware-based fault injection could be carried out in about 30 μs. This means the intermittent resistive fault injection has been accelerated by seven orders of magnitude.

As has been shown by several measurements, IRFs may cause real-life failures in more complex CMOS logic systems. The hardware IRF generator allows us to have a framework for further investigation and study of IRFs. Because IRFs are fixed in place, early detection and localisation of them will make system troubleshooting and repair considerably easier and less costly.

3.5 Summary

This chapter has addressed research questions Q1 and Q2. The thermal cycling technique has been carried out to study the behaviour of intermittent resistive faults (IRFs). Several cold solder joints and loose connectors were created for the measurement experiments. Our experimental results show that IRFs can be invoked under local temperature changes at the board level. The measurement results

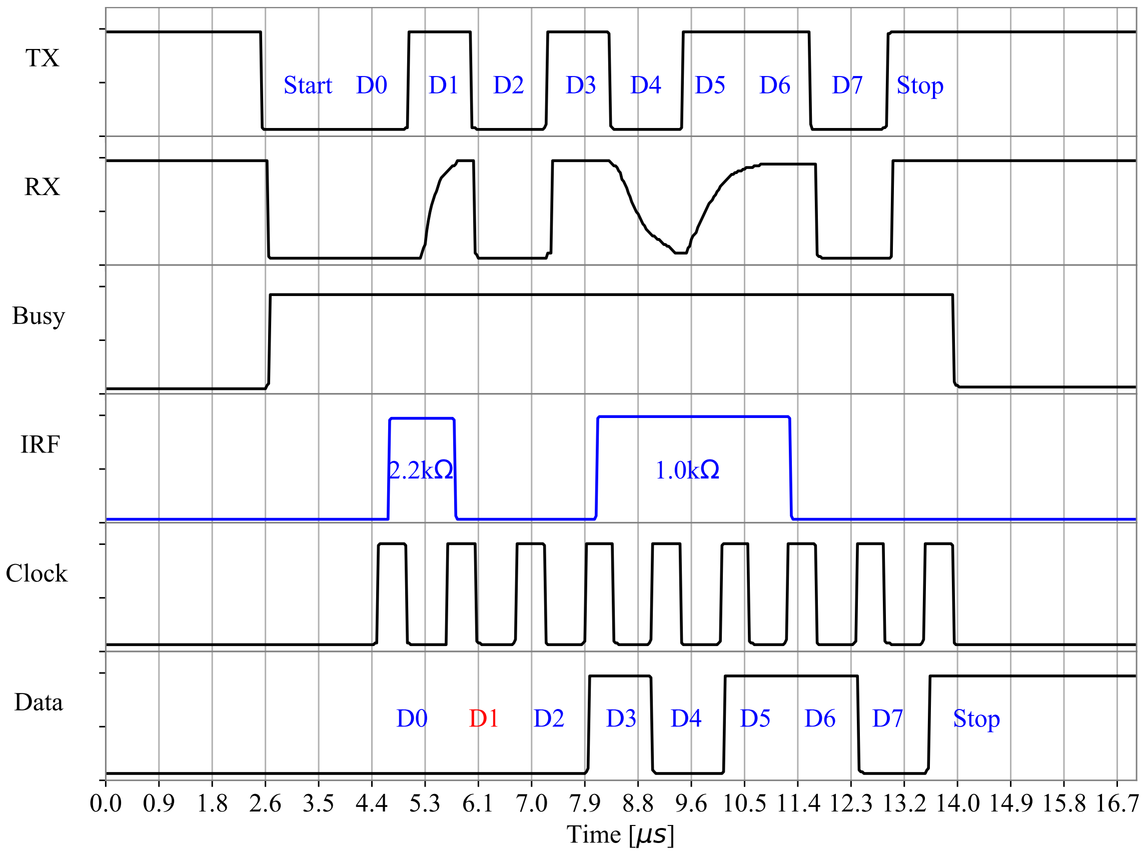


Figure 3.20: Measurement example of an IRF injection using the hardware IRF generator in the UART transmission line. The injected IRF caused one bit-flip (D1) in the transmitted data (shown in red).

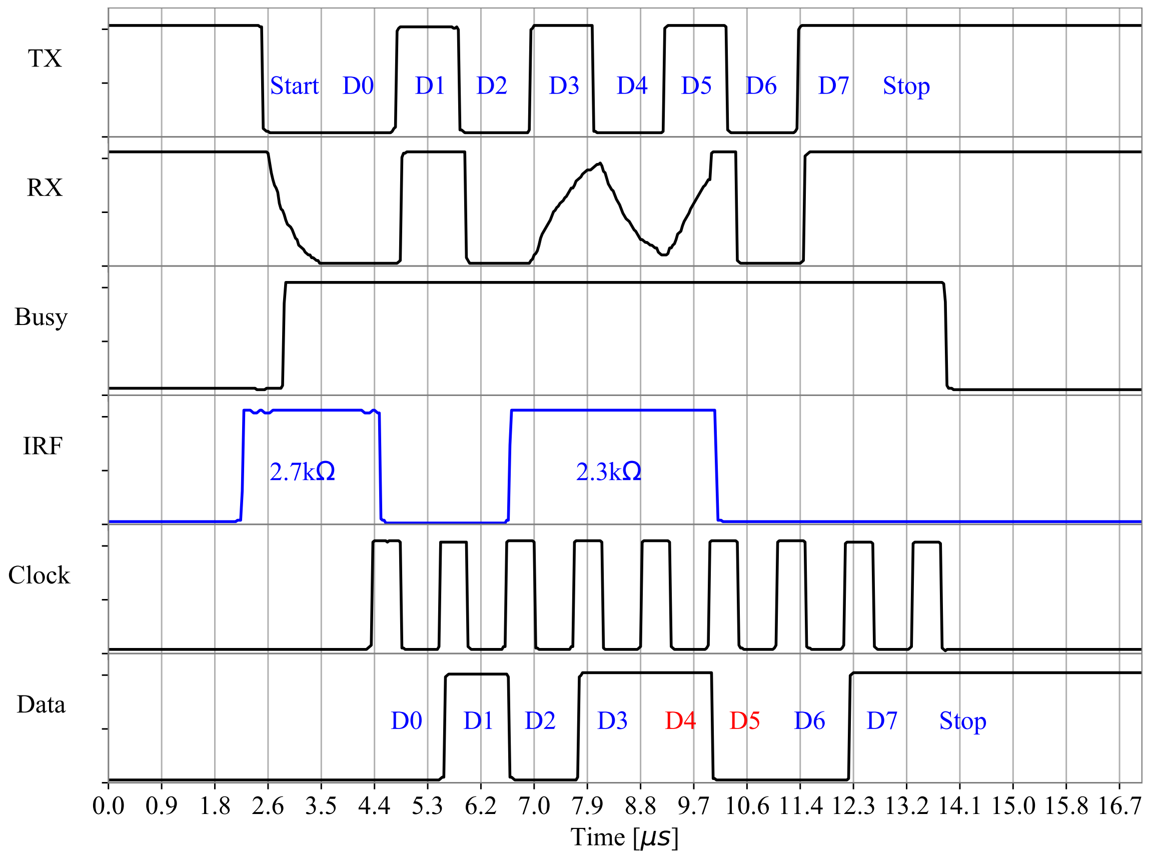


Figure 3.21: Measurement results of a UART using a different IRF resulting in multi-bit faults (D4 and D5) in the transmitted data (shown in red).

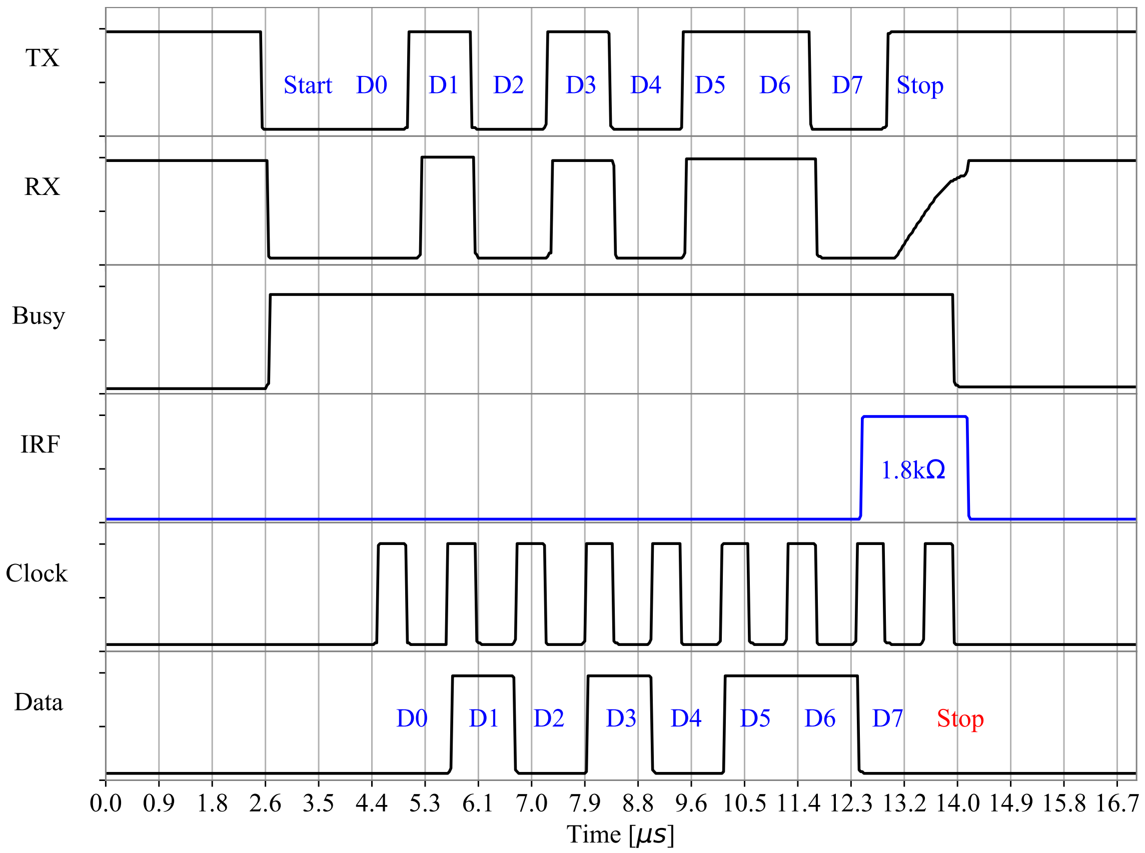


Figure 3.22: Measurement results of a UART that injected IRF resulting in a frame error.

confirmed IRFs occur randomly over time. If an IRF occurs at a specific location, there is a possibility of its recurrence at that same location later. Additionally, it has been observed that IRFs have the potential to evolve and worsen when subjected to prolonged thermal stress.

Based on our measurements, an IRF model and a general procedure for IRF generation were proposed. This IRF generator was used later to study the behaviour of circuits under the influence of IRFs at both the transistor and board levels. A simulation-based IRF generator was utilised for IRF injection at the transistor level. In the end, a hardware-based IRF generator was proposed to speed up IRF injection experiments. The simulation and experimental results reveal the vulnerability of digital systems to IRFs. Moreover, IRFs can create random timing errors in data paths and transmission lines. In addition, they can lead to single and multiple bit-flips, as well as frame errors in UART data transmission.

The experimental results showed that the hardware-based IRF generator could accelerate the IRF injection procedure by about seven orders of magnitude compared to a simulation-based IRF generator. The proposed IRF generator can be used to evaluate the sensitivity of new designs regarding IRFs. Moreover, it allows designers to validate their new IRF detection methods and techniques. The IRF generator can also be used in analogue and mixed-signal circuits. The following chapters employ the IRF generators to verify our new IRF embedded test instruments at chip and board levels.

Chapter 4
Intermittent Resistive Fault Detection at Chip Level

Abstract– *Intermittent resistive faults may randomly occur in any interconnection inside an integrated circuit. They can arise in any interconnection at any moment during the operational time. This kind of fault may appear repetitively in a location and can gradually become increasingly severe during the lifetime of a system. Eventually, they may evolve into a permanent fault. Therefore, detecting and repairing such faults is crucial before they become permanent, resulting in a system failure, especially in safety-critical systems.*

This chapter proposes two embedded test instruments (ETIs) for IRF detection at the chip level. The ETIs are fully-digital, scalable and compatible with the IJTAG standard. The effectiveness of the ETIs has been validated via simulation-based fault injections. It will be shown that the ETIs can be used to distinguish between intermittent, ageing, and permanent faults.

In the second part of the chapter, an algorithm is proposed to find the best locations in a design for IRF monitoring and ETIs insertion in the chip. The objective of this algorithm is to maximise the IRF coverage of a circuit. The algorithm maximises the IRF coverage for the most vulnerable parts of the circuit to IRFs, such as wire interconnections, vias and input pads. Two different selection strategies have been investigated. The first strategy selects nodes from end nodes. In the second strategy, the node selection is from all nodes (internal and end nodes). The simulation

Parts of this chapter were published in:

- [Ebr16b] H. Ebrahimi and H. G. Kerkhoff, "Detecting intermittent resistive faults in digital CMOS circuits," in IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 87–90, 2016.
- [Ker17] H. G. Kerkhoff, A. Ghazanfar, and H. Ebrahimi, "An automotive MP-SoC featuring an advanced embedded instrument infrastructure for high dependability," in IEEE International Test Conference in Asia (ITC-Asia), pp. 65–70, 2017.
- [Ebr20] H. Ebrahimi and H. G. Kerkhoff, "A new monitor insertion algorithm for intermittent fault detection," in IEEE European Test Symposium (ETS), pp. 1–6, 2020.
- [Ebr21] H. Ebrahimi and H. G. Kerkhoff, "Embedded test instrument for intermittent resistive fault detection at chip level and its reuse at board level," in IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), pp. 75-80, 2021.

results reveal that the algorithm achieves a higher IRF coverage with the second strategy. The efficiency of the algorithm is verified using simulation-based fault injections.

At the end of this chapter, a periodic testing method for IRF detection at the chip level has been introduced. A Network-on-Chip (NoC) infrastructure with a mesh topology was utilised as a case study. The implementation of the testing method requires a small chip area and power consumption overheads being 3 % and 2.5 %, respectively. The simulation results reveal that as the number of test cycles rises, the IRF coverage increases.

4.1 Introduction

The detection of intermittent resistive faults (IRFs) is a challenging task. There are two main problems to deal with in a practical situation [Qi08, Con08]. The first is the moment of occurrence, which can be any time in the future. A major issue is that in the worst case, it rarely occurs [Yi08]. The second problem is that the duration of the occurrence can be very short. This necessitates detecting extremely brief events, which typically necessitates very high sample rates or precise small-delay management in the case of a purely digital technique. Therefore, conventional test methods are unlikely to detect these faults in practice.

One alternative approach is on-line monitoring [Jef05, Kat15]. On-line monitoring is defined as concurrently monitoring a system while it continues its intended function [Nic98]. This approach is based on employing embedded test instruments (ETIs) to monitor the health of a system while it is in operation. Since IRFs may remain inactive during test phases, the on-line monitoring technique is preferred for IRF detection. In the on-line monitoring approach, the system is monitored in operation during its particular environmental conditions (e.g. voltage, temperature, and vibration).

Another alternative approach for intermittent fault detection is periodic testing [Kra06]. In this approach, the system is periodically tested. Therefore, the probability of intermittent fault detection increases by increasing the number of tests at the cost of downtime and power consumption.

In this chapter, both on-line monitoring and periodic testing will be investigated for IRF detection. The rest of the chapter is organised as follows. In Section 4.2, two embedded test instruments (ETIs) are proposed for IRF detection at the chip level. It will be demonstrated in Section 4.3 that the ETIs may be utilised to differentiate between intermittent, ageing, and permanent faults. Section 4.4 presents an algorithm for locating and inserting ETIs in a circuit to maximise IRF coverage. The objective of the algorithm is to cover the most vulnerable parts of the

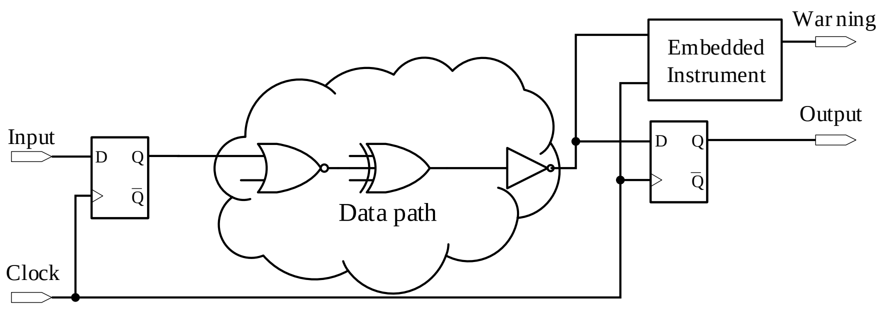


Figure 4.1: An example of a data path with an embedded test instrument.

circuit to IRFs, such as wire interconnections, vias and input pads. In Section 4.5, a periodic testing method for IRF detection at the chip level has been introduced. An implementation of this method in a Network-on-Chip (NoC) infrastructure is demonstrated. The chapter is concluded with Section 4.6.

4.2 IRF Embedded Test Instruments

An IRF consists of a sequence of random resistance changes occurring with random durations. As seen in the previous chapter, the occurrence of IRFs in an interconnection of a data and control path can cause random delay faults. Similar to other delay faults [Bow09, Rah14], random delay faults caused by IRFs may result in timing deviations in paths.

In a data path, a timing failure may occur in the case that the delayed data does not meet the set-up requirement of the endpoint flip-flop and causes a late transition near the clock edge. On-line monitoring is a technique for detecting timing violations in data paths in an integrated circuit. In this approach, embedded test instruments (ETIs) will be placed in various locations inside the chip. Our ETIs monitor and report timing violations of the circuit while it is operational.

Figure 4.1 depicts an example of a data path with an embedded test instrument. This figure shows how an ETI can be employed at the end of a data path to monitor timing deviations. If a timing violation occurs, the ETI issues a warning signal.

The ETI can only detect a timing violation if it causes a signal transition larger than the timing slack of the affected path. The timing slack is defined as the delay between the data arrival time and the active edge of the clock minus the flip-flop set-up time [Reb11].

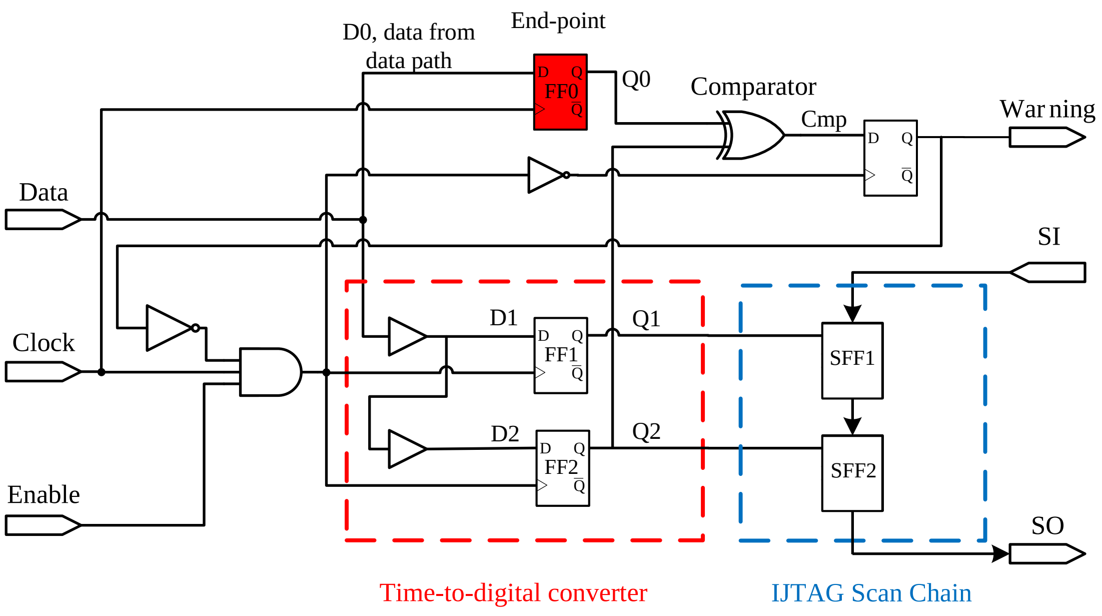


Figure 4.2: The schematic of the IRF-ETI Type 1.

Some examples of on-line monitoring for timing violation detection have been presented in [Lai14, Sha17, Sal15, Das09]. These a address the timing violations induced by ageing or process, voltage and temperature variations. To the best of our knowledge, no existing approach addresses IRF detection at the chip level.

In the following, the usage of ETIs for IRF detection has been investigated. Two different digital ETIs which can detect IRFs will be presented. The motivation is to design scalable and fully-digital ETIs which can detect IRFs. Moreover, the ETIs should be compatible with the IJTAG standard [Shi14] to facilitate fault localisation and management.

4.2.1 IRF Embedded Test Instrument Type 1

Figure 4.2 shows the schematic of our first IRF embedded test instrument (IRF-ETI Type 1). It consists of a time-to-digital converter (TDC), an IJTAG scan chain, and a comparator (the XOR gate). The TDC consists of a delay chain and two flip-flops (FF1 and FF2). If a timing violation occurs, this TDC captures the degree of the timing violation and converts it into a digital value.

The delay chain can be created using digital standard cells such as buffers or inverters based on the required resolution for the TDC. Note that the flip-flop coloured red (FF0) is the end-node of a data path in the design and not a part of the IRF-ETI.

The data can be captured at the rising clock edges (see Figure 4.3). The comparator checks whether the IRF-ETI and the end-node flip-flop capture the same data. The captured data in the end-node and the IRF-ETI will be compared at the falling edges of the clock. In the case of discrepancy, a warning signal will be issued. If a warning occurs, the captured data in the FF1 and FF2 flip-flops (a sequence of zeros and ones) will be passed onto the IJTAG scan flip-flops (SFF) [IEE14]. This data will be interpreted into the amount of violence in picoseconds in the fault-management software.

The resolution of the IRF-ETI depends on the resolution of the delay elements in the delay chain. For example, if the IRF-ETI is built using TSMC 40 nm CMOS technology and a buffer cell is utilised as the delay element, then the resolution of the IRF-ETI will be 5 ps. It should be noted that the number of flip-flops and the length of the delay chain can differ depending on the required accuracy for IRF detection.

The IRF-ETI must have at least two flip-flops to allow the fault-management software to classify the detected fault as transient, intermittent, or permanent. Our classification strategy is as follows. If several faults with different degrees of timing violations are detected during a specific duration, it can be classified as an IRF. If only one timing violation occurs, it can be classified as a transient fault. Finally, it can be considered a permanent fault if the number of violations is more significant than a specified threshold.

The clock in the IRF-ETI is gated by the *Enable* and the *Warning* signals. This ensures that the IRF-ETI does not detect a new timing violation and does not issue a new warning if either the IRF-ETI is not activated or a warning has been previously issued. The IRF-ETI can be activated using an IJTAG network.

An example of IRF detection is drawn in Figure 4.3. The signals are the following. The data at the end of the path is shown by *D0*. Signals *D1* and *D2* are the data that pass through the delay chain. *Q0* is the data captured by the end-node flip-flop at the rising edge of the clock. *Q1* and *Q2* are the values captured in the IRF-ETI flip-flops. The figure shows how the IRF-ETI detects a late transition at the end of the data path. As can be seen, the *D0* signal has a high-to-low transition in the detection timing window, resulting in a timing violation being captured in flip-flop FF2 (Q2=1). The *CMP* signal, generated by the XOR gate, shows the comparison between the *Q0* and *Q2* signals. Since *Q0* and *Q2* are not equal, the *Warning* signal is raised at the falling edge of the clock.

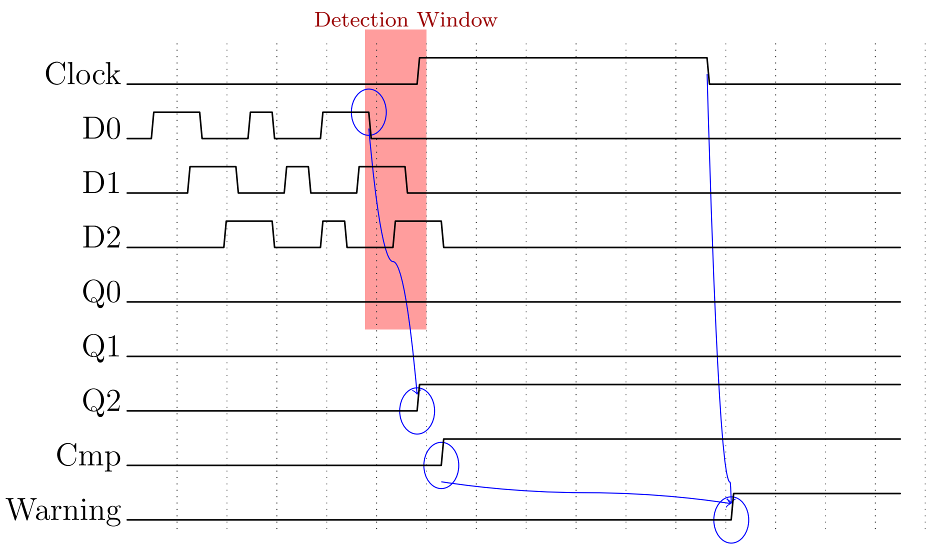


Figure 4.3: An example IRF detection using the IRF-ETI Type 1.

4.2.2 IRF Embedded Test Instrument Type 2

Another approach in designing an IRF-ETI is to utilise an extra guard-band signal. Instead of having a delay chain on data, the delay chain in this method is applied to an input referred to as *guard-band*.

In the previous IRF-ETI (Type 1) that does not include a guard-band signal, there is the possibility that small glitches in the data line become masked. This may reduce the resolution of the IRF-ETI to detect small IRFs. Another advantage of IRF-ETI Type 2 is that the guard-band signal can be programmable. As a result, this IRF-ETI can have a programmable detection window.

Figure 4.4 shows the schematic of the IRF-ETI Type 2. The inputs of the ETI are as follows. The clock in the design is represented by the *Clock* signal. The *Data* input represents the signal for monitoring at the selected data path. The *Guard-band* input determines the detection timing window. The *Enable* input is for enabling the ETI. *Clk1* and *Clk2* signals are delayed versions of the Guard-band signal, which are generated by the delay chain in the ETI. They are connected to the clock inputs of FF1 and FF2 flip-flops, respectively. The *Data* input is connected to the *D* input of the FF1 and FF2 flip-flops. Therefore, the FF1 and FF2 can capture any late transition in the *Data* signal. The XOR gate continuously compares the values captured in the end-point flip-flop (FF0) and the FF1 flip-flop and issues a warning signal in the case of discrepancy.

If a warning is issued, the stored values in FF1 and FF2 will be transferred to the IJTAG scan flip-flops (SFFs). The guard-band signal can be generated in a variety of ways. A phase-lock loop (PLL) can be used to create a programmable guard-band.

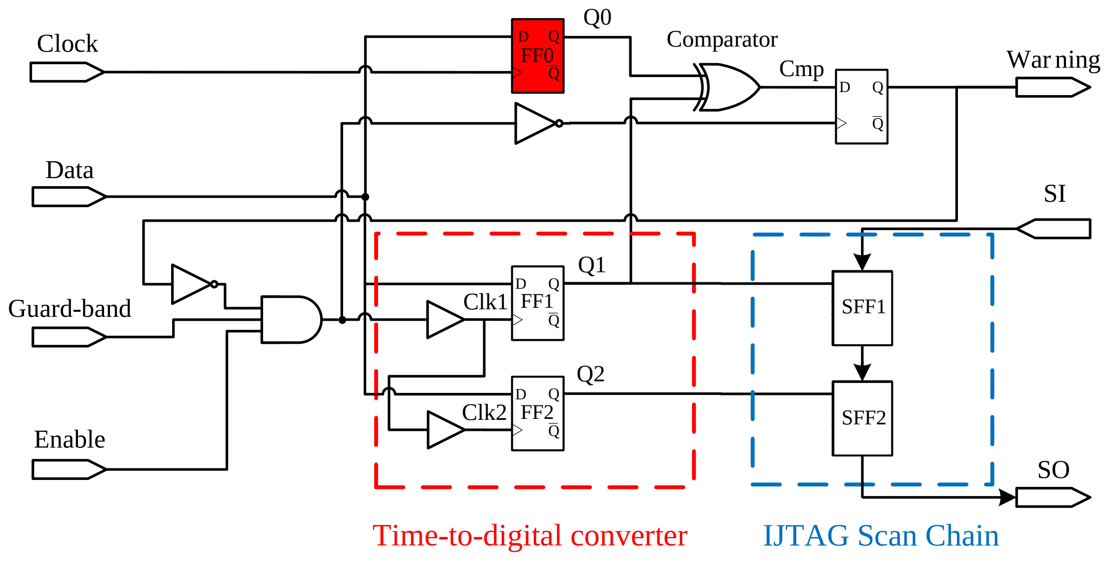


Figure 4.4: The schematic of the IRF-ETI Type 2 with a programmable Guard-band signal.

Alternatively, a fixed guard-band signal can be obtained utilising delay elements. An example of an IRF detection by the IRF-ETI Type 2 is illustrated in Figure 4.5. In this figure, the *Guard-band* signal creates a detection window (shown in red). The *Data* signal shows the data at the data path selected for monitoring. The *Clk1* and *Clk2* signals indicate when FF1 and FF2 capture the *Data* signal, respectively. The *Q1* and *Q2* signals are the captured values by FF1 and FF2, respectively. As can be seen, a late transition on the *Data* signal has occurred in the detection window. This transition has been captured in FF1, and as a result, the *Q1* signal became '1'. The *Cmp* signal shows the result of the comparison of the *Q0* and *Q1* signals. The transition in the *Q1* signal has resulted in a transition in the *Cmp* signal. At the end, the change of the *Cmp* signal issues the *Warning* signal.

4.2.3 IJTAG wrapping for IRF-ETIs

Figure 4.6 shows the IRF-ETI Type 1 wrapped in an IJTAG wrapper. The IRF-ETI is the same as that shown in Figure 4.2. In addition, the *reset* signal is shown. A simple access procedure written in PDL [IEE14] of the IRF-ETI is presented in Figure 4.7. It consists of a simple logic synchronisation mechanism between the IJTAG controller and the IRF-ETI.

This controller continuously polls the warning flag until a warning is raised (lines 4-8). Then, it reads the output data and concurrently writes a *1* in the acknowledge register to instruct the instrument to reset its warning and data outputs (lines 9-11). Finally, the controller resets the acknowledge flag in the next scan cycle (lines 13-14)

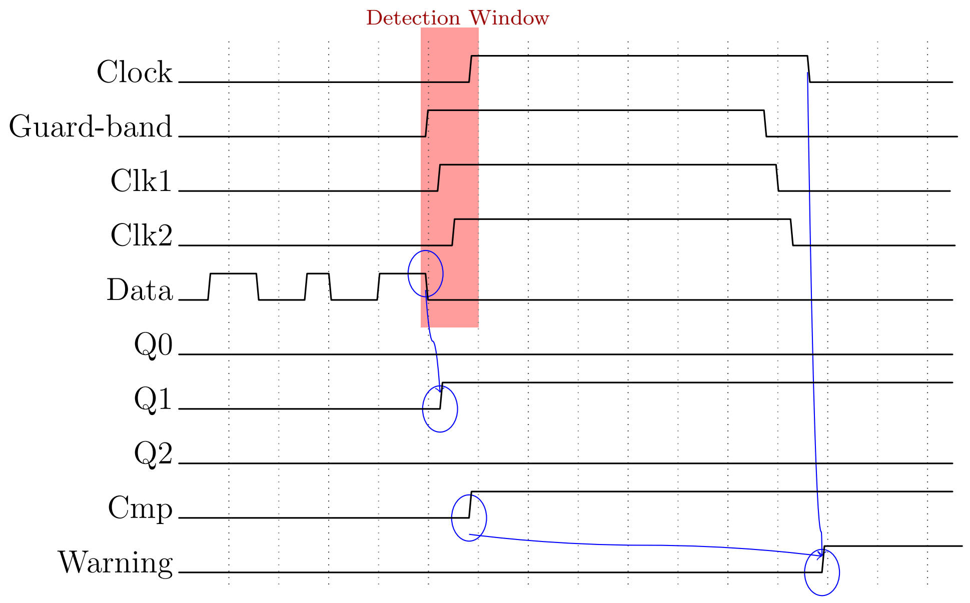


Figure 4.5: IRF detection using the IRF-ETI Type 2 with a guard-band.

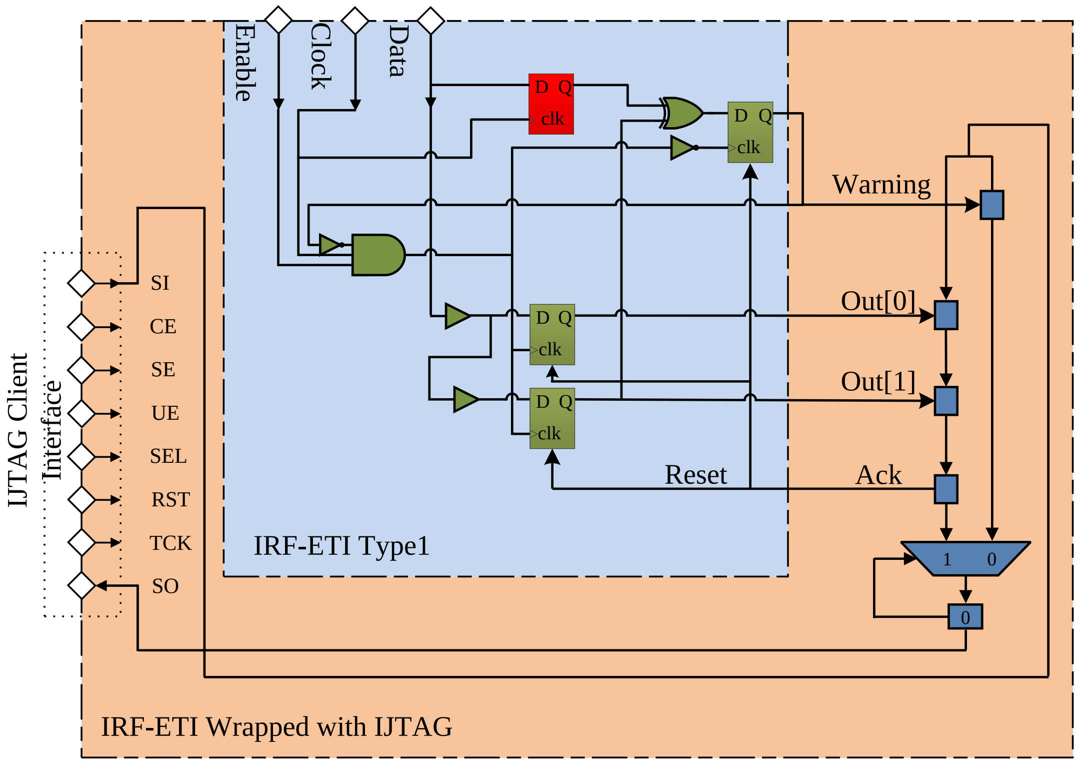


Figure 4.6: An IRF-ETI Type 1 wrapped in the IJTAG standard wrapper (orange part).

```
iPDLLevel 1
iProcsForModle IRF_ETI
set Status 0

while {$Status == 0} {
    iRead IRF_ETI.Warning
    iApply
    set Status [iGetReadData IRF_ETI.Warning]
}

iRead IRF_ETI.Out
iWrite IRF_ETI.Ack 0b1
iApply
set IRF_Out [iGetReadData IRF_ETI.Out]
iWrite IRF_ETI.Ack 0b0
iApply
```

Figure 4.7: Simple example of a PDL script for the IRF-ETI Type 1 wrapped in the IJTAG network.

to allow the IRF-ETI to restart the fault monitoring process.

Note that the IJTAG wrapper and PDL script are the same for both types IRF-ETIs.

4.3 IRF and Ageing Simulations

In this section, the capability of our ETIs to detect IRF and ageing effects at the chip level has been investigated using simulations. Post-synthesis simulations have been carried out. In the following, IRF injection and the effect of ageing at the transistor level are presented for IRF-ETI Type 2.

4.3.1 IRF Detection Using the ETIs

The IRF generator used for fault injection is the same described in Section 3.2. A software module has been developed to generate these faults in the Cadence Virtuoso platform [Cad14]. Six parameters can be set by determining their minimum and maximum possible values. In addition, according to the specific requirements, any (random) distribution, such as uniform and Gaussian, can be chosen. The values and distributions applied for the simulations in this section are listed in Table 4.1.

For example, an AES-128 encoder circuit [AES19] using the Nangate 45 nm CMOS technology [Nan08] has been selected as a case study. The encoder circuit was

Table 4.1: Range of used parameters in the IRF generator for fault simulations.

Parameter	Minimum	Maximum
Start time	1 ns	10 ns
Resistance	100 Ω	100 $k\Omega$
Active time	0.1 ns	2 ns
Inactive time	0.1 ns	2 ns
Burst length	1	20
Safe time	1 ns	1 s

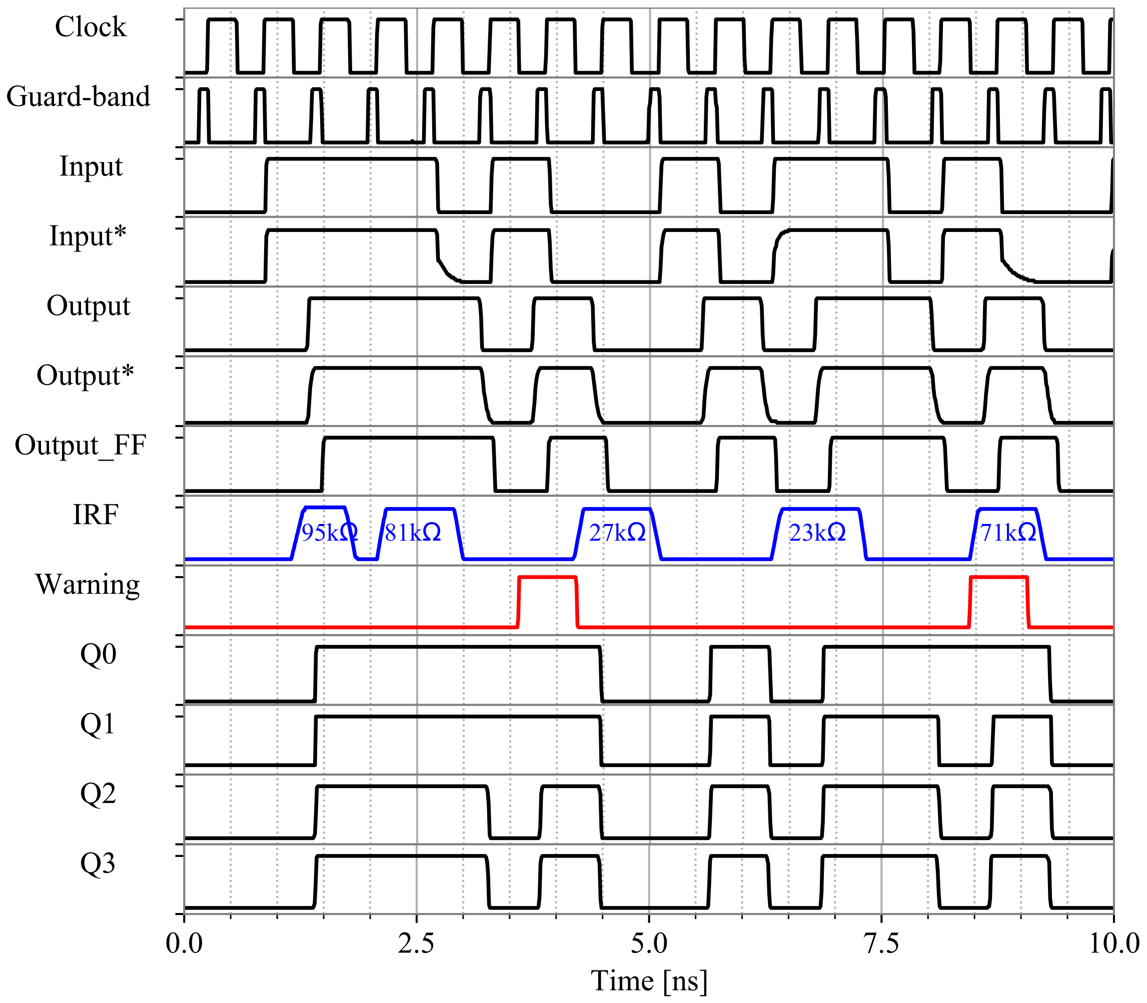


Figure 4.8: Simulation results of the IRF-ETI Type 2 after IRF injection.

synthesised using the Synopsys Design Compiler tool [Kur12] with a performance-driven strategy. The synthesis resulted in a clock frequency of 1.6 GHz for the circuit. A gate-level netlist generated by the synthesis was exported to the Cadence Virtuoso tool [Cad14] to perform IRF injection at the transistor level. Several critical paths and near-critical paths were selected for our IRF injection. The IRF-ETI Type 2 was placed at the end-node of the selected paths.

One example of a simulation result is shown in Figure 4.8. The injected IRF is shown in blue. It shows a burst of five resistance changes during 10 ns. An IRF-ETI Type 2 with four flip-flops was chosen for this simulation. The delay chain consisted of buffer cells with a 5 ps propagation delay.

In Figure 4.8, the clock of the system is shown on the top. Other signals from top to down are as follows. The *Guard-band* signal indicates when the IRF-ETI starts to capture the input data. The *Input* signal is a sample of data transitions at the beginning of the critical path. The *Input** signal shows the signal *Input* after the IRF injection. The *Output* and *Output** signals are the outputs of the path in fault-free and faulty cases, respectively. *Output-FF* is the signal captured by the flip-flop at the end of the path. As can be seen, there is not any functional error in the signal, although the IRF-ETI detects two small delay degradations. The *Warning* signal, shown in red, is the output of the IRF-ETI (Type 2). It shows the IRF-ETI detected two late transitions in the data (signal *Output**). The last four signals (*Q0*, *Q1*, *Q2* and *Q3*) are the outputs of the flip-flops of the IRF-ETI. The captured values by these flip-flops show the degree of degradation and timing violation.

At the time 3 ns, the IRF-ETI detects a timing violation. As a result, the *Warning* signal is issued, and two D-flip-flops out of four in the IRF-ETI have captured '1' (*Q3..Q0=0011*). It means that the guard-band in the path has been violated as much as two buffer delays (10 ps). The *IRF* signal shows that the IRF induced a very high resistance (81 kΩ) at this time. Similarly, at 8 ns, a timing violation is detected as much as one buffer delay (5 ps) because the value of 1 (*Q3..Q0=0001*) is stored in the flip-flops. It can be seen that the value of the IRF has been near 71 kΩ at 8 ns.

4.3.2 Ageing Simulation

The experimental results in [Sto10] show that NBTI is one of the dominating factors resulting in timing degradation in CMOS technologies. NBTI has been shown to cause shifts in threshold voltages of up to 50 mV over an operating lifespan of 10 years in 65 nm CMOS technology.

To model the delay induced by ageing faults (NBTI), the V_{th} of all PMOS

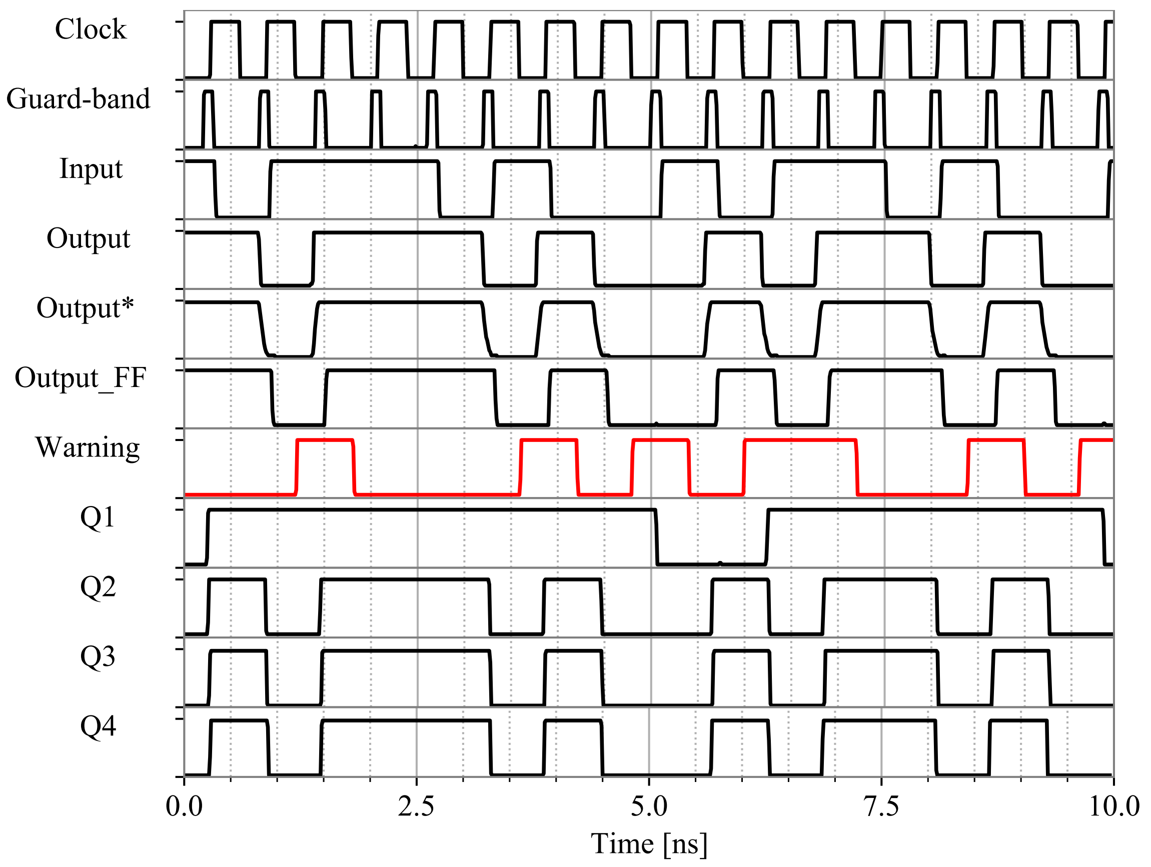


Figure 4.9: Simulated results of the IRF-ETI Type 2 after NBTI ageing (50 mV increase in V_{th} over 10 years).

transistors in the circuit and the IRF-ETI were increased by 50 mV in the Cadence simulator. Note that the ageing model used here is only a rough ageing estimate. However, this model is suitable for comparing the effects of ageing and IRF on our IRF-ETIs.

Figure 4.9 shows the Cadence Virtuoso simulation results of a combinational circuit where its transistors are subjected to ageing. The output (logic) behaviour of the circuit and the IRF-ETI Type 2 have been evaluated. The signals in Figure 4.9 are identical to those in Figure 4.8.

In Figure 4.8, the signal *Output* shows the simulation results of the circuit before ageing. As expected, the IRF-ETI detects no timing (guard-band) violation. The signal *Output** shows the simulation results of the selected data path after ageing. It can be seen from the *Warning* signal that the IRF-ETI has detected some timing violations. The first flip-flop of the IRF-ETI has captured a timing violation, but the other three flip-flops have not detected any violations.

Based on the above simulations, it can be concluded that one flip-flop for an

IRF-ETI is sufficient for ageing-degradation detection. This is because the severity of timing violations caused by ageing faults is not random and gradually increases over a system lifetime. However, in IRFs, the severity of timing violations varies randomly, necessitating at least two flip-flops to capture timing violations.

In both ageing and IRF detection, a warning flag is raised after a timing violation. Consequently, the captured value in the flip-flops will be transferred to the IJTAG [Shi14] registers. Using the IJTAG standard to extract timing information and fault locations, one can determine whether the fault is caused by ageing or intermittent resistive faults.

The above results demonstrate that our IRF-ETIs can detect certain IRFs in the data paths of the design. However, because of the additional cost of area and power consumption overhead, using the IRF-ETIs for all interconnections in a system is not practical. Therefore, an efficient location-selection algorithm is required to achieve maximum IRF coverage using the IRF-ETIs. In this chapter, the IRF coverage is defined as the ratio of the total number of vias and interconnections that our IRF-ETIs can monitor to the total number of vias and interconnections in the design. An efficient selection algorithm for IRF monitoring is introduced in the next section. The proposed algorithm provides the maximum IRF coverage for a given area or a power overhead constraint. The efficiency of the algorithm is verified using simulation-based fault injections.

4.4 Node Selection and ETI Insertion Algorithm

This section will propose a node selection algorithm for IRF monitoring. First, the node-selection flow is introduced. Then, the concept of IRF propagation and coverage will be explained. Next, the node-selection algorithm will be introduced. At the end, the results of simulation-based fault injections are presented.

Two distinct strategies were investigated. In the first strategy, the algorithm only chooses monitoring locations from end-nodes. The node selection in the second strategy, on the other hand, is made from all nodes (internal and end nodes).

Both proposed IRF-ETIs can be used for IRF monitoring. For simplicity, only IRF-ETI Type 1 has been chosen in this section. Figure 4.10 shows how an IRF-ETI Type 1 can be used at an end-node and an internal-node of a data path. The output of the IRF-ETI is a *Warning* signal, which indicates whether an IRF is detected or not. The IRF-ETIs continuously monitor signals at the end of the selected paths. A small slack is an indication that the circuit is close to failing. Measuring timing slack can give an early warning of deterioration and trigger preemptive actions to avoid failure because of IRFs. It ensures there is never a timing violation during operation

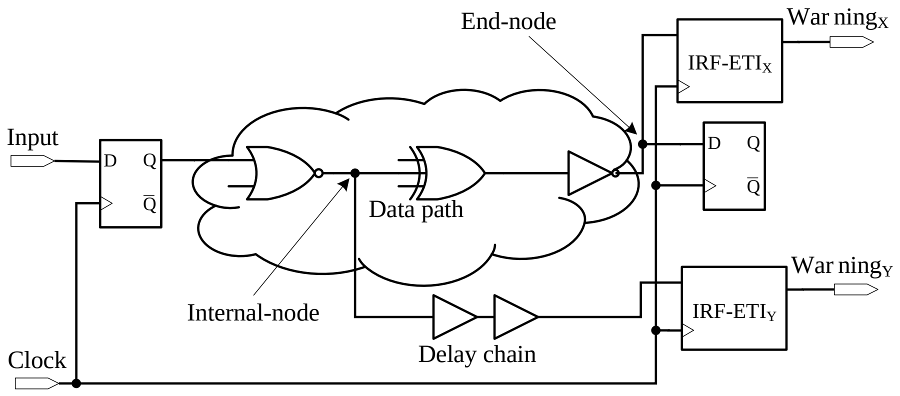


Figure 4.10: An example of the insertion of IRF-ETIs Type 1 at the end-node and an internal-node of a data path.

to avoid capturing incorrect data.

4.4.1 The Selection Flow

Our algorithm can be applied either after synthesis or after the place and route of a design. The layout information should be estimated if the algorithm is used after the synthesis. In the case that the algorithm is applied after the place and route phase, a re-place and re-route procedure should be carried out after the IRF-ETI insertion.

The flow of the proposed node selection is depicted in Figure 4.11. After synthesising (or after placing and routing) a given design, a static timing analysis reveals the slack of paths, the critical and near-critical paths of the design. The algorithm avoids inserting IRF-ETIs in critical or near-critical paths to prevent performance degradation. Therefore, the nodes on the critical and near-critical paths were given a lower priority to be chosen by the algorithm.

In the next phase, the probability of an IRF being propagated and captured by an IRF-ETI is calculated for each net. These calculations provide a set of nets for each node. Each set comprises all nets (including interconnections and vias) that can be covered by inserting an IRF-ETI at the corresponding node. The nodes that cover more nets with higher fault probability are ideal candidates for IRF-ETI insertion. Based on the extracted information in the previous phases, the algorithm selects a set of nodes that provide a maximum IRF coverage within a given (e.g. area-overhead) constraint. Finally, the IRF-ETI will be inserted at the selected nodes. Clearly, the area overhead is a limiting factor that restricts the number of IRF-ETIs and,

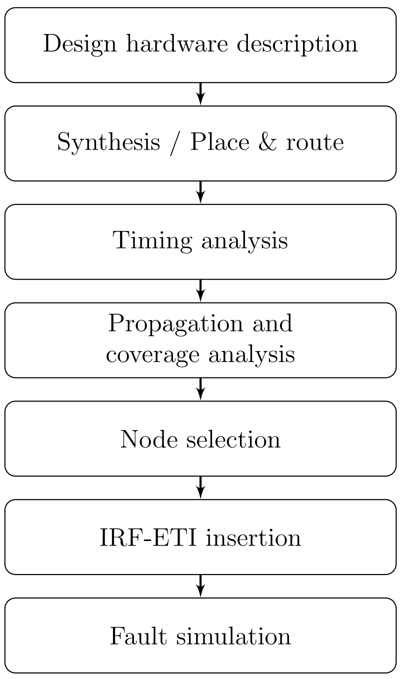

Design hardware description
Synthesis / Place & route
Timing analysis
Propagation and coverage analysis
Node selection
IRF-ETI insertion
Fault simulation

Figure 4.11: The flow of the proposed node selection and the IRF-ETI insertion.

consequently, the total IRF coverage of the system.

It should be noted that the probability of an IRF occurring in a net varies depending on the layout of the design after the placing and routing phase. The number of wires and vias in each net determines the sensitivity of the net to IRFs. Therefore, the node selection algorithm should be executed after the placing and routing phases. However, one can use the algorithm after the synthesis. Because layout information is unavailable during the synthesis, the number of wire interconnections and vias must be estimated. To extract the correlation between the number of vias and interconnections required for a net based on its fan-out number, several ISCAS'89 benchmarks [Brg89] and AES-128 benchmark [AES19] have been used. After the placing and routing, via and track utilisation were extracted from layouts. The findings revealed a linear relationship between the number of fan-outs and the number of vias and interconnections a net requires to be routed.

An example of this relation for the AES-128 benchmark [AES19] is shown in Figure 4.12. This figure depicts the relation between the number of fan-outs for each net and the number of its vias for the AES-128 benchmark. The average of the actual number of vias obtained from the placing and routing phase is presented. Moreover, a calculated regression line is depicted in the figure. The occurrence number of each fan-out in the layout is indicated in blue. For example, 430 nets in the design have a fan-out of 3. These nets require, on average, 5 vias for routing. As expected, the number of nets with high fan-outs is less than those with low fan-outs. Note that the

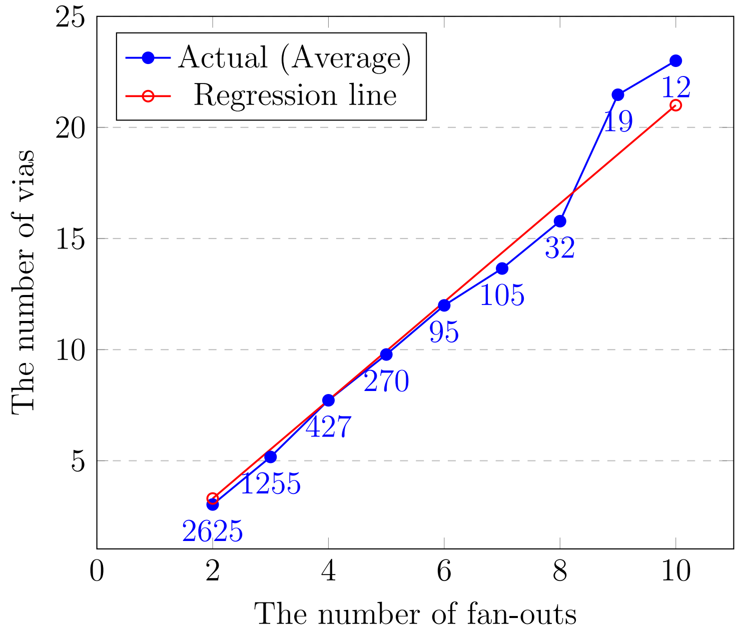


Figure 4.12: The number of vias versus the fan-outs for the nets in the AES-128 benchmark.

fan-outs with an occurrence of less than 3 are not shown in this figure. As can be seen, there is a linear relation (with a standard error of 0.045) between the number of fan-outs and the number of required vias for each net.

Figure 4.13 shows another example of the linear relation between the number of fan-outs and the number of vias for benchmark s13207 from the ISCAS'89 benchmarks [Brg89]. The relation between the wire length and fan-outs for the nets in the AES-128 and s13207 benchmarks are shown in Figures 4.14 and 4.15, respectively. A linear relation between the wire length and fan-outs of nets exists with standard errors of 0.54 and 1.14 for the AES-128 and s13207 benchmarks, respectively. As the number of fan-outs increases, the number of nets having that fan-out decreases; therefore, the average value for the number of vias deviates more from the regression line.

The Synopsys Design Compiler tool [Kur12] was used for the synthesis. The fault injection and simulation were executed using the QuestaSim tool [Que16]. The other phases of the flow were implemented using our Python scripts.

4.4.2 IRF Propagation Probability

The circuit structure and input stimuli have a significant impact on fault propagation. Moreover, IRFs can occur in every net (wire interconnections and vias) of the design. They may cause timing delays in the affected net. The induced timing delay

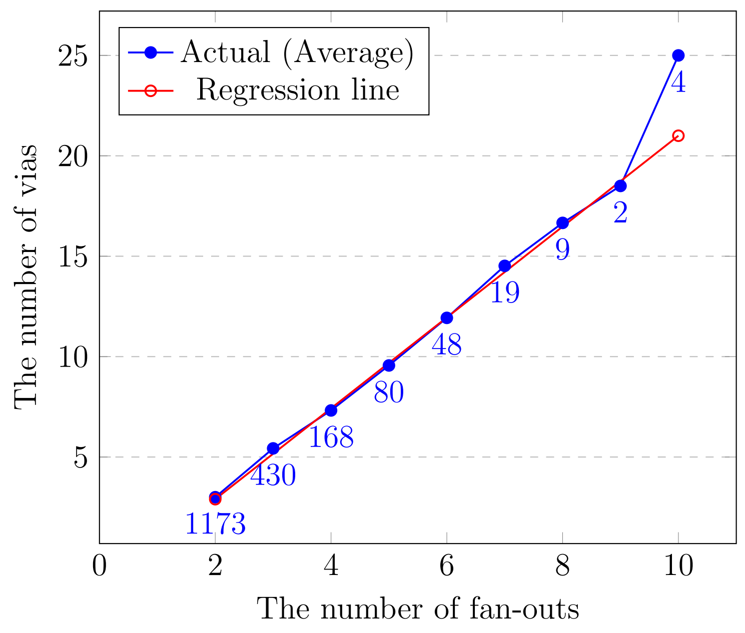


Figure 4.13: The number of vias versus the fan-outs for the nets in the s13207 benchmark.

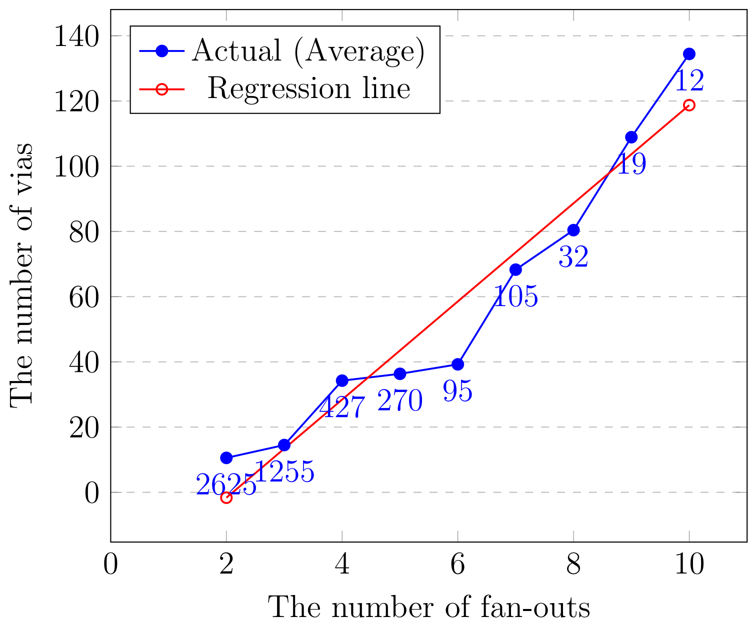


Figure 4.14: The total wires length versus the fan-outs for nets in the AES-128 benchmark.

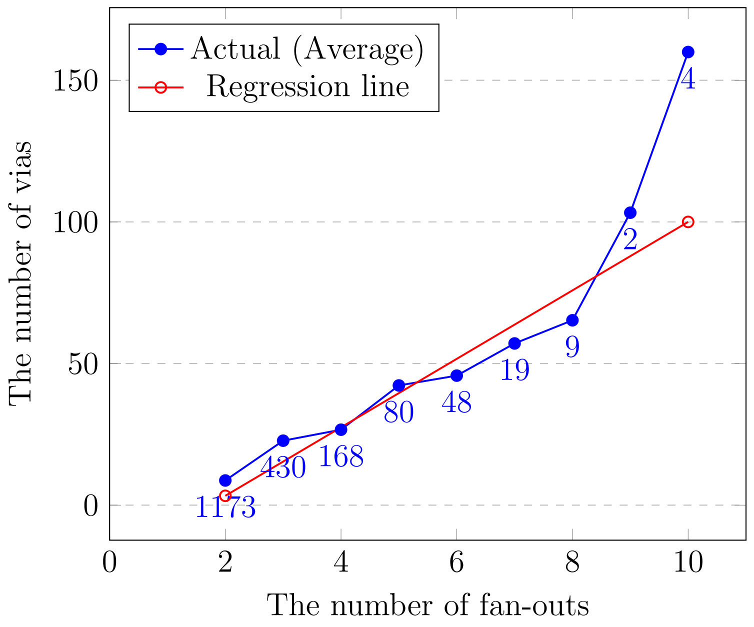


Figure 4.15: The total wires length versus the fan-outs for nets in the s13207 benchmark.

may be either masked by a logic gate or propagated through the path and cause a logic error. Therefore, to determine which locations are suitable for monitoring IRFs, the fault propagation probability for each net should be calculated. The proposed selection algorithm will use this information to select nodes that can cover nets with a high probability of fault propagation.

In the literature, fault-propagation calculations have been used to estimate soft-error rates [Gha18, Che13]. A similar calculation was used to estimate IRF propagation in this chapter.

The propagation probability calculation is similar to [Gha18]. An example of the propagation probability calculation for a simple circuit is shown in Figure 4.16. The propagation probability is calculated for net x. As can be seen, the signal x can be propagated via two paths. One is represented by the blue line, and the other by the red line. The signal probability of each net is written on top of the net. The propagation probability of each logic gate is written underneath it. In this example, the signal probability for the input nets is assumed to be 0.5. In the red path, the probability that the signal x propagates via the NAND gate is 0.5. After the NAND gate, the signal x can be propagated via the NOT and XOR gate without being masked. For the blue path, the propagation probability of the NOR gate is 0.25 (1 - 0.75). Therefore, the probability that a fault on x is propagated to a flip-flop (FF1 and FF2) is 0.5 + 0.25 - (0.5 * 0.25) = 0.625. Since the red path propagation probability is higher than the blue one, the FF1 flip-flop is better for IRF monitoring.

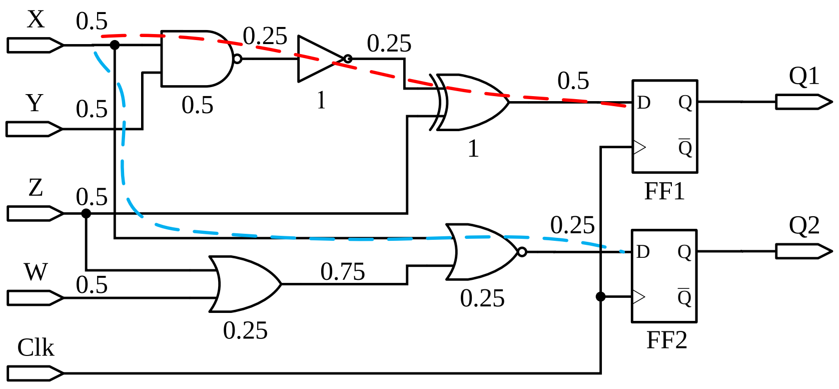


Figure 4.16: An example of the calculation of the propagation probability. The numbers on the top of the nets and under the gates represent their propagation probability.

4.4.3 The Selection Algorithm

The node-selection algorithm is shown in Algorithm 1. The inputs of this algorithm are a graph of netlist G, an area constraint A and the timing data T. The netlist is a directed graph. In this graph, edges are nets, and vertices are either logic gates or nets. The timing data T is the output of the synthesis tool, which contains the information on the switching activity of each net. The algorithm produces a list of selected nodes S that will be utilised for IRF-ETI insertion. The algorithm consists of three main functions and a while loop.

The function *Propagation_and_Coverage_Analysis* is used to calculate the fault-propagation probability for all nets of the netlist (lines 17-24). This function receives the graph of the netlist G as an input. The output of this function is a list for each end-node/internal-node of the circuit. The lists contain a set of nets that the corresponding node can cover. The propagation probability for each net has been calculated using the equations described in the previous subsection. This function uses a depth-first search (DFS) to traverse the netlist. The DFS function will be called for every net of the netlist ($n \in N$) only if the net is not processed before ($n \notin V$). The output of the DFS function is a list of nodes (L) for net n. A list of nets (P) will be extracted from these lists for each node. List P contains all nets that terminate at a given node as well as the propagation probability for each net. It should be noted that the correlation between paths due to reconvergent fan-out [Jah19] are considered in this function. This provides a more accurate and less pessimistic fault-propagation probability calculation. An example of a path with a reconvergent fan-out is depicted in Figure 4.17. The logic gate NOT has a reconvergent fan-out with two outputs in this figure. These two outputs reconverge at the logic gate OR. The propagation probability of the net after the OR gate, P_4, can then be calculated from the probabilities of P_1, P_2 and P_3 with the following

Algorithm 1: Selection algorithm

```
Input: G; A; T
Output: $S$ ={Selected nodes}
 1: $P \leftarrow Propagation\_and\_Coverage\_Analysis(G)$
 2: while Area_overhead < A do
 3:     $e \leftarrow Find\_best\_node(T, P)$
 4:     Add $e$ to $S$ set
 5:     Area_overhead ← $Estimate\_area\_overhead(S)$
 6: function FIND_BEST_NODE(T, P)
 7:     E={all internal or end nodes in the netlist G}
 8:     for all $e \in E$ do
 9:         $varFaultCoverage \leftarrow Fault\_coverage(P, e)$
10:         $varMaximum \leftarrow 0$
11:         $objective \leftarrow \alpha\ (varFaultCoverage)$
12:         if $objective > varMaximum$ then
13:             $varMaximum \leftarrow objective$
14:             $SelectedNode \leftarrow e$
15:         Remove_nets_from_search_space(SelectedNode)
16:     return $SelectedNode$
17: function PROPAGATION_AND_COVERAGE_ANALYSIS(G)
18:     $N$ ={all nets in the netlist G}                    ▷Set of all nets
19:     $V$ ={}                                             ▷Set of visited nets
20:     for all $n \in N$ do
21:         if $n \notin V$ then
22:             $L \leftarrow DFS(G, V, n)$
23:     $P \leftarrow Create\_list\_of\_nets\_for\_each\_node(L)$
24:     return $P$
25: function DFS(G, V, n)
26:     $F$ ={Flip-flops and Outputs}
27:     $S \leftarrow Extract\_direct\_successors(n)$
28:     for all $s \in S$ do
29:         if $s \notin V$ & $s \notin F$ then
30:             Calculate the propagation probability for s
31:             $DFS(G, V, s)$
32:         else
33:             Update the list of nodes for net n
34:     return $L = \{the\ list\ of\ nodes\ for\ net\ n\}$
```

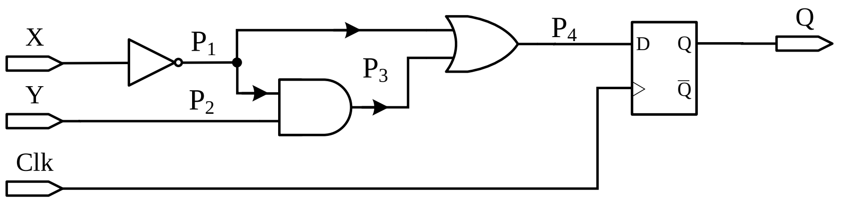


Figure 4.17: An example of a circuit containing a reconvergent fan-out.

equation:

$$P_4 = 1 - (1 - P_1P_2)(1 - P1) \tag{4.1}$$

A pseudocode of the function *DFS* is presented in lines 25-34 of Algorithm 1. This function is recursive and calls itself until it reaches either an end-node or until all nets of the netlist have been visited. The function calculates the propagation probability for each net while traversing it (line 30). This information will be returned to the function *Propagation_and_Coverage_Analysis* as a list L. The function *Find_Best_Node* receives the information of timing (T) and the coverage and propagation probability of nets (P) as inputs. The value of the IRF coverage (*varFaultCoverage*) can then be calculated for each node using the following Equation:

$$IRF\_Coverage = \frac{\sum_{n=1}^{M} P_{F_n}}{N} \tag{4.2}$$

where M is the number of nets covered by the given node, and N is the total number of nets in the design. The variable P_{F_n} is the fault-propagation probability for each net n (without reconvergent fan-out) which can be calculated from Equation 4.3.

$$P_{F_i} = \prod_{g=1}^{n} P_{T_g} \tag{4.3}$$

where P_{T_g} is the propagation probability for a logic gate g and n is the number of logic gates in the path. This equation indicates that an IRF in a net i can only be captured by a path end-node if it can pass through all of the gates in the path.

The objective of the function *Find_Best_Node* is to find a node with the maximum IRF coverage value. At the end of the function calculation, the covered net by the selected node will be removed from the search space.

The algorithm selects a set of nodes with the maximum IRF coverage in a while loop using the functions mentioned above. The algorithm estimates the area required for the IRF-ETIs based on the number of selected nodes and then repeats the process until the area overhead reaches the given A constraint. The while loop condition can easily be substituted to satisfy new objectives such as maximum IRF coverage. The algorithm greedily searches the entire search space for the best node with the highest IRF coverage.

4.4.4 Simulation Setup

The proposed selection algorithm has been evaluated using a simulation-based fault injection at the logic gate level. The IRF model and injection procedure are based on the one established in Section 3.2. In this section, IRF injection has been carried out at the logic gate level. Simulations in SPICE-like simulators are required to inject an IRF as a burst of random resistances. A SPICE simulation is required to simulate an IRF as a burst of random resistances. However, running a SPICE simulation takes a long time, particularly for large designs.

As concluded in Chapter 3, IRFs may cause random delay faults in data and control paths. One solution for IRF simulations for large designs is using the intermittent delay model [Gar14, Pan12]. As a result, IRFs have been modelled in this section using intermittent delay faults. Several parameters were used to initialise the fault injection procedure. These parameters consist of the number of pulses in a burst, the start and stop times of the burst, active and inactive times, and the delay for each pulse. Random (intermittent) delay faults were injected in a VHDL description of each design.

The range of delay, active and inactive times were chosen based on the clock period of the design being 800 ps (1.25 GHz frequency) for the AES-128 benchmark. The injected delay ranged between 1% (10 ps) and 50% (400 ps) of the clock period to generate various small and large slack violations. These values are analogous to IRFs with resistances ranging from 1 kΩ to 40 kΩ, assuming a capacitance load of 10 fF on the path. To generate IRFs with short and large active times regarding the clock cycle, the active and inactive times were chosen randomly, ranging from 12.5% (0.1 ns) to 150% (12 ns) of the clock period.

The values and distributions applied for the fault injection are shown in Table 4.2.

A Python script was used to implement the IRF model. The script randomly generates IRFs based on Table 4.2 parameters. The output of the script is a TCL file [Fly12], which was used as input for the fault-injection procedure in the Questasim tool [Que16].

Table 4.2: Range of uniformly distributed parameters used for IRF generation during simulations.

Parameter	**Minimum**	**Maximum**
Start time	1 ns	20 ns
Delay	8 ps	400 ps
Active time	0.1 ns	12 ns
Inactive time	0.1 ns	12 ns
Burst length	1	10
Safe time	1 ns	1 ms

The QuestaSim tool [Que16] was used for fault-simulation experiments. The tool receives the procedure of random IRF generation via a TCL file. The technique of the so-called *saboteur* [Gil15] was utilised for the fault injection. It means a VHDL component referred to as *saboteur* is defined for IRF injection. This component was inserted at every net of the benchmark selected for fault injection. A saboteur is inactive during fault-free operation but injects a random intermittent delay fault in the selected net when it becomes active.

4.4.5 Simulation Results

All experiments were conducted on a Linux system with 8 Intel 2 GHz cores and 16 GB RAM. Five of the largest benchmarks of ISCAS'89 [Brg89] and an AES-128 encoder [AES19] were used for our experiments.

The benchmarks were synthesised with the Synopsys Design Compiler tool [Kur12] based on TSMC 40 nm CMOS technology. The performance-driven synthesis was used for the benchmarks. After synthesis, the obtained clock frequency for the benchmarks s38584, s38417 and AES-128 was 1.25 GHz, while it was 1.66 GHz for the others.

Table 4.3 gives an insight into the structure of the benchmarks, such as the number of inputs, outputs, and sequential and combinational cells. The number of combinational cells represents the number of internal nodes. The number of sequential cells determines the number of end-nodes in the design. Besides, the areas for the combinational and sequential parts are presented in terms of square micrometres.

Two different strategies have been investigated. First, the selection was performed only among the end-nodes of a design. In the second strategy, the node selection was carried out on all design nodes (internal and end nodes). The execution times of the proposed selection algorithm for the two selection strategies are shown in Table 4.4. As can be seen, the execution times increase as the number of nodes in the

Table 4.3: Synthesis reports for the benchmarks. The number of input and output ports, and combinational and sequential cells are listed for each benchmark. The area is reported for combinational and sequential cells in μm^2.

Benchmarks	**AES**	**s38584**	**s38417**	**s35932**	**s15850**	**s13207**
Inputs	36	38	28	35	77	62
Outputs	38	317	106	320	150	152
Comb. cells	4619	5772	4163	2445	1885	1379
Seq. cells	910	1275	1564	1728	513	625
Comb. area	24458	23633	17131	12045	6732	4973
Seq. area	12556	16782	20214	9207	6805	7760
Total net #	5630	7177	5759	4211	2560	2131
Fan-out ave. #	3.78	3.22	3.25	3.20	2.84	2.73

benchmark design increases.

Table 4.4: The execution times (in seconds) of the proposed algorithm.

Benchmarks	**End nodes**	**All nodes**
s13207	1.60	1.79
s15850	2.15	2.44
s35932	2.39	2.82
s38417	6.22	7.42
s38584	9.10	10.64
AES-128	16.77	20.23

The results of the algorithm for the six benchmarks are shown in Figure 4.18. The result of the area overhead versus the IRF coverage for the entire interconnection network is depicted for each benchmark. The area overhead represents the additional space required to equip a design with our IRF-ETIs. For example, benchmark s35932 is drawn with a black line. It shows that the algorithm can provide 20 % and 40 % IRF coverage at the cost of 4.3 % and 8 % area overheads, respectively. In s38584, the same IRF coverage can be obtained with 2.8 % and 7.5 % area overheads, respectively. It can be seen that the algorithm requires, on average, 9.2 % area overhead to reach 50 % IRF coverage.

In Figure 4.18, the IRF coverage for each benchmark starts with a different number because the algorithm first starts with providing the full IRF coverage for the input ports. The reason is that input ports are more susceptible to IRFs than internal vias and chip-level interconnections. The IRF coverage for input ports allows using our on-chip IRF-ETIs to detect the IRF occurrence at the board level. For example, having full coverage for all inputs of S13207 (in the first stage of the algorithm) results in 30 % IRF coverage. This IRF coverage includes full input and partial coverage for all internal on-chip connections. The results of full IRF coverage

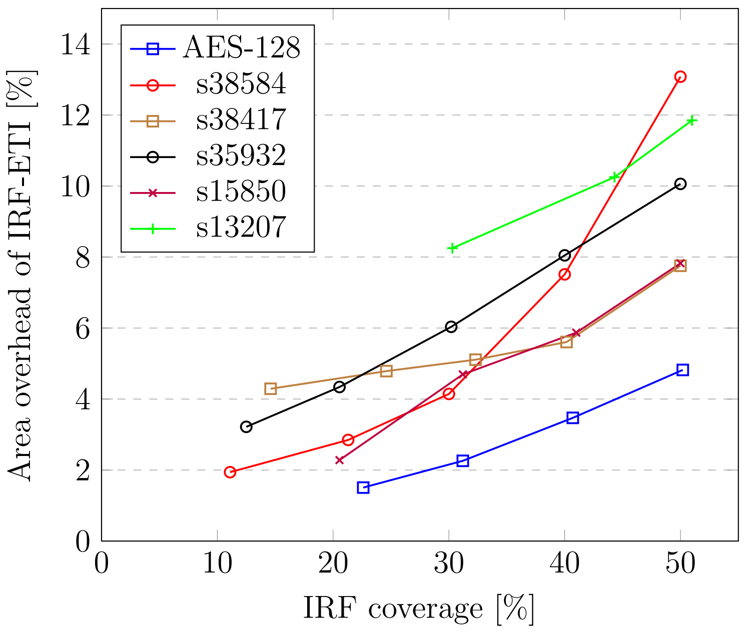


Figure 4.18: Area overhead of IRF-ETI Type 1 required to reach 50% IRF coverage for six different benchmarks.

Table 4.5: Area overhead required for full IRF coverage (100%) of all input ports.

Benchmarks	**Area overhead %**
AES	1.51
s38584	1.94
s38417	4.29
s35932	3.22
s15850	2.28
s13207	8.25

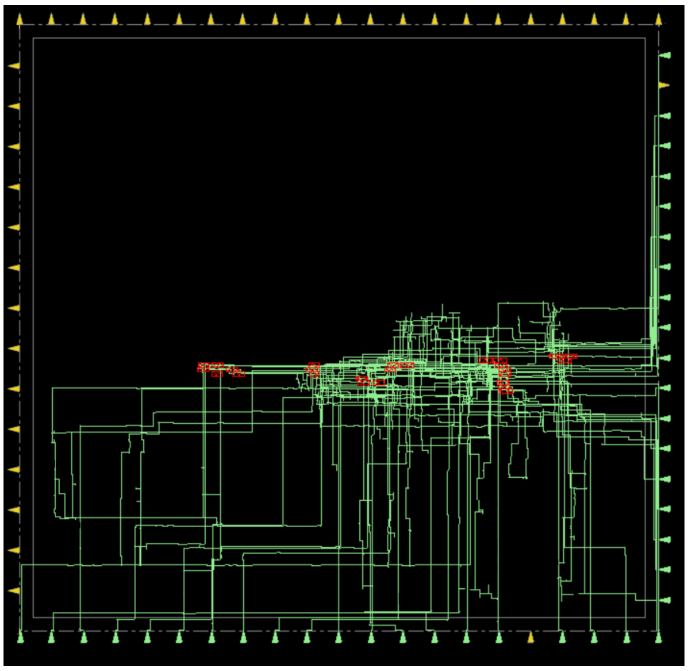


Figure 4.19: A layout of the AES-128 benchmark. The cells coloured in red are the inserted IRF-ETIs at selected internal and end-nodes.

for all input ports (excluding clocks and resets) of the benchmarks are listed in Table 4.5. The IRF coverage of input ports for the largest benchmarks, AES and s35485, can be obtained with 1.51 % and 1.94 % area overhead. On average, the proposed algorithm can provide 100 % IRF coverage for input ports at the cost of about 3.5 % area overhead.

An example of full IRF coverage for input ports of the AES-128 benchmark [AES19] is shown in Figure 4.19. It shows that the full IRF coverage for input ports can be reached by only nine IRF-ETIs. In addition, these nine IRF-ETIs can cover 22 % of the entire chip.

Figure 4.19 shows the resulting chip layout of the insertion flow for the AES-128 benchmark. In this figure, our selection algorithm has chosen the location and the number of IRF-ETIs. In total, nine IRF-ETIs have been selected to provide full IRF coverage (100%) for all input ports. The IRF-ETIs are coloured in red. The input ports, vias and interconnection wires which are covered by IRF-ETIs are coloured in green. The output, clock, and reset ports are coloured in yellow.

The node-selection algorithms for ageing detection are commonly based on the delay slacks of data paths in the design [Lai14, Ebr16c, Sad18]. We will refer to these algorithms as delay slack-based algorithms.

To compare our IRF-aware algorithm with one that is only based on delay slacks,

Table 4.6: IRF injection results of the AES-128 benchmark.

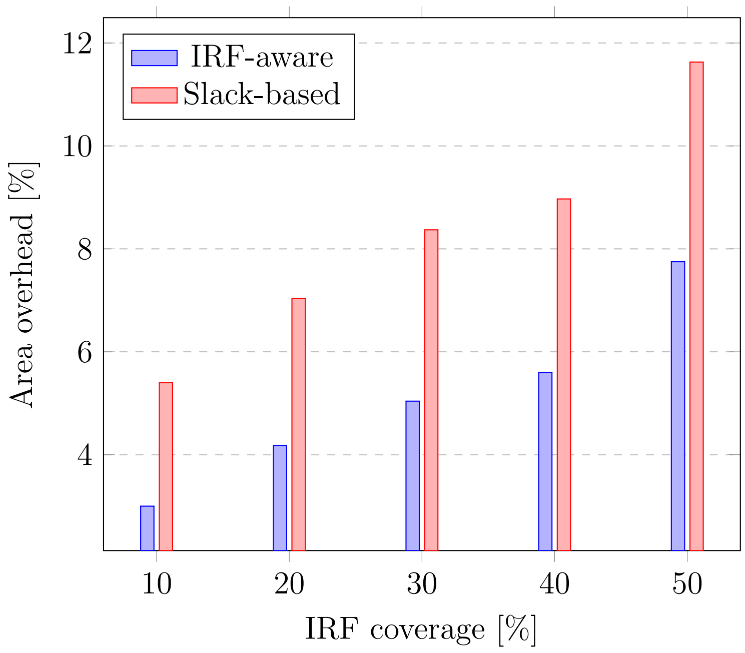
IRF	Percentage
Masked	32.5 %
Detected	38.2 %
Undetected	21.8 %
Logic Error	7.3 %

the algorithm presented in [Sad18] was implemented. The comparison between our IRF-aware algorithm and the delay slack-based algorithm in [Sad18] is shown for the two benchmarks s38417 and AES-128 in Figure 4.20. In the slack-based algorithm, the node selection is performed only based on the delay slack of each net. For example, it depicts that our IRF-aware algorithm can reach an IRF coverage of about 50 % for entire interconnections of the benchmark with an area overhead of 7.7 % employing our IRF-ETI Type 1. Whereas, the slack-based algorithm can provide the same IRF coverage with 11.6 % area overhead.

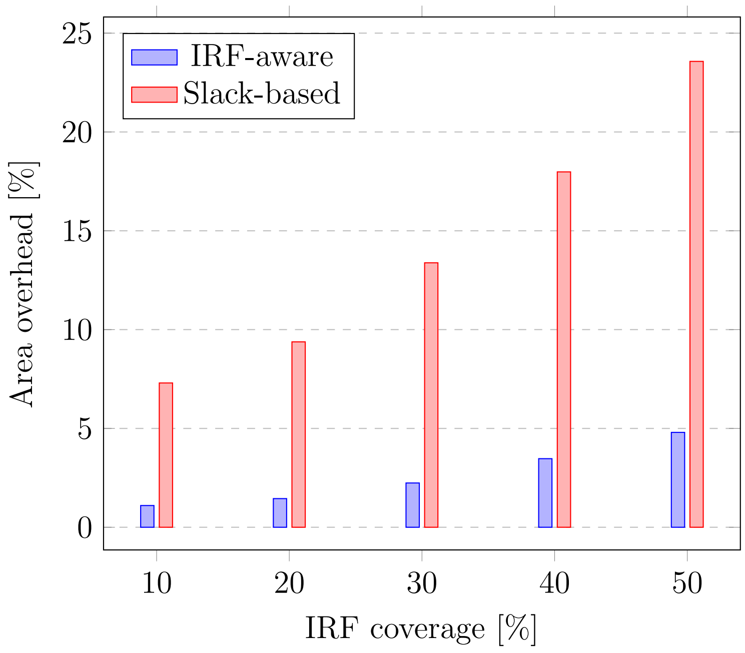
In general, our algorithm provides the same amount of total IRF coverage with about 3 % and 11.9 % less area overhead in comparison to the slack-based algorithm for benchmarks s38417 and AES-128, respectively. This indicates that our selection algorithm can provide better IRF coverage compared to delay slack-based algorithms. The reason is that our algorithm is IRF-aware. In contrast to the slack-based approach, our technique considers the probability of IRF propagation and IRF occurrences while selecting nodes.

A simulation-based IRF injection was used to evaluate the effectiveness of the proposed algorithm for IRF detection. The AES-128 benchmark was selected for IRF injection. A post-synthesis simulation was carried out using the QuestaSim tool [Que16]. The algorithm was executed on the AES benchmark with a 4.7 % area overhead constraint. The algorithm yielded a maximum IRF coverage of 50 % for the entire design (all ports and internal interconnections). A total of 10000 IRFs were injected into 1000 nets randomly selected from the benchmark. The results are shown in Table 4.6.

From all injected IRFs, about 7.3 % lead to *logic errors*. The rest of the IRFs did not cause any logic errors. These IRFs can be divided into three categories: masked, detected and undetected. The *masked* IRFs are defined as the IRFs that were masked by combinational logic and have not been propagated to any end node. In our simulation, about 32.5 % of injected IRFs were masked. From the IRFs that were propagated to at least one end-node, 38.2 % of IRFs were *detected* by IRF-ETIs. Around 21.8 % of faults have caused a slack-delay reduction in some end-nodes of the design, which were not selected for IRF monitoring. Therefore, the injected IRFs remained *undetected*, however, still did not cause a logic error.



(a) s38417 benchmark,



(b) AES-128 benchmark

Figure 4.20: Area overheads for a slack-based approach and our IRF-aware algorithm for benchmarks (a) s38417 and (b) AES-128.

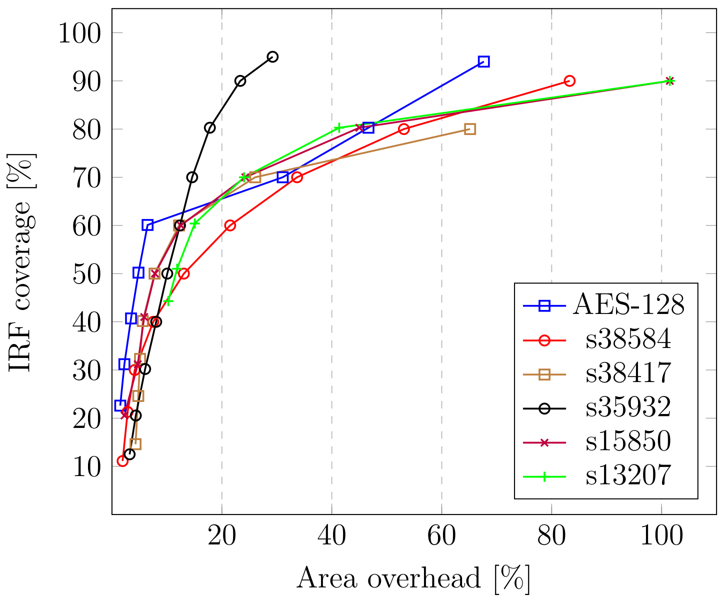


Figure 4.21: Maximum IRF coverage for six different benchmarks.

The maximum IRF coverage for the second selection strategy (node selection from all nodes) is shown in Figure 4.21. As can be seen, the maximum IRF coverage differs per benchmark. It highly depends on the net-list topology of the benchmark. For example, our algorithm can obtain a maximum IRF coverage of 95 % for s35932. This value for s38417 is 82 %. However, full IRF coverage for all nets in a design is not feasible. Therefore, the algorithm maximises the IRF coverage for connections that are more susceptible to IRFs, i.e. the input ports and the nets with many vias.

The simulation results reveal that the algorithm finds the best locations for IRF monitoring based on the propagation probability of each net of the system.

4.5 Periodic Test for IRF Detection

IRFs can intermittently change the propagation delay of interconnections by affecting their resistance and capacitance characteristics. Therefore, monitoring the propagation delay of an interconnection using an embedded test instrument enhances IRF detection at the interconnection during the operation of the system in the field. In order to measure the propagation in an interconnection, our IRF-ETI Type 1 can be employed. As shown in Section 4.2, IRF-ETI Type 1 was used to detect slack violations at the end of a data path. By connecting the *Clock* and *Data* inputs of IRF-ETI Type 1 to the two ending nodes of an interconnection, one can use this IRF-ETI to measure the propagation delay of the interconnection.

Figure 4.22 shows how IRF-ETI Type 1 can be reused to measure the propagation

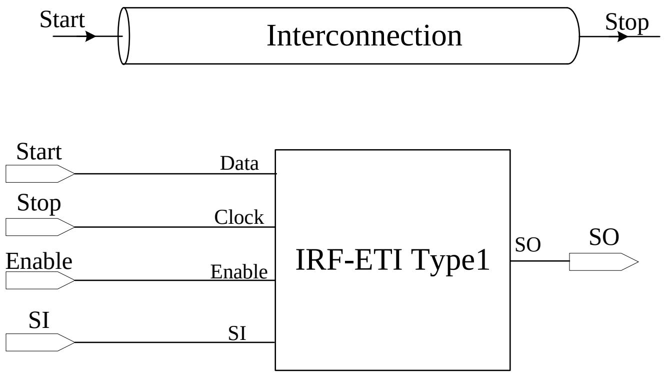
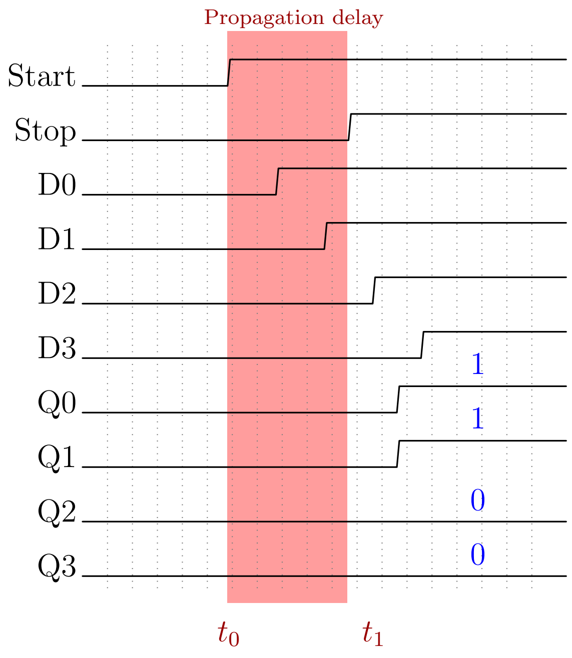


Figure 4.22: Reusing the IRF-ETI Type 1 introduced in Section 4.2 for periodic IRF testing.

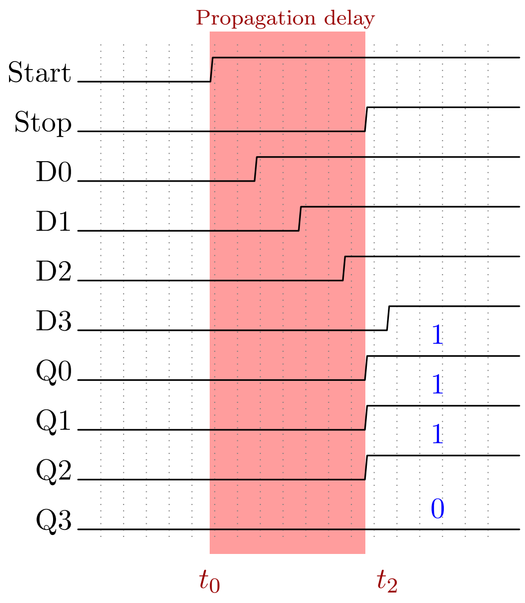
delay of an interconnection. The IRF-ETI measures the time interval between two rising-edge events on *Start* and *Stop* signals. The measured value will be stored in the IRF-ETI and accessed through an IJTAG network connected to *SI* and *SO* ports. The stored value indicates the magnitude of the time difference between rising-edge events on *Start* and *Stop* signals. The stored value in the IRF-ETI can be collected by fault-management software to monitor the health of the interconnections under testing. In the fault-free case, the collected value for an interconnection should remain constant. Note that because the test interconnections are part of the path under test, the IRF-ETI can also identify an IRF in them.

An example of an IRF detection by the IRF-ETI Type 1 composed of four D-flip-flops is shown in Figure 4.23. The measurement starts at the rising-edge of the *Start* signal. The *Start* signal undergoes a delay shift due to passing through the interconnection under test. The *Stop* signal represents the shifted version of the *Start* signal after passing through the interconnection. The timing difference between the *Start* and *Stop* signals is shown by a red-coloured window in Figure 4.23. The *D0*, *D1*, *D2*, and *D3* signals are time-shifted versions of the *Start* signal due to passing through the delay chain in the IRF-ETI. The measurement stops in the case the *Stop* signal becomes asserted. The *Q0*, *Q1*, *Q2*, and *Q3* signals represent the value that the IRF-ETI captures. In this example, the captured value is *Q3..Q0*="0011" in the fault-free case (see Figure 4.23a). The captured value in the IRF-ETI varies when the propagation delay changes due to IRFs (t_1 increased to t_2). In this figure, the value is changed into "0111" in the case of an IRF (see Figure 4.23b).

Designers can utilise this IRF-ETI for IRF detection at the chip level. An implementation of the proposed testing method based on the IRF-ETI Type 1 in a network-on-chip (NoC) [NoC20] is depicted in Figure 4.24. Figure 4.24(a) depicts a typical router of a NoC. It consists of a crossbar, multiplexers and input buffers. The router sends and receives packets to and from other routers as well as the processor connected to the router (shown by P). The inputs of multiplexers are *N*, *E*, *S*, *W* and *P*. These signals represent the input signals received from other routers located

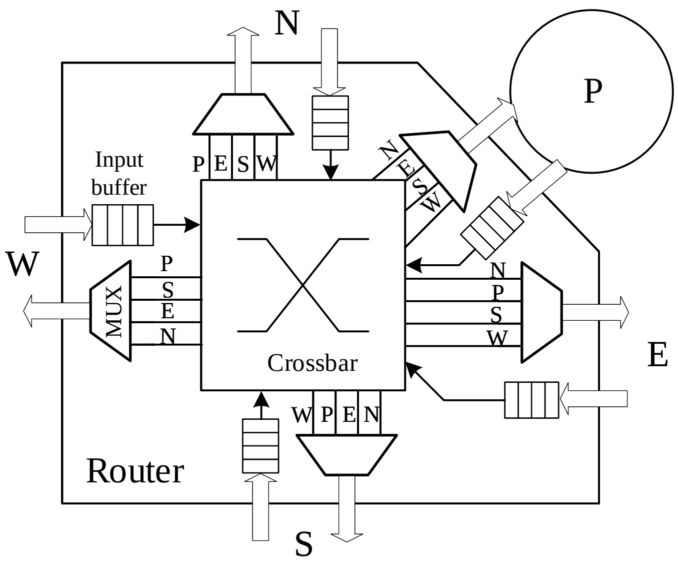


(a) Fault free,

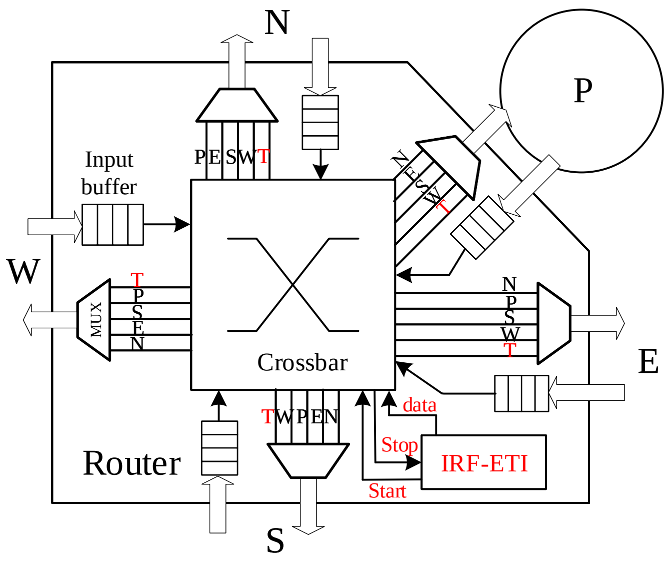


(b) IRF detected

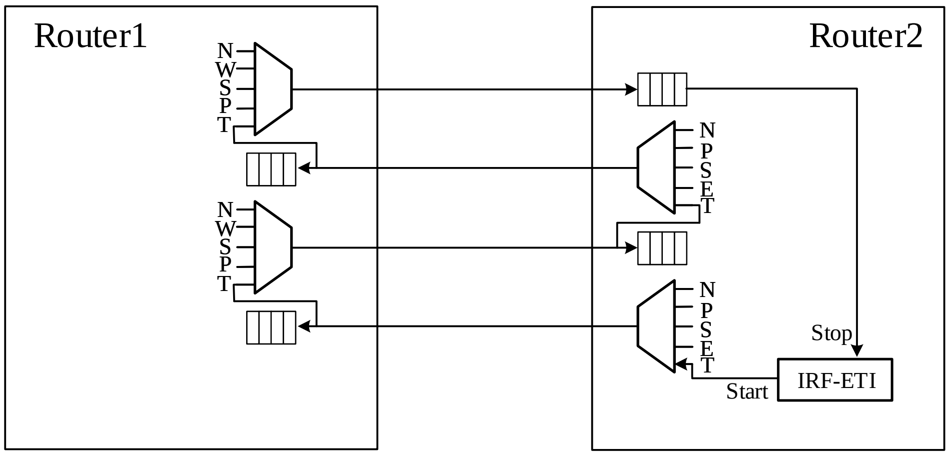
Figure 4.23: An example of a propagation-delay measurement with an IRF-ETI Type 1 with four D-flip-flops. a) Fault free b) IRF detected.



(a)



(b)



(c)

Figure 4.24: (a) A router architecture in a NoC. (b) A router equipped with an IRF-ETI Type 1 (c) An example of IRF detection for four interconnections between routers 1 and 2.

in the North (N), East (E), South (S) and West (W) of a router and the processor connected to the router, respectively.

Figure 4.24(b) shows how the router can be equipped with our IRF-ETI Type 1. For IRF monitoring in test mode, the multiplexers in the routers have been adjusted to form a chain of interconnections (both data and control). The input T indicates which signal should be transferred to the output in the case of the test mode.

An example of IRF monitoring for four interconnections between two routers is shown in Figure 4.24(c). It shows how an IRF-ETI can monitor multiple interconnections between two routers. The extra interconnections T, *Start* and *Stop* have been used to create a closed path for testing. The IRF-ETI can detect IRFs in the entire path, but no exact location diagnostics within the path.

To avoid performance loss, the testing phase can be activated periodically in the case the channel is idle. The captured values in the IRF-ETIs can be read via an IJTAG network. Alternatively, the values can be read by the processor connected to the router, or sent to a different processor for analysis. The captured data by the ETIs will be compared with the one captured previously. In the case of any discrepancy, a warning signal can be asserted.

Several different techniques are possible to design the delay chain of the IRF-ETI. There is a variety of implementations of delay chains in ASICs as well as FPGAs [Che18a]. Typically, using the clock of the system for sampling allows only nanosecond resolution, which is not sufficient for IRF measurements. Fortunately, there are several techniques to reach picosecond resolution. For example, standard gates in a chip (e.g. buffer, inverter, carry chain), routing elements and a phase-locked loop [Che18a] can be utilised to create the delay chain of the IRF-ETI with picoseconds sampling resolutions.

As shown in the previous section, selecting the best locations to insert IRF-ETIs depends on the structure and the layout information of the circuit under test. In the case of NoCs, an IRF-ETI can be inserted for each channel since they have symmetrical structures.

4.5.1 Case Study

Simulation-based fault injections have been used to study the effects of IRFs on NoCs in integrated circuits. For this goal, the proposed method has been implemented on an open-source NoC [NoC20] employing a 4x4 mesh topology, 16 routers and the XY routing algorithm. The data width of the channel was set to 8 bits. Each channel had a buffer with 4 bits to store input data. The design was synthesised using TSMC 40 nm CMOS technology. The area and power consumption per module are listed in Table 4.7.

Table 4.7: Synthesised summary in terms of area and power consumption for a 4x4 mesh NoC.

Design	**Area [μm^2]**	**Power [mW]**
NoC without IRF-ETI	30259	33.56
NoC with 24 IRF-ETIs	31197	34.39
One IRF-ETI with 3 FFs	32	0.004

For every channel, an IRF-ETI Type 1 with three flip-flops was used to cover all data and control interconnections in the channel. In total, 24 IRF-ETIs (one IRF-ETI per channel) were implemented to cover all channels between routers in the NoC. The area scale is square micrometres, and the power consumption is milliwatts. The dynamic power consumption was calculated based on a frequency clock of 1.45 GHz. As can be seen, the implementation of the IRF testing based on IRF-ETI Type 1 requires a small silicon area and power consumption overheads being 3.09 % and 2.47 %, respectively.

The QuestaSim tool was used for post-synthesis simulations. For every simulation, one interconnection between two routers was selected for the IRF injection. A total number of 50 IRFs were injected in every simulation. The IRF model was the same as used in [Ebr19]. IRFs with active times ranging from 1 ns to 10 ns and the potential to induce delays ranging from 5 ps to 15 ps were injected randomly. About 1500 data packets were produced randomly in every simulation, with random source and destination. The simulation execution time was 10,000 clock cycles. IRF detection was activated during the idle time of the corresponding channel. Every test required only 1.3 ns (2 clock cycles) to be completed.

The results of IRF detection for three different channels are shown in Figure 4.25. The figure shows the relation between the IRF detection rate and the number of cycles performed by the test. As can be observed, as the number of test cycles increases, the IRF detection rate rises. The results show that the IRF detection rate can reach 90 % in the case the testing execution cycles are 10 % of the total number of cycles.

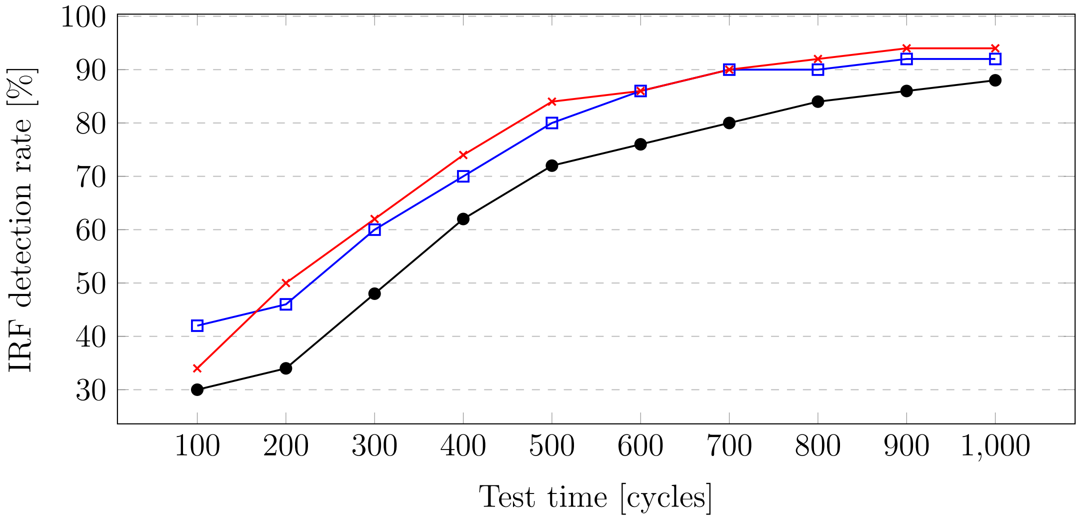


Figure 4.25: The relation between the IRF detection rate and test time for three different channels.

4.6 Summary

This chapter has focused on addressing research questions Q3 and Q4. Two embedded test instruments (IRF-ETI Type 1 and Type 2) were proposed for IRF monitoring and detection at the chip level. IRF-ETI Type 1 has a fixed detection timing-window, whereas IRF-ETI Type 2 has a programmable detection timing-window thanks to a programmable guard-band input. The proposed IRF-ETIs are scalable, fully digital, and compatible with the IJTAG standard. Simulation-based fault injections were employed to verify the capabilities of the IRF-ETIs for IRF detection. The simulation results validate the effectiveness of the IRF-ETIs in IRF detection. It was shown that the IRF-ETIs could differentiate between intermittent, ageing, and permanent faults.

In the second part, a node-selection algorithm was proposed for IRF monitoring at the chip level. The proposed algorithm tries to find the best locations that provide maximum IRF coverage based on the IRF propagation probability as well as the number of vias and interconnections of the design.

According to the simulation results based on the ISCAS 89 benchmarks, our algorithm can provide 100 % IRF coverage for input ports at the cost of about 3.5 % area overhead on average. Moreover, the results showed that having full IRF coverage for the entire design interconnections is very costly in terms of area overhead. However, the proposed algorithm can provide 50 % IRF coverage for the entire interconnection network of a design with 9.2% area overhead on average.

At the end of the chapter, a new periodic testing method based on our IRF-ETI Type 1 was proposed for on-line IRF detection at the chip level. An implementation of our testing method in a NoC was investigated as a case study using post-synthesis simulations. The simulation results reveal that IRFs can affect data and control signals in a NoC, and cause logic errors. About half of the injected IRFs resulted in logic errors. Moreover, the ability of the proposed testing method to detect IRF at the chip level was verified using simulations. The results showed that the IRF detection rate could be up to 90 % in the case the testing execution cycles are 10 % of the total number of cycles.

Chapter 5
Intermittent Resistive Fault Detection at Board Level

Abstract– *This chapter addresses IRF detection at the board level. Several on-line testing methods have been proposed to detect intermittent resistive faults in digital as well as analogue components at the board level. In addition, the potential for reusing our on-chip embedded test instruments (ETIs) for board-level IRF detection has been explored.*

First, the capability of the IEEE 1149.4 standard for IRF detection has been investigated. Then, our enhanced version of the IEEE 1149.4 standard was introduced to tackle IRF detection at the board level. It will be shown that the proposed method can detect IRFs in the interconnections with mixed-signals.

Second, the effects of IRFs on board-level data transmissions have been investigated. The on-chip ETIs presented in the previous chapter have been reused for IRF detection at the board level. As case studies, two widely used on-board transmission protocols, namely the Universal Asynchronous Receiver Transmitter (UART) and the Serial Peripheral Interface bus (SPI), have been selected. The experimental results show that the ETIs can be reused to detect IRFs at the board level.

At the end, the usage of the ETIs for periodic IRF testing has been investigated. Hardware-based fault injections and thermal cycling experiments on actual solder joints have been used to verify the effectiveness of periodic testing at the board level. The results show that periodic testing can be used to detect IRFs in solder joints.

In conclusion, this chapter has demonstrated the capability of the IEEE

Parts of this chapter were published in:

- [Ker15] H. G. Kerkhoff and H. Ebrahimi, "Detection of intermittent resistive faults in electronic systems based on the mixed-signal boundary-scan standard," in IEEE Asia Symposium on Quality Electronic Design (ASQED), pp. 77-82, 2015.
- [Ebr19] H. Ebrahimi and H. G. Kerkhoff, "A digital on-line monitor for detecting intermittent resistance faults at board level," in World Scientific Journal of Circuits, Systems and Computers (JCSC), vol. 25, no. 03, pp. 194000310-194000315, 2019.
- [Ebr21] H. Ebrahimi and H. G. Kerkhoff, "Embedded test instrument for intermittent resistive fault detection at chip level and its reuse at board level," in IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), pp. 75-80, 2021.

1149.4 standard and our enhanced version for IRF detection at the board level. It has also been shown that the on-chip ETIs can be reused for board-level IRF detection. Finally, the usage of periodic testing for IRF detection has been explored.

5.1 Introduction

The complexity of printed circuit boards (PCBs) continues to proliferate. The number of devices on a board, such as sensors, actuators and memories, increases. It leads to having more signals and data communications at the board level. The reliability of on-board communications highly depends on the reliability of interconnections. Furthermore, as the number of devices on boards grows, interconnections between them increase. This results in circuit boards with more layers of interconnections, thinner interconnects, and more vias. The reliability of interconnections is becoming increasingly important as they become more prevalent [Qi08].

Several causes of IRFs in PCBs have been reported in [Tho02]. For example, substrate cracks, open bumps, excess leakage currents at high temperatures, connector–ramp mating problems, lead frames unsoldered from housing, shorted lead frames, open leadfoots, cracked solder, cracked resistors, cracked capacitors, and wire bond lifts can cause IRFs in PCBs. Furthermore, IRFs can be caused by poor design, insufficient soldering, and improper cleaning. Furthermore, moisture, dust, and other contaminants can cause IRFs in PCBs. Finally, improper PCB handling and storage can result in IRFs.

In general, unstable and marginal interconnections are the most common causes of IRFs. Corrosion, electromigration, temperature and mechanical stress can contribute to increased interconnection instability. In contrast to the IRF at the chip level, corrosion and mechanical stresses have a more substantial role in causing IRFs at the board level. Moreover, human mistakes have a significant impact on causing IRFs. For example, loose connectors and cold solder joints can be mistakenly created by humans.

Similar to IRFs at the chip level, the behaviour of IRFs in PCBs is highly unpredictable. During the operation of a system, IRFs can occur intermittently in any interconnection. They may cause an intermittent delay in data and control paths due to increased resistance [Con03]. Therefore, IRFs can lead to timing errors or performance degradation. During the lifetime of a system, the IRF occurrence rate at specific locations may gradually increase and become increasingly severe. Eventually, they may evolve into a permanent fault [Yar15]. As a result, detecting IRFs before they become permanent and cause a system failure is critical, especially in safety-critical systems [Yar15].

In contrast to the chip level, after detecting the cause of IRF, the defected component or interconnection can be repaired or replaced in PCBs. Moreover, visual inspections such as laser and x-ray are more feasible and convenient at the board level.

On-line monitoring and periodic testing at the board level, like IRF detection at the chip level, are two common approaches to IRF detection. On-line monitoring is the continuous monitoring of a circuit using various embedded test instruments while it is in operation [Ker15]. Periodic testing, on the other hand, increases the likelihood of detecting IRFs by retesting the system regularly [Kra06]. In Chapter 4, IRF detection at the chip level was investigated, and two IRF embedded test instruments (IRF-ETIs) were proposed. In this chapter, IRF detection at the board level will be studied. It will also be demonstrated how IRF-ETIs can be used for on-line IRF monitoring and periodic IRF testing at the board level.

In the following, two methods for IRF detection are proposed, one for mixed-signal (analogue/digital) and another fully-digital. In the first method, one board-level testing standard, being the IEEE standard 1149.4 [Par16], has been enhanced to allow IRF monitoring. In the second method, the utilisation of on-chip IRF-ETIs for board-level IRF detection has been investigated.

The rest of the chapter is organised as follows. Section 5.2 shows to what extent IRFs can be detected using the IEEE 1149.4 standard. Then, our version of the IEEE 1149.4 standard enhanced for IRF detection is introduced in this section. In Section 5.3, the influence of IRFs in board-level data transmission has been studied. Moreover, reusing on-chip ETIs to detect IRFs in boards has been investigated with hardware-based fault injections. A periodic IRF testing method for IRF detection at the board level is introduced in Section 5.4. The chapter is concluded with Section 5.5.

5.2 IRF Detection Using Analogue Boundary Module

To detect IRFs, continuous (on-line/concurrent) monitoring is required of straight wires between inputs and outputs of digital ICs without disturbing the functional signals. Moreover, this task must be carried out in parallel for all relevant interconnections between the ICs.

The most likely candidate for carrying out the above task, the boundary-scan standards IEEE 1149.X, is briefly discussed in the following. It is shown that in the case of digital ICs on a PCB, still, the mixed-signal IEEE 1149.4 standard [Par16] forms the best basis for accomplishing the goal of IRF detection.

In this section, IRF detection using the boundary-scan IEEE standards 1149.4 has been investigated. First, the standard is briefly described. Then, using simulation, it will be shown how the standard can be enhanced to detect IRFs at the board level.

5.2.1 IRF Detection using Boundary-Scan Techniques

Because passive components are rarely designed into (digital) PCB interconnections, it is sometimes claimed that a digital boundary-scan is sufficient for pure digital ICs on a PCB. However, this is not true in the case of unwanted resistive faults in PCB interconnections. For directly determining resistive faults in interconnections, measurements are required, which can only be implemented via the mixed-signal boundary-scan variant. An advantage of IRF detection is that very accurate resistance measurement is not required; instead, only an increasing trend in resistance value must be monitored. Besides, small increments in resistance are already sufficient for IRF detection.

The IEEE 1149.4 standard [Par16] is a mixed-signal boundary-scan standard that enables the testing of digital and analogue components on a PCB. It is based on the IEEE 1149.1 standard [Ble11], which is a digital boundary-scan standard. The 1149.4 standard adds analogue test capabilities to the 1149.1 standards, allowing for the testing of analogue components such as resistors, capacitors, and inductors. In this chapter, an enhanced version of the 1149.4 standards is proposed to add the capability to detect IRFs on a PCB.

In IEEE 1149.4 standard, a set of test pins (TAPs) are used to access the boundary-scan cells of each IC on the board. These TAPs are connected to the boundary-scan cells of each IC on the board, allowing for the testing of each IC's inputs and outputs, the modules at the inputs and outputs of a chip are called Analogue Boundary Modules (ABM). The additional control block is known as Test Bus Interface Circuit (TBIC). For all details, the reader is referred to the IEEE boundary-scan standard [Wil00].

Due to the unpredictable occurrence of IRFs over time, an on-line monitoring scheme is a must. In the case of mixed-signal boundary-scan, this can be accomplished in principle using the well-known PROBE TAP instruction [Par16].

In a typical application of the 1149.4 standards, the analogue *AT1* and *AT2* buses are connected to (external) ATE resources. This approach is effective in detecting static resistive faults on boards. However, this is not practical in the case of on-line IRF monitoring. One can either connect embedded test instruments (ETI) in the chips to the bus or include elementary resources (e.g. on-chip current sources, voltage comparators) using transmission gates to the buses in the ABM like in this chapter.

ATE resources are no longer required in both cases.

An upgrade of the standard has been discussed in [Han12]. In two papers [San09, Lin14], the sinus output response was suggested for better characterisation, while the second paper introduces precision on-chip measurements. However, the authors only study static (resistive) faults, not intermittent ones.

On the other hand, another complication in the case of IRFs is that, in principle, every PCB interconnection between chips should be included. Therefore, the need for multiple observations of the different interconnection lines must be considered. This could be restricted to fewer lines if the probability of IRF occurrences is taken into account (e.g. many soldering contacts in the interconnection). Nevertheless, this will have consequences on the (A)TAP controller instructions [Par16] as in our case, a selected number (not a single ABM) of ABMs have to operate at the same time. As a consequence, the controller has to be upgraded to this requirement.

As can be expected, there will be limitations concerning detecting the smallest durations of resistive pulses. However, thermal cycling experiments in Chapter 3 have revealed that, besides increasing resistance value, an increase in the duration of IRF pulses has been noticed. Clearly, this intermittency additionally complicates the situation. This vital issue will be treated in Section 5.2.3.

5.2.2 The Proposed Analogue Boundary Module (ABM)

Two simple digital chips have been used for the case study, each being an inverter. Each respective output and input are connected via mixed-signal ABMs. The generic scheme of our approach is depicted in Figure 5.1. It shows two PCB interconnection lines from Chip A to Chip B, which can be potentially disrupted by IRFs (shown by variable resistances). The ABMs are described in detail in [Ble11, Par16].

The voltages are monitored using appropriate V_{th} values for the 1-bit digitisers [Bus04]. Their data (larger or smaller than a specific voltage) is stored in the registers part of the TDI and TDO registers chain (shown by the blue line). Apart from the required functional interconnections, only the analogue buses, control lines, (A)TAP lines [Par16, Bus04], and TDI/TDO line are present between Chip A and Chip B. We have enhanced the TBIC block (highlighted in green), replacing the ATE in the conventional IEEE 1149 standard. It contains a programmable current generator or programmable voltage evaluator in the ABM.

The simplified basic scheme used for our simulations, which has been derived from the Cadence schematics, is depicted in Figure 5.2.

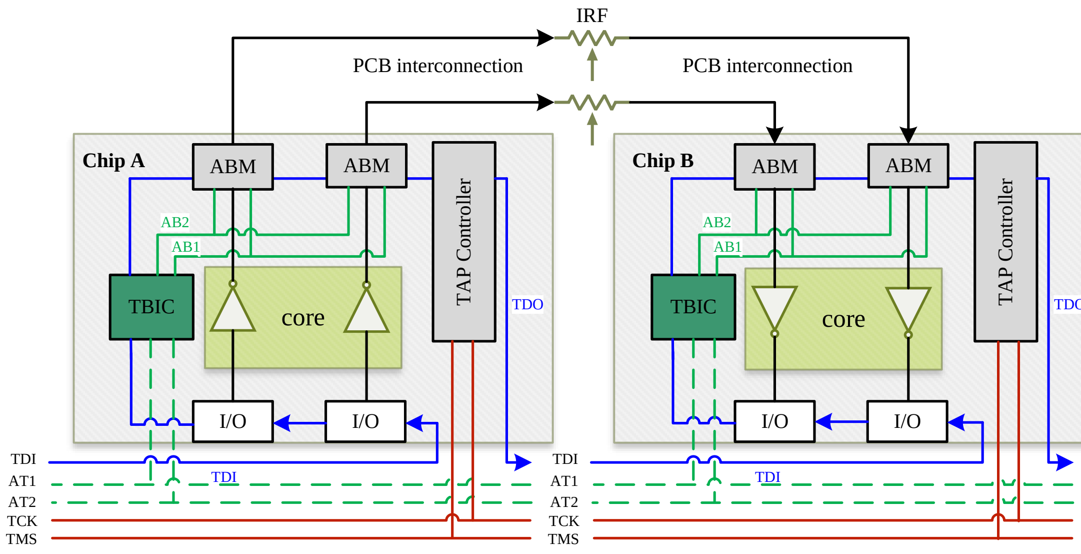


Figure 5.1: Generic path with IRFs between two digital integrated circuits (Chip A, Chip B) interconnections on a PCB.

The circuits operate at a clock frequency of 200 MHz (5 ns). All transistor implementations using the Nangate 45 nm CMOS technology [Nan08] are provided at a lower hierarchical level. The transmission gates were optimised for low on-resistance [Nov08].

In our transistor implementation, the 1-bit digitisers [Par16, Bus04], are three in reality in our case (Figure 5.2), and have been implemented as full-custom analogue comparators. To keep the example simple, the internal logic in both chips is just a single inverter library cell. The principle used for determining the resistance (arrow) is straightforward: injecting an appropriate current (in our case $30\mu A$) from a current source and measuring any voltage change in the transmitted signal. Note that our approach differs from the one commonly used with the standard, as the logic functionality must be preserved.

For measuring the voltage changes, comparators with references have been used to distribute the outcome via the TDI/TDO registers. This is a different approach than the one suggested in [Jef05]. Special care has been taken not to disrupt the actual functional logic behaviour.

The resolution of our design depends on several parameters. Time resolution depends on the hold time of the latch used for detection. The resolution of the resistance depends on the amplitude of the current source and the value of the reference voltage V_{th}, and the sensitivity of the ABM comparators.

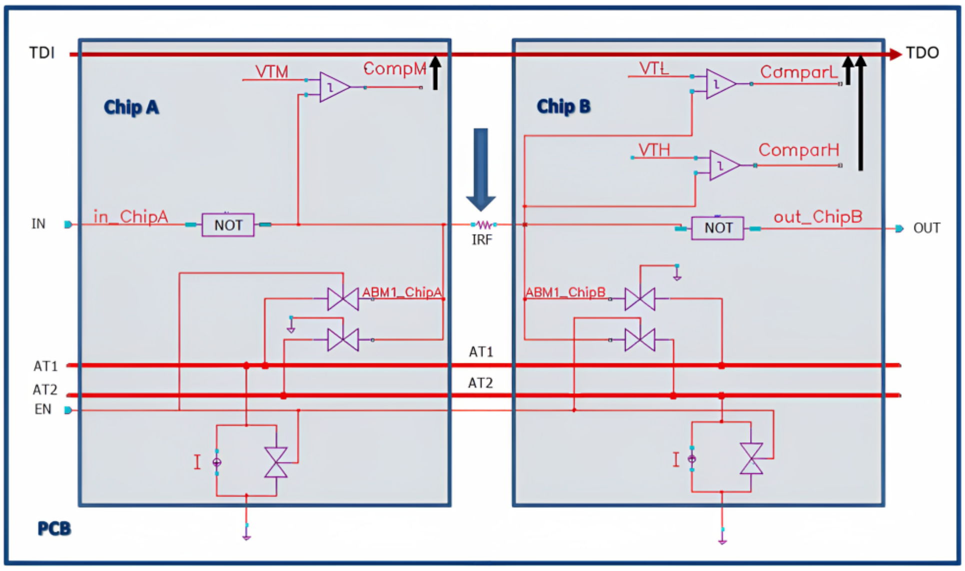


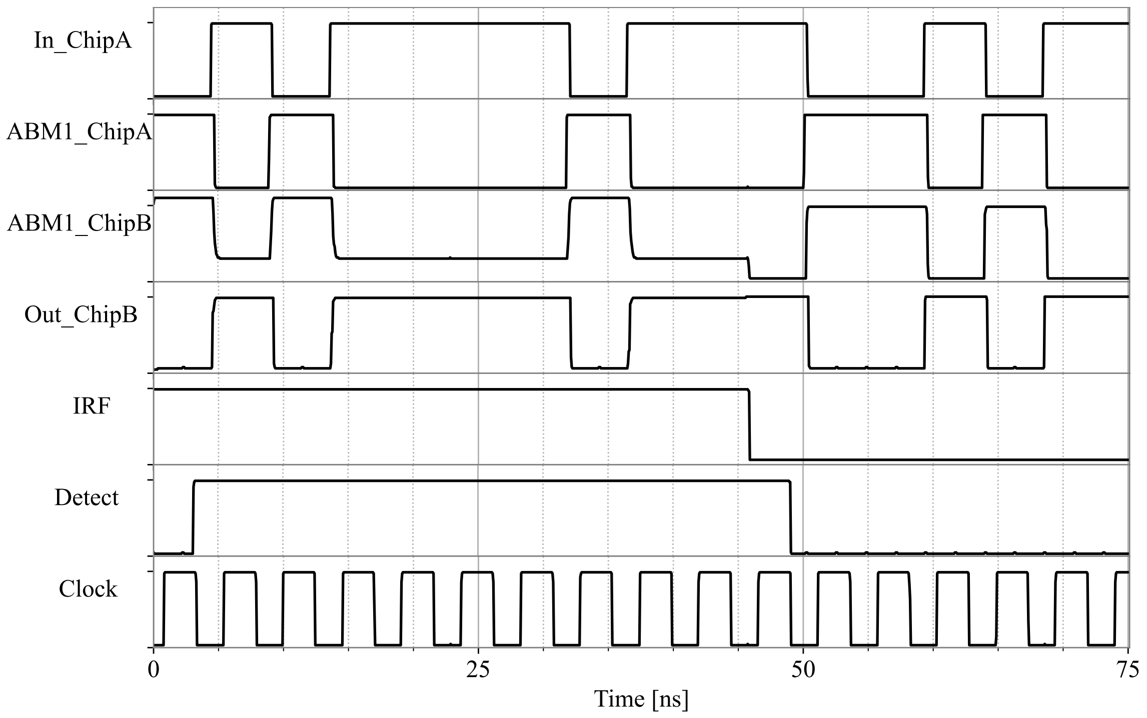
Figure 5.2: Cadence simplified simulation scheme of two integrated circuits (inverters, modified ABMs) and a single line resistive fault generator (thick blue arrow).

To determine the resolution of the design, the minimum resistance and active time that the design can detect should be determined. First, the resolution of resistance detection has been investigated using simulations. For this purpose, IRFs with large pulse widths (active times) have been injected. The injected IRFs consist of pulses that are active for several clock cycles.

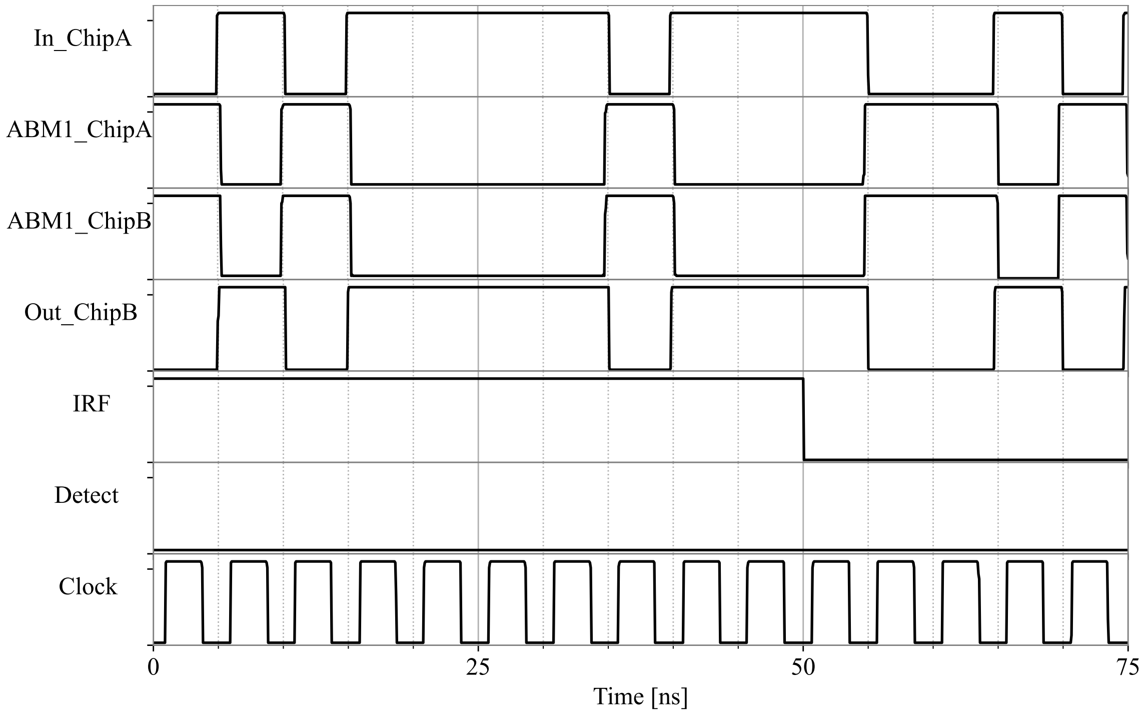
An example of IRF detection with high resistance (10 $k\Omega$) is shown in Figure. 5.3a. In contrast, Figure 5.3b shows the case that the injected IRF remains undetected. Table 5.1 lists all signals that can be seen in Figures 5.3 and 5.4. The input signal of Chip A is shown with *In_Chip A*. The *ABM1_Chip A* signal shows the input signal after passing through the internal logic (here, an inverter) and the ABM of Chip A. The *IRF* signal indicates when the fault is injected into the interconnection between Chip A and Chip B. The *ABM1_Chip B* signal represents the transmitted data after distortion and serves as the input signal for Chip B.

As can be seen in Figure 5.3a, the IRF caused the *ABM1_Chip B* signal to have a voltage shift during 0 and 46 ns. This voltage shift has been detected by the ABM. On the other hand, Figure 5.3b depicts the case where an IRF with a small resistance (1 $k\Omega$) is not being detected using the same current source.

The value of the current source determines the minimum resistance detection. The simulation results showed that the new ABM using a current source of 30 μA can detect IRFs with resistance higher than 1 $k\Omega$.



(a) detection of a 10 $k\Omega$ resistive fault, employing a 30 μA current source.



(b) non-detection of a 1 $k\Omega$ resistive fault, employing a 30 μA current source.

Figure 5.3: Fault simulation of two chips under the influence of a resistive fault in the interconnection line between them.

Table 5.1: Signals shown in the simulation results of Figures 5.3 and 5.4.

Signal name	**Explanation**
In_Chip A	Functional input signal inverter chip A
ABM1_Chip A	Output signal of the ABM1 of chip A
ABM1_Chip B	Input signal of ABM1 of chip B
Out_Chip B	Functional output signal inverter chip B
IRF	IRF signal for injecting intermittent resistive faults
Detect	Signal showing possible detection IRF
Clock	The chip system clock

5.2.3 Simulation of on-board IRF detection

In the previous subsection, the minimum resistance that can be detected by the new ABM was determined. From the simulations, it was concluded that our setup can detect IRFs with resistance larger than 1 $k\Omega$.

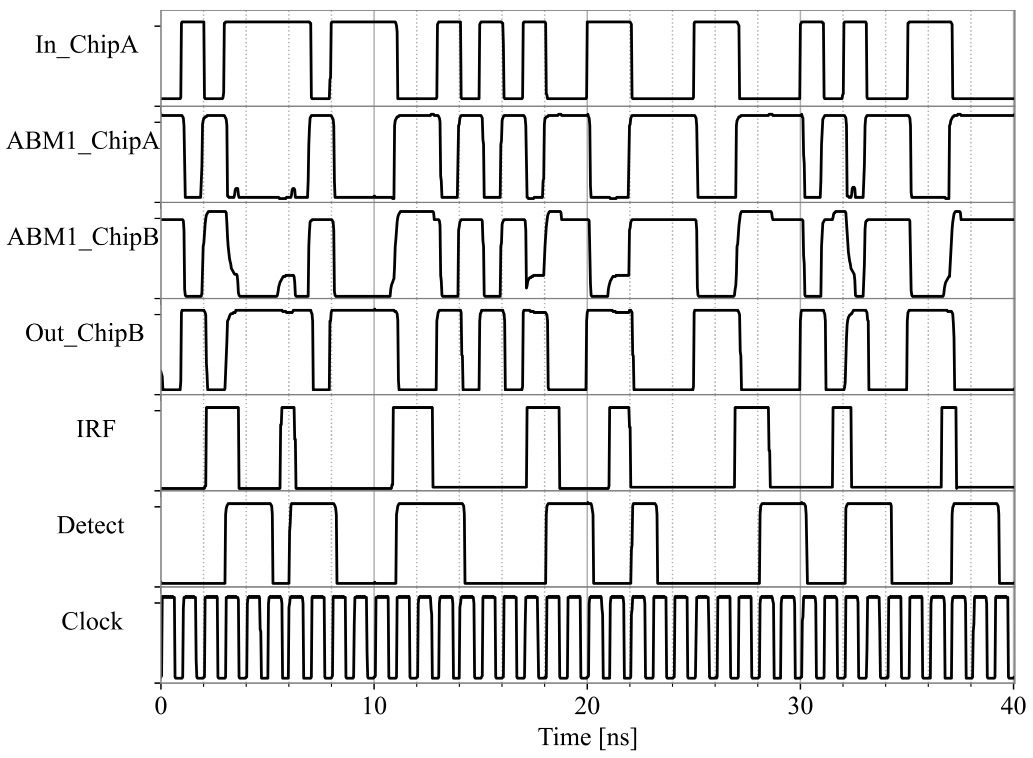
To determine the minimum active time that ABM can detect, IRF injection was conducted with random active times. The IRFs with resistances larger than 1 $k\Omega$ were chosen, to eliminate undetectable IRFs with low resistances from the simulation. The simulation results showed that the new ABM can detect IRF with active times larger than 0.5 ns.

Two examples of IRF injection are shown in Figures 5.4a and 5.4b. In Figure 5.4a, an IRF with a burst of 8 pulses was assumed. The active time of pulses was chosen at random between 0.5 and 2 ns. In addition, the resistance of pulses was chosen at random between 1 $k\Omega$ and 2.5 $k\Omega$. The explanation of the signals is listed in Table 5.1. The *Detect* signal shows that the ABM can detect all the injected pulses.

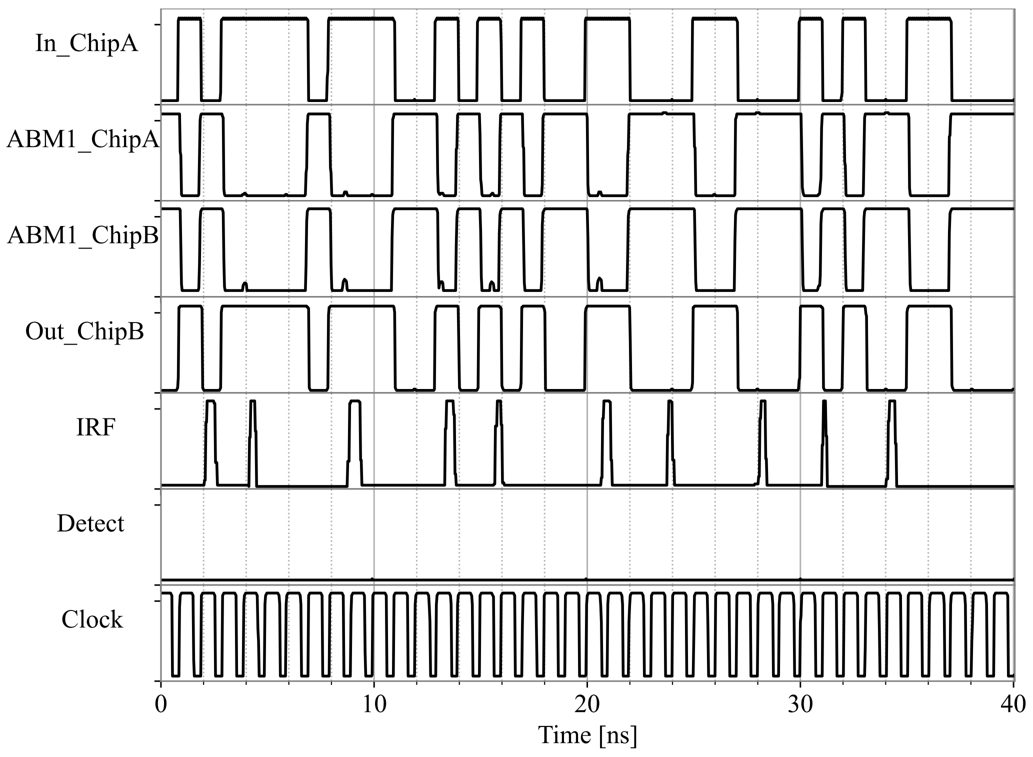
Figure 5.4b shows the simulation of an IRF consisting of pulses with an active time ranging from 0.02 to 0.49 ns. The injected IRF consists of a burst of 10 pulses. All of the pulses remained undetected, as indicated by the *Detect* signal.

It is emphasised that IRFs are a progressive category of faults; pulse durations tend to extend as a function of time (ageing). As long as detection takes place before the actual functional failure, it is sufficient for our dependability purpose.

In terms of overhead, two simple current sources, multiplexers, and an additional comparator were added per two ABM (library) cells.



(a) Detection of IRFs with active times of 0.5 ns and larger.



(b) non-detection of an IRF with active times less than 0.49 ns.

Figure 5.4: Simulation results of IRFs employing the proposed ABM.

5.2.4 Conclusions on Analogue Boundary Module

In this section, we have investigated to what extent 1149.4 can detect IRFs for PCB-level implementation of electronic systems. The mixed-signal boundary-scan standard, the IEEE 1149.4, was naturally considered for on-line detection of IRFs in boards, and enhancements have been provided to deal with them. Possible solutions for the problematic case of very short pulses have been provided and validated by simulations.

5.3 Reusing On-chip IRF-ETIs for Board-level IRF Detection

In this section, the ability to use our on-chip IRF-ETIs (introduced in Chapter 4) for detecting IRFs in boards has been investigated. Two widely used on-board transmission protocols, UART and SPI, were chosen as case studies.

5.3.1 Fault Management Framework

The stored data in the flip-flops of the IRF-ETI will be read by a fault-management framework via an IJTAG [IEE14] network. Based on the degree of the timing violation, the fault location, and its occurrence rate, the fault-management software can classify the detected fault type. It determines whether it is caused by a transient, intermittent or permanent fault. In short, if, during a specific period, several faults with various degrees of timing violation occur in a line, it can be classified as an IRF. If only one timing violation occurs, it can be classified as a transient fault. Finally, it would be considered a permanent fault if the number of violation occurrences is higher than a certain chosen threshold.

5.3.2 Intermittent Resistive Faults Model

The IRF model and generation used here are based on what has been introduced in Section 3.2. Table 5.2 lists the values and distributions that have been used for the fault injection experiments in this section.

5.3.3 The Fault-injection Framework

The fault-injection framework is composed of fault-generator software and fault-injector hardware. The fault-generator software runs on a personal computer and

Table 5.2: Range of used parameters in the IRF generator.

Parameter	Minimum	Maximum
Burst length	1	5
Resistance	1.0 Ω	2.5 $k\Omega$
Start time	0.1 μs	2.0 μs
Active time	0.2 μs	1.5 μs
Inactive time	0.2 μs	1.0 μs
Safe time	1.0 μs	20 μs

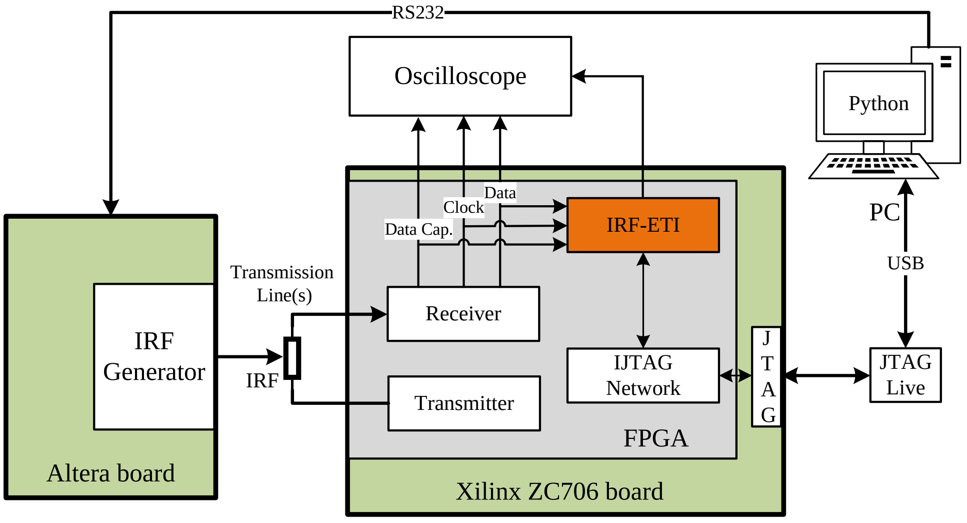


Figure 5.5: A block diagram of the fault-management and injection frameworks for IRF injection and detection experiments.

generates IRFs based on the IRF model. The generated IRF can be transferred to the fault-injection hardware via a serial communication link. The fault-injection hardware [Ebr16a] is composed of a custom printed-circuit board and an FPGA board (see Figure 3.18). The FPGA accurately controls some fast digital switches and potentiometers on the printed-circuit board to generate the sequences of IRFs. Figure 5.5 shows how fault management and injection frameworks have been used for fault injection and measurement.

5.3.4 Experimental Setup

To evaluate the usage of our on-chip IRF-ETIs (see Sections 4.1 and 4.2) for board-level IRF detection, two widely used serial protocols for on-board communication, namely UART [Fan11] and SPI [SPI03], were used as case studies. The transmitters and receivers for both protocols were designed at the logic gate level and implemented on an FPGA. The experimental set-up of the fault injection and monitoring is the same as shown in Figure 3.18. Two FPGA evaluation boards were utilised in this set-up: an Altera DE1 board for the fault generator and a Xilinx ZC706 board for the transmitter and receiver, as well as the IRF-ETI and IJTAG networks. It should be mentioned that the transmitters could be implemented on a separate board, but for simplicity, they were implemented on the same board as the receivers. As can be seen, the fault generator was connected to the transmission line in the middle. In practice, the IRF generator was connected to the transmission line in series. Through this construction, we could emulate a transmission line with IRFs. In our experiments, the transmission baud rate and power supply were set to 3 MHz and 2.5 V, respectively.

IRFs were injected into the transmission line by the fault-injection framework and simultaneously monitored by the fault-management framework. The parameters used by the fault-injection framework to generate the injected IRF are listed in Table 5.2. The injected pulses were randomly generated with resistances ranging between 1 Ω and 2.5 $k\Omega$. This range is relatively small and less likely to result in logic errors in the data transmission line, but can still cause timing distortions. Therefore, it is suitable for evaluating the IRF-ETI.

In the case of UART, IRFs were injected into the transmission line between the transmitter and the receiver by our hardware IRF generator [Ebr16a]. In the SPI experiment, fault injection was executed in all four signal lines clock $SCLK$, chips selectCS, master-output slave-input $MOSI$ and master input slave output $MISO$, while the receivers were equipped with IRF-ETI Type 2 (see Figure 4.5).

Generally, serial transmissions can be protected from logic errors by adding parity and framing checks. In UART and SPI communications, the even-parity check was employed. A parity error can occur when an odd number of bits is flipped during a transmission. In addition, a framing check is implemented by counting the number of received bits in each data frame. A frame error occurs when the amount of received bits in a frame does not match the predefined frame size. Our simulation results reveal that due to IRFs, many logic errors can occur in a serial transmission that can not be detected by parity and frame checking. This section aims to discover timing violations caused by IRFs before they cause logic errors. IRFs have been injected into the transmission line by the fault-injection framework and simultaneously monitored by the fault management framework. Table 5.2 shows the parameters used by the fault-injection framework to generate the injected IRF. The injected resistance ranged from 1 Ω to 2.5 $k\Omega$. This range was relatively low and hence less likely to cause logic errors while inducing timing violations in the transmission line. A frame error occurs in the case that the number of received bits of a frame is not matched with the default frame size. An example of an IRF injection in a UART transmission is shown in Figure 5.6. The time scale is 0.8 μs, and the voltage scale is 2 V. In this

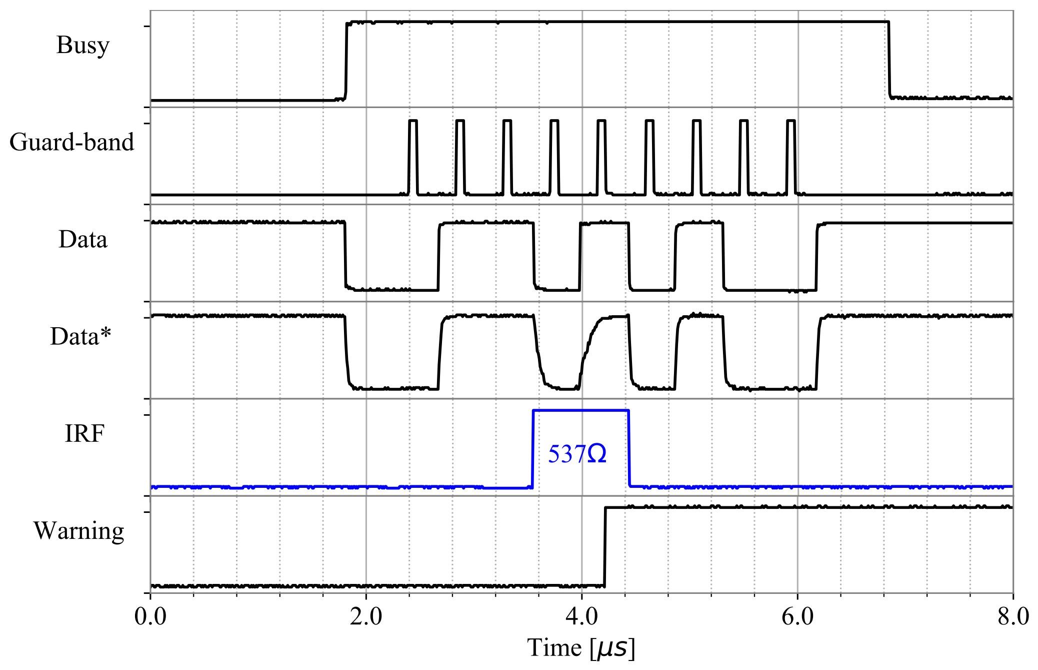


Figure 5.6: Measured results of the IRF-ETI Type 2 under hardware-based IRF injection for the UART protocol.

figure, the *Busy* signal shows when the UART receiver is receiving data. *Data* and *Data** show the bitstream received by the receiver before and after fault injection in the transmission line, respectively. Every frame of data in a UART transmission starts with a start bit, continues with eight data bits and ends with an even parity bit and two stop bits. Data bits are transferred from the less significant bits to the most significant bits. The *IRF* indicates amplitude, duration, and the timing of the IRF injected IRF. The *Guard_band* signal shows when the IRF-ETI Type 2 is active and detects all violations happening at that timing window. As shown in Figure 5.6, the IRF-ETI has detected the timing violation and has raised the *Warning* signal. The detected IRF was active for 1.1 μs, with a 537 Ω resistance value. The degree of violation is not represented in this figure, but it is stored in the IRF-ETI flip-flops with the value *0111*. It should be noted that the IRF has not caused a logic error, and the sent bitstream *10101100* was transferred intact without parity or frame error. However, the IRF-ETI detected the timing violations caused by the injected IRF. This ability of the IRF-ETI can help a system avoid future failure.

Figure 5.7 shows an example of an IRF injection using the SPI protocol. An SPI protocol was implemented by a master and a slave with four signal connections. Each data frame starts with the less significant bit of eight bits and ends with one (even) parity.

In Figure 5.7, signals *CS*, *SCLK* and *MOSI* are chip select, SPI clock and Master Output Slave Input, respectively. *MOSI** shows the *MOSI* signal after IRF injection.

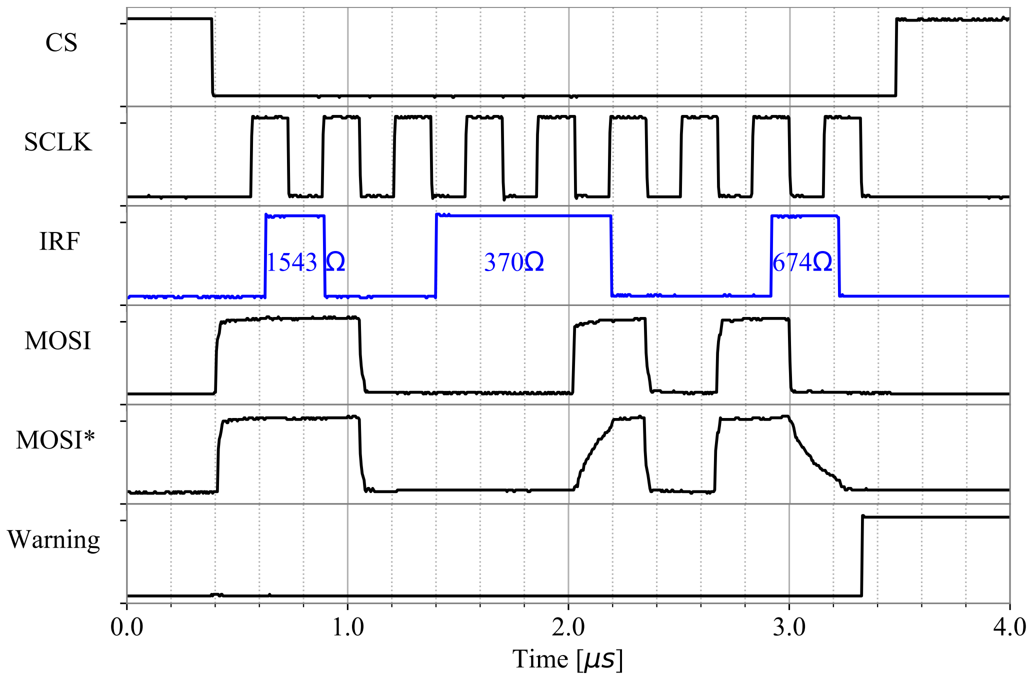


Figure 5.7: Measured results of the IRF-ETI Type 1 under hardware-based IRF injection for the SPI protocol.

The *Guard-band* signal is similar to the previous experiment and is not shown in this figure for simplicity. The *IRF* signal indicates the injected IRF by the IRF generator in the *MOSI* line. It consists of three pulses with different amplitudes and durations. The amplitudes are 1534 Ω, 370 Ω and 674 Ω, respectively. The first pulse occurred during 560 ns and 820 ns, but since there was no transition in the *MOSI* signal at this duration, the injected IRF did not cause any timing violation. The second pulse was active between 1400 ns and 2180 ns. As can be seen from the figure, this pulse has distorted the *MOSI* signal. Hence the rising edge of the *MOSI** signal has increased. Although the injected IRF resulted in a timing violation at this duration, the induced change was insufficient to be detected by the IRF-ETI. Finally, the last pulse of the IRF induced a significant timing violation. The IRF-ETI has detected the violation, and the *Warning* signal became "1".

Our experiment results show that our IRF-ETIs (Figure 4.4) can detect timing violations during UART transmissions due to IRFs with amplitudes larger than 470 Ω. In the case of SPI transmissions, several fault injections have been carried out regarding the clock (SCLK) and chip select (CS) signals. It can be concluded that the clock and chip select signals are more sensitive to IRFs than the data signals (*MOSI* and *MISO*). Moreover, the results showed that the timing violations induced by IRFs in the *MOSI* line with amplitudes larger than 500 Ω and durations more than 100 ns could be detected by our IRF-ETI. IRFs with resistance larger than 1800 Ω can result in a parity or frame error.

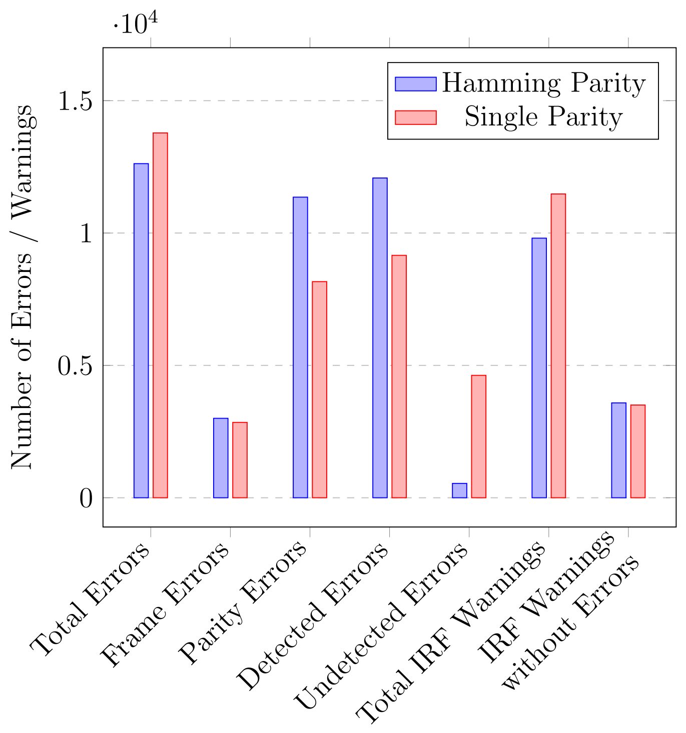


Figure 5.8: Fault injection results for cases of Hamming and single parities error detection and IRF-ETI warning detection.

Serial transmissions can be protected against logic errors by framing checks or adding parity bits. The vulnerability of two frequently used error-checking techniques, single parity and Hamming (7,4) coding, against IRFs has been explored in this section. A single-parity mechanism can detect an error when an odd number of bits are flipped during a transmission. A Hamming (7,4) error check mechanism is similar to the single-parity mechanism, but it utilises three parity bits for four data bits instead of one parity for a byte. Besides the parity and Hamming mechanisms, a framing check was implemented in our experiments. A frame error occurs when the number of received bits in a frame does not match the predefined frame size.

For each protocol, 20,000 fault injections have been carried out on random data transmissions. Figure 5.8 shows the result of the UART protocol transmission for two cases of single parity and Hamming (7,4) error-checking mechanisms in the presence of our IRF detection. It should be noted that the result for the UART and SPI protocols are similar, but for the sake of brevity, only the result for the UART protocol has been shown here. In Figure 5.8,*Total Errors* shows the total amount of logic errors that have been caused by IRF injection. A logic error occurs in the case the transmitted data is not the same as the received data. This value is calculated by comparing the transmitted data and the received data after simulation.

Of these total errors, some have been detected by the error detection mechanism, and some remained undetected. These values are shown as *Detected Errors* and *Undetected Errors.*

In the case of the Hamming mechanism, 95 % of the total number of logic errors were detected by either parity or frame error check or both. This ratio is 66 % for the single-parity mechanism. As expected, the Hamming mechanism can detect more logic errors than the single-parity mechanism. Because of the limitation of the parity mechanism, some logic errors remain undetected. The ratio of undetected logic errors is 5 % and 34 % for Hamming and the single-parity mechanism, respectively.

As mentioned earlier, even with parity and frame checking, some logic errors may remain undetected during serial data transmission. Using the IRF-ETI beside the error-detection mechanisms allows the serial receivers to detect IRFs at early stages before they lead to undetected logic errors. In Figure 5.8, the value of *"Total IRF Warnings"* indicates that the IRF-ETI could detect a timing violation in more than 50 % of the total number of the IRF injections. The value of *"IRF Warnings without Errors"* is essential. It shows how often the injected IRFs have caused timing violations in the transmission line, yet without causing any logic (parity or frame) errors. This especially occurs at the early stages of IRFs. Almost 25 % of our IRF injections consisted of early stages (with resistances below 1.2 KΩ). The IRF-ETI has detected all the injected IRFs with resistance values higher than 0.46 KΩ and duration longer than 0.1 μs, which were not detected by any coding. Those IRFs with resistance values exceeding 1.2 KΩ caused either parity, frame, or both errors.

To accurately estimate the area and power overhead of our design, all modules have been implemented at the logic-gate level and been synthesised using TSMC 40 nm technology. The area and power consumption per module are listed in Table 5.3. The area is measured in square micrometres, and the power consumption is measured in microwatts. The dynamic power consumption has been calculated based on a clock frequency of 50 MHz. As can be seen, the IRF-ETI has a small area and power consumption overhead. For the UART and SPI protocols, the area overheads of inserting the IRF-ETI and IJTAG network are 3.6 % and 3.3 %, and the power consumption overheads are 2.9 % and 6.8 %, respectively.

5.3.5 Conclusions on reusing IRF-ETI at board level

This section presents a fully digital embedded test instrument to address one of the most challenging interconnection reliability problems: intermittent resistive faults at the PCB level. Our IRF-ETI (Figure 4.4) has been evaluated by actual hardware using our fault-injection and fault-management frameworks. The experimental results show that the proposed ETI can detect IRFs at the board level in their early stages

Table 5.3: Area and power consumption overheads for all modules.

Modules	**Area [μm^2]**	**Power [μW]**
UART RX	606.3	15.9
UART TX	512.0	15.1
SPI Slave	418.6	1.7
SPI Master	805.0	11.6
IRF Monitor	40.2	0.9
IJTAG Network	329.8	6.4
TAP Controller	1003.1	6.1

with a small area and power overhead before they lead to logic errors.

5.4 Periodic Test Method for IRF Detection

As explained in Section 4.5, IRFs can change the propagation delay of an interconnection. Two IRF-ETIs were proposed to monitor interconnections propagation delay in chips. This section investigates the reuse of the on-chip IRF-ETI for IRF detection at the board level.

Figure 5.9 shows a simplified example of reusing our in-chip IRF-ETIs for IRF detection on input ports of a chip and interconnections on a PCB. This figure shows how multiple IRF-ETIs can be employed to detect IRFs concurrently in selected interconnections on a printed-circuit board (PCB). Board designers should design multiplexers or switches to facilitate the selection of interconnections for monitoring on the board. The *IRF-ETI controller* starts the IRF monitoring by first asserting an *Enable* signal and then triggering the *Start* signal. The IRF-ETI controller is only a simple FSM module responsible for generating the *Start* signal and enabling IRF monitoring. It should be noted that adding extra signals for path selections on the board (e.g. *Enable* signals) does not make the design more vulnerable to IRFs, since the IRF-ETI will detect IRFs on these signals.

The captured values in the IRF-ETIs can be read by fault-management software running on the CPU through an IJTAG network. The fault-management software receives captured data by the IRF-ETI, compares them with the previously captured data, and creates a warning in the case of discrepancies.

Several different techniques are possible to design the delay-chain part of the IRF-ETI. There are various implementations of delay chains in ASICs [Eno18, Ali19] as well as FPGAs [Wan17, Che18a]. Typically, using the system clock for sampling allows only nanosecond resolutions which are insufficient for IRF measurements. Fortunately, there exist several techniques capable of achieving picosecond resolutions.

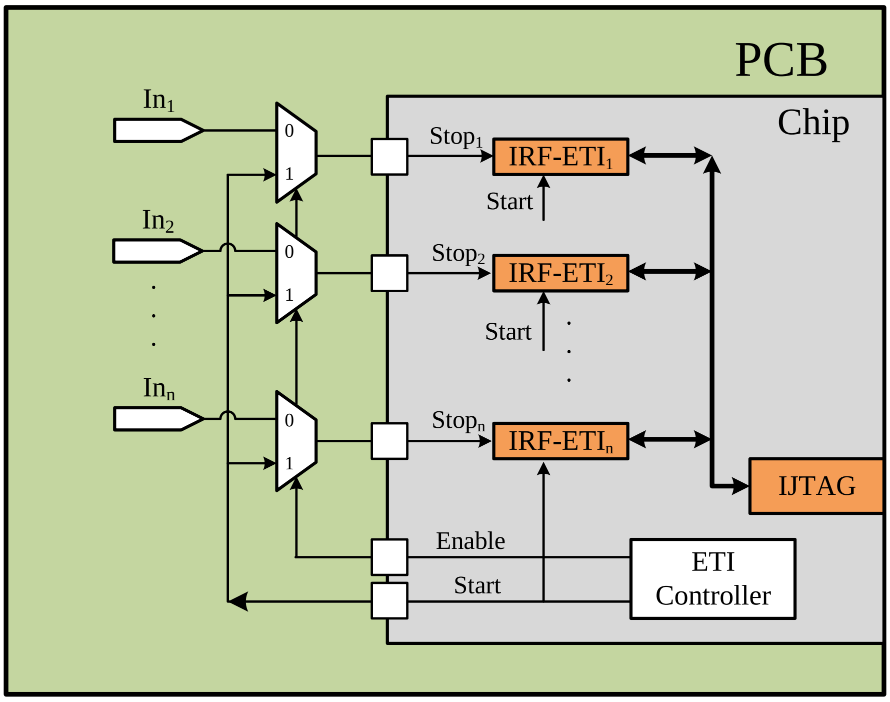


Figure 5.9: An implementation of the proposed method in a chip for parallel IRF detection in the number of n interconnections on the board.

For instance, standard gates commonly found on a chip, such as buffers, inverters, and carry-chains, have been utilised in previous studies [Ali19, Wan17, Che18a] to create the delay chain of the IRF-ETI with picoseconds sampling resolution. Other elements, such as routing elements [Sui18] and a phase-locked loop [Che16], have also been employed for this purpose.

In this chapter, the delay chain for the IRF-ETI is created using carry elements. This approach was chosen because it offers picoseconds resolution with minimal variation in propagation delay.

5.4.1 Experimental Setup and Measurements

Hardware implementations have been used to validate the efficiency and capability of the IRF detection method. Two case studies were conducted for this objective. First, single monitors were evaluated by injecting IRFs using a hardware IRF generator. This allows us to distinguish between intrinsic interconnection delay fluctuations and actual IRF occurrences. Second, one of the main causes of IRFs on boards being cold-solder joints [Kha12] were used under thermal cycling to create actual IRFs. In this case, the ability to concurrently monitor and detect IRFs has been evaluated under harsh environmental conditions. In this section, the reuse of the on-chip IRF-ETIs for board-level IRF detection has been investigated. First, the measurement setup for the IRF injection procedure is described briefly. Then, the measurement results extracted from measuring actual cold-solder joints under thermal cycling are presented.

In order to verify our design in a realistic environment, an IRF injection framework [Ebr19] was employed. The framework is composed of software and hardware parts. The software randomly generates IRFs and applies them to the hardware [Ebr16a] to be injected into selected interconnections for the IRF test. Figure 5.10 shows a simplified representation of the measurement setup and IRF injection architecture utilised in this section. It demonstrates how IRFs were injected into an interconnection between two chips.

The experimental setup is shown in Figure 5.11. In this figure, the devices are the following: an FPGA, an interconnection between an input port and an output port of the FPGA selected for IRF injection, and a microcontroller with our IRF generator mounted on it. The FPGA was used to implement the proposed testing method. The FPGA was mounted on an evaluation board [Tre20]. For simplicity, both chips 1 and 2, as shown in Figure 5.10, were implemented on the same FPGA in Figure 5.11.

The CARRY4 primitive [Xil14] of Xilinx FPGAs was chosen to implement the

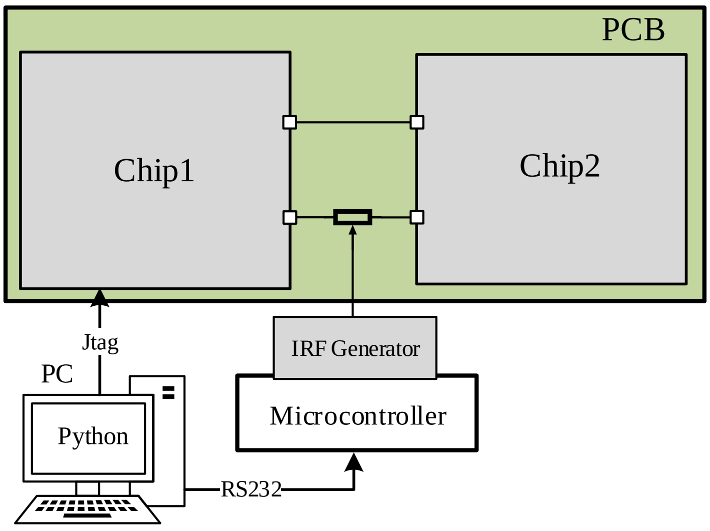


Figure 5.10: The IRF injection framework and the measurement setup.

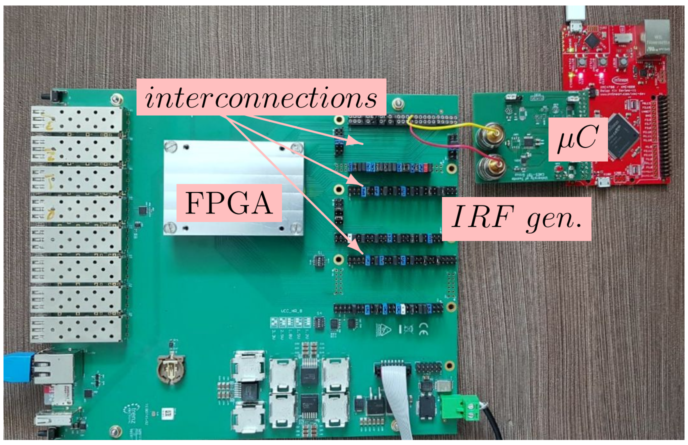


Figure 5.11: Experimental setup of the IRF injection and monitoring infrastructure.

delay chain in the IRF-ETI. The reason is that CARRY4 primitives have small propagation delays with low variations. After the placing and routing phase, the value of the propagation delays for adjacent CARRY4 primitives in the different IRF-ETIs was extracted using the Vivado tool [Xil14]. The propagation delay for the CARRY4 primitives had a low variation with averages of 59.1 ps and variances of 1.58 ps for 24 IRF-ETIs.

In the following, the effect of IRFs in interconnections between two chips at the board level, such as vias, traces, solder joints, and input/output ports, have been investigated. As a result, in our measurements, the circuits on the chips and the communication protocol between them are unimportant.

5.4.2 IRF injection and measurements

A hardware IRF injection technique was utilised to calculate the relationship between resistance changes in interconnections and the value (the number of ones) collected by the IRF-ETI. The experimental setup is shown in Figure 5.10 and Figure 5.11. The results of three different measurements on different input-output ports are depicted in Figure 5.12. The measurements show that monitors with 20 flip-flops can measure IRF changes ranging from 1 Ω to 80 Ω. The resolution of the IRF-ETI was measured to be about 3.86 Ω in terms of resistance. As can be seen in Figure 5.12, there is a linear relationship between the value of injected resistances and the number of 1's captured in the IRF-ETIs. However, there is a slight deviation from linearity if the succeeding CARRY4 primitive is located in another LUT. Because every LUT contains 4 CARRY4 primitives, 5 LUTs are required to implement the delay-chain of the IRF-ETI.

5.4.3 Thermal cycling experiment on solder joints

One of the main causes of IRFs in boards is poor (cold) solder joints [Kha12]. Thermal cycling is a practical method to evoke IRFs caused by solder joints in laboratories [Han19]. A board with several solder joints was created to test the capabilities of the proposed design in IRF detection. The measurement setup is depicted in Figure 5.13. This figure shows the monitoring of several solder joints on the board. Every IRF-ETI is connected to a pair of input and output ports (indicated by squares). Every input/output port of the FPGA is connected to a solder joint on the board (indicated by circles).

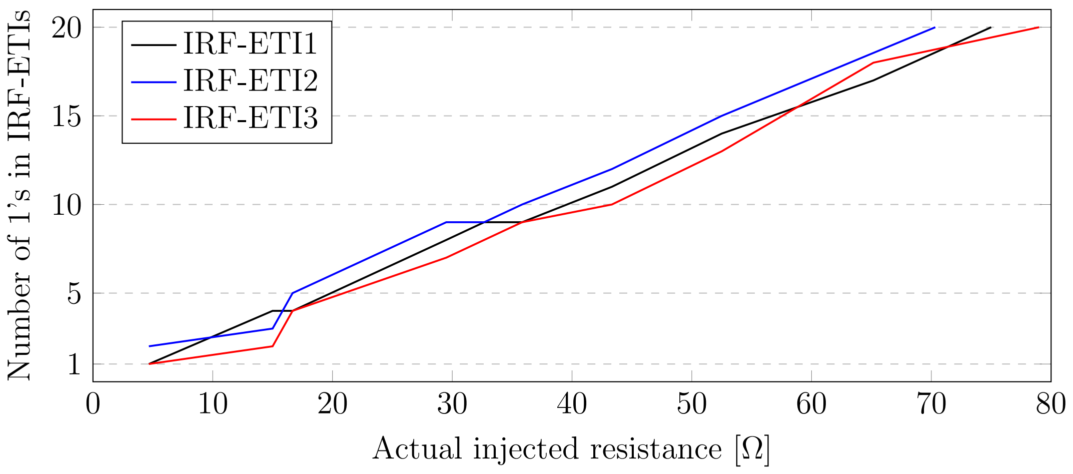


Figure 5.12: The relation between resistances injected by the IRF injector and the number of ones captured by IRF-ETIs.

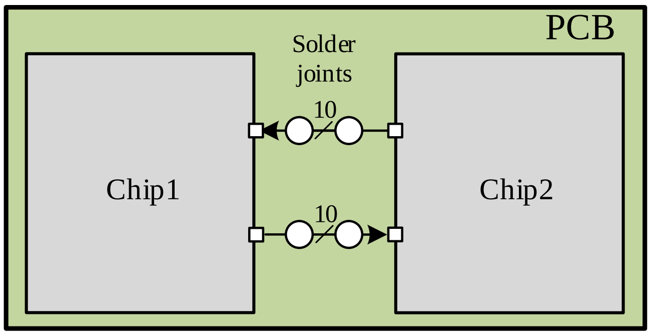


Figure 5.13: Experimental setup for IRF detection in interconnections and solder joints between two chips on a PCB board.

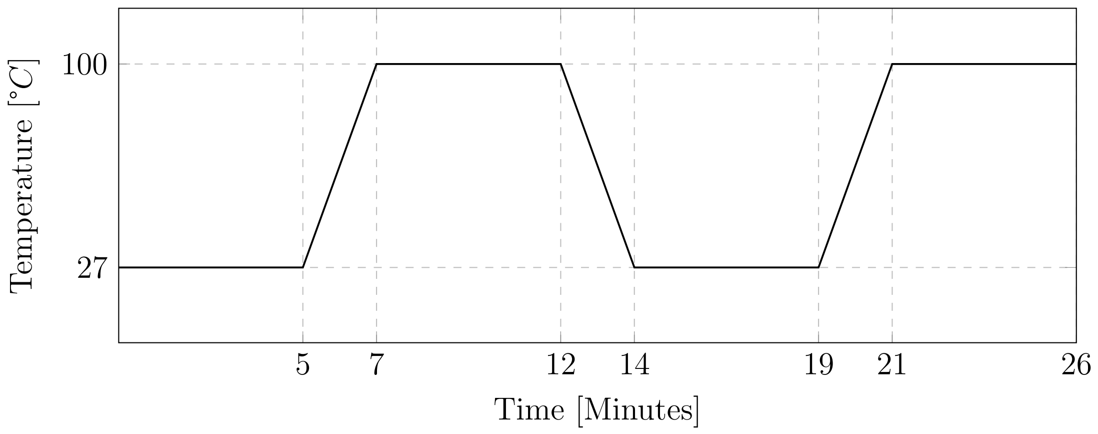


Figure 5.14: Thermal cycling profile with 5 minutes dwelling time.

In total, 20 pairs of input/output ports and 40 solder joints were used for the experiment. Of the 40 solder joints, 20 consisted of cold-solder joints. The remaining solder joints (20 joints) were correctly created with a sufficiently high soldering temperature in order to differentiate IRF effects from noises.

The behaviour of solder joints on the board was investigated under thermal cycling conditions. The process of creating cold-solder joins has been already explained in Section 3.1. Figure 5.14 shows the thermal cycling profile applied to the interconnections under test. The profile indicates that in every cycle, the board was heated up from 27 $°C$ to 100 $°C$ and subsequently cooled down to 27 $°C$ repeatedly. It should be noted that the highest temperature (100 $°C$) was selected to be lower than the melting point of the solder (180 $°C$). Otherwise, the cold solder may become a proper solder joint, which hinders the observation of the intermittent behaviour of the cold solder.

To calculate the test speed, the maximum value of the propagation delay for interconnections should be measured. As an example, the result of measurements of the propagation delays for three different sets of interconnections is shown in Figure 5.15. As expected, there is variation in the propagation delays of the interconnections in the fault-free case. Possible reasons are the heuristic algorithms used for the place and route as well as the process variations in the chip. In Figure 5.15, every set consists of 20 pairs of input and output ports and their corresponding soldering. As can be seen, the mean values of propagation delays are 5.0 ns, 6.6 ns and 6.7 ns for set1, set2 and set3, respectively. Besides the maximum value of propagation delay, the time that the IRF-ETI requires to capture data is essential for calculating the test speed. As previously stated, an IRF-ETI with a length of 20 flip-flops has a timing window of roughly 1.2 ns. By adding up these two values, the clock period of testing is obtained to be around 20 ns. Therefore, the clock speed was about 50 MHz for the experimental setup. These variations lead to extra offsets in the IRF measurement. These offsets can be removed with software or hardware techniques. In the software approach, the offset can be easily removed by subtraction. In the hardware technique, the offset can be obligated by creating an extra delay in the *Start* signal. We utilised the hardware technique to eliminate the average offset in our experiments.

In total, 120 thermal cycles were carried out for this experiment. From 10 pairs of cold solder joints, 5 of them were detected with IRFs. Out of 5 solder joints, one became an open circuit and a permanent fault after 106 thermal cycles. Figure 5.16 shows two examples of resistances calculated from the captured values in the IRF-ETI Type 1 while monitoring the cold solder joints under thermal cycling. Figure 5.16(a) shows an IRF in a cold-solder joint after 12 thermal cycles. As shown in Figure 5.16(b), the resistance value of the IRF becomes larger after 28 thermal

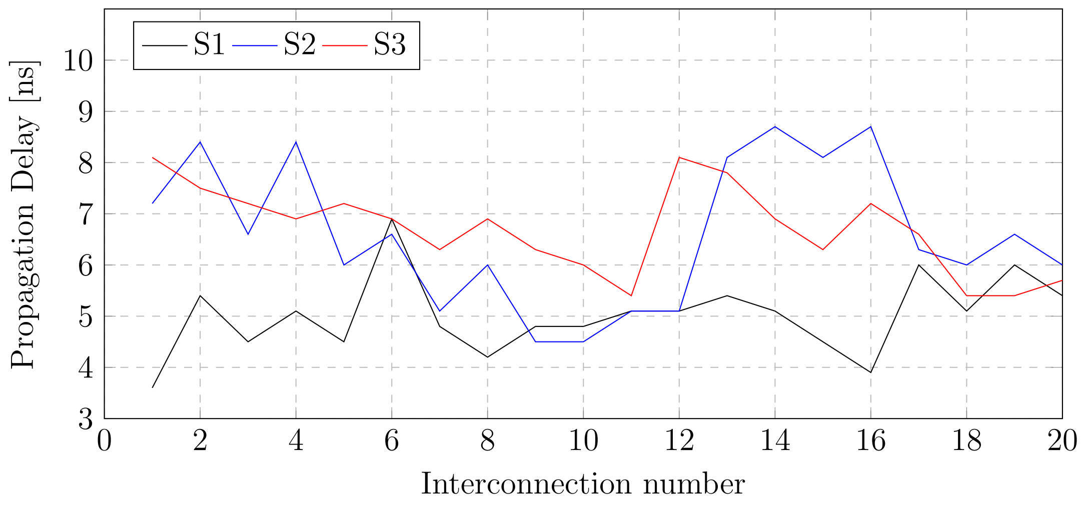


Figure 5.15: Measured propagation delays of the interconnections in the fault-free case for three sets of interconnections.

Table 5.4: Comparison of our design method based on IRF-ETIs and the design presented in [Kan20].

Design	Fault type	Minimum Resistance	Speed	Fabric
Proposed	Intermittent	3.86 Ω	50 MHz	FPGA
[Kan20]	Permanent	125 Ω	0.5 MHz	ASIC

cycles. Similar behaviour was observed for the other four solder joints with IRFs.

To the best of our knowledge, there is no similar work in the literature on this subject. The most related work has been recently published in [Kan20]. The comparison between the proposed method with the most related work published in [Kan20] is presented in Table 5.4. The design proposed in [Kan20] can detect permanent resistive faults with a minimum resistance of 125 Ω at the speed of 0.5 MHz. On the other hand, our device can detect permanent and intermittent resistive

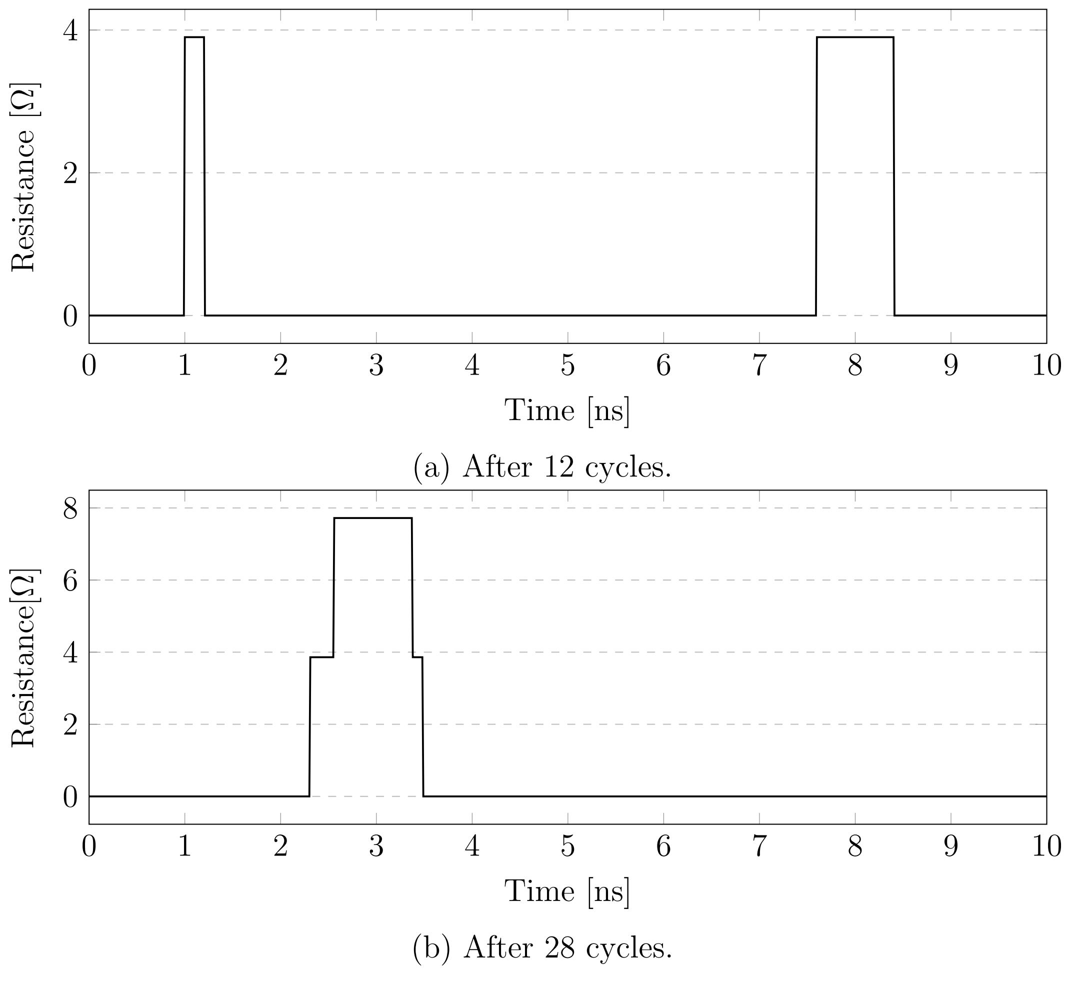


Figure 5.16: Two examples of IRFs induced by a cold-solder joint under thermal cycling detected by the IRF-ETIs. (a) After 12 thermal cycles (b) After 28 thermal cycles.

faults with a minimum of 3.86 Ω resistance. Moreover, the speed of testing in our design is 100 times faster.

5.5 Summary

The primary focus of this chapter was to address research question Q5. IRF detection at the board level was investigated, and several on-line testing methods were proposed.

First, the capability of the IEEE 1149.4 standard for IRF detection was investigated. Then, an enhanced version of the IEEE 1149.4 standard was proposed to tackle IRF detection at the board level. It was shown that the enhanced standard could detect IRFs in the interconnections with mixed-signals and digital signals.

Second, the effect of IRFs on data transmissions at the board level was studied. The on-chip ETIs were reused for IRF detection at the board level. Two popular on-board transmission protocols, the Universal Asynchronous Receiver Transmitter (UART) and the Serial Peripheral Interface bus (SPI), were chosen as case studies. It was shown that parity and frame checking cannot always detect IRFs and might lead to logic errors during serial data transmission. The injected IRFs resulted in undetected logic errors in 5 % and 34 % of the total transmitted data for the Hamming and the single-parity mechanisms, respectively. Hardware-based fault injections revealed that reusing our on-chip IRF-ETI allows the serial receivers to detect IRFs at early stages before they lead to logic errors.

At the end of this chapter, the reuse of our on-chip embedded test instruments for IRF detection at the board level was evaluated using hardware fault injections. Moreover, the ability of IRF detection using a combination of periodic testing and the ETIs was verified by evoking actual IRFs caused by cold solder joints under thermal cycling. The experimental results show that the proposed method can detect IRFs at the board level. It was shown that the IRF-ETI could detect IRFs in cold-solder joints with a minimum of 3.8 Ω resistance and a few nanoseconds duration. The periodic testing is relatively fast and can be carried out at 20 ns during system idle time.

Chapter 6
Evoking Intermittent Resistive Faults at Chip Level via On-chip Temperature Cycling

Abstract– *Evoking IRFs is very crucial during test mode. Without using evoking techniques, IRFs may remain inactive and, as a result, stay undetected. To speed up IRF detection, IRF evoking and monitoring should be employed simultaneously.*

In this chapter, IRF evoking at the chip level has been investigated. A thermal cycling technique has been proposed to evoke IRFs at the chip level. A software procedure using customised workloads has been introduced for creating and performing thermal cycling. Furthermore, the feasibility of creating local hotspots in a chip has been investigated using two processors as case studies: the Plasma and Xentium processors. The simulation results demonstrate that the thermal cycling technique produces local hotspots only with a few degrees of centigrade variation between functional units in the processors. Therefore, the thermal cycling technique can be used to evoke IRFs on the entire chip rather than locally. Our IRF-ETIs introduced in Chapter 4 can be employed during thermal cycling to detect and localise IRFs.

6.1 Introduction

No-fault-founds (NFFs) cause many product returns in the car and avionic industries, resulting in significant additional test and maintenance costs. A product is labelled as NFF if it has been reported to have faulty behaviours in the field. However, no fault can be found when the product is returned to the laboratory and

Parts of this chapter were published in:

- [Roh16] A. Rohani, H. Ebrahimi, and H. G. Kerkhoff "A software framework to calculate local temperatures in CMOS processors," in IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), pp. 183-188, 2016.

tested [Kha14, Ben02]. One reason is that the fault(s) can not be revoked during the test time. Knowing that intermittent resistive faults (IRFs) are one of the main causes of NFFs, evoking IRFs is critical for reducing IRF-induced test and maintenance costs.

IRFs should be evoked to be detected during the test time. The probability that an IRF becomes activated without extra stimuli while testing is very low. External stimuli, such as temperature changes, should therefore be used in a controlled environment to increase the likelihood of IRF detection.

Our experimental results in Chapter 3 demonstrate that thermal cycling effectively activates and evokes IRFs at the board level. The temperature changes activate IRFs in board interconnections such as loose connectors and cold solder joints.

Conventional techniques to raise the temperature in a chip, such as a processor, involve using thermal ovens [Mön06, Den05] or infrared techniques to heat up and then measure the temperature of the processor. However, these methods suffer from non-controllability. In other words, it is almost impossible to control the internal temperature of a chip while the entire chip is heated up externally.

In [Agh15], an algorithm is proposed to schedule a structural test to increase temperature locally and create a temperature gradient in a chip. Alternatively, this chapter investigates the possibility of employing functional workloads to control temperature changes at the chip level.

IRF evoking at the chip level can be used only if online IRF monitoring and detection are employed simultaneously with IRF evoking. IRF detection at the chip level has been discussed in Chapter 4. Two IRF-ETIs have been proposed for IRF detection at the chip level. The IRF-ETIs can be utilised to continuously or periodically monitor the chip in-field or while testing in the laboratory.

This chapter addresses the IRF evoking at the chip level by leveraging on-chip temperature cycling. First, employing workloads to control the temperature in a chip is examined. Following that, a method for generating thermal cycling using functional workloads in processors is proposed. As case studies, two processors, the Plasma [Pla15], and Xentium [Rec15, Wal11] processors, have been selected. The simulation results will show that customised workloads can be generated and employed to increase and decrease the temperature of processors cyclically. The generated thermal cycling can be utilised to speed up IRF evoking and, as a result, IRF detection at the chip level.

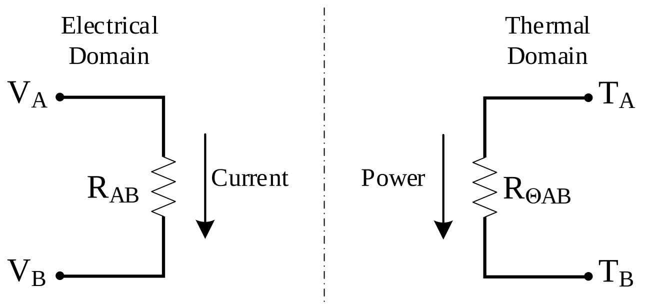


Figure 6.1: Fundamental relationships between the electrical and thermal domains.

6.2 Thermal Model

The first phase in creating a temperature profile is to have a thermal model capable of extracting detailed spatial granularity of heat distributions in a chip. Such a thermal model enables the designer to extract the temperature of each local block of a design. An essential factor is that the spatial granularity of the temperature profile should be sufficiently flexible to be determined by the designer. To accomplish this, a correlation between local temperature and switching activity (dynamic power dissipation) must first be established. For example, the temperature of each functional block in a processor can be determined based on the activity that the workload imposes on it.

The thermal model used in this chapter is based on the switching activity of each block, and the duality between heat transfer and electrical phenomena [Ska03], as shown in Figure 6.1. In this duality, the temperature difference is analogous to voltage, and the heat flow passing through a thermal resistance is analogous to electrical current. The heat dissipation in the thermal domain is proportional to the current delivered to that block in the electrical domain. In this model, an RC network of thermal resistances and capacitances is built for each functional block. The power dissipation is considered to be delivered at the centre of each block. This dynamic power dissipation is averaged over the entire workload execution and can be extracted using power-analysis tools, such as the Synopsys power analyser [Kur12]. The electrical circuit created using the thermal model can be solved by employing a circuit simulator such as Pspice or an analytical equation solver such as Hotspot [Ska03]. By solving the electrical circuit, the temperatures at the centres of functional modules are computed, and consequently, the thermal map for the entire processor.

The following section demonstrates the application of this model for two processors.

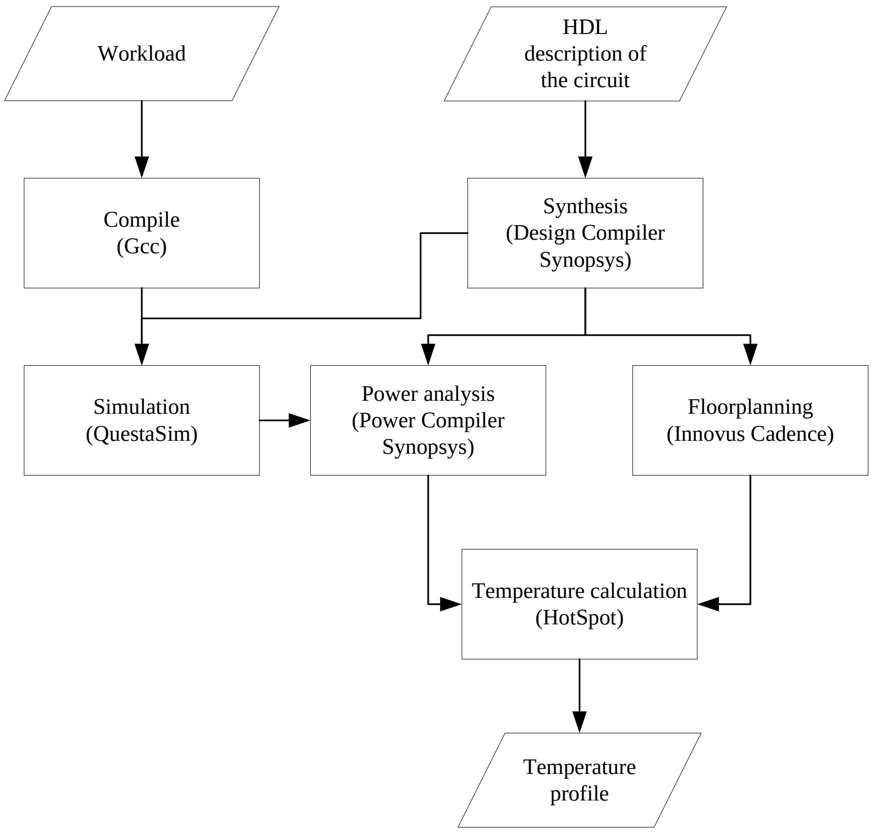

Workload
HDL description of the circuit
Compile (Gcc)
Synthesis (Design Compiler Synopsys)
Simulation (QuestaSim)
Power analysis (Power Compiler Synopsys)
Floorplanning (Innovus Cadence)
Temperature calculation (HotSpot)
Temperature profile

Figure 6.2: Flow to generate a thermal map of a given design under a specific workload.

6.3 Simulation Results

This section presents the simulation results for applying different workloads to two processors and extracting their temperature profiles.

The flow of creating a temperature map for each workload of a processor is shown in Figure 6.2. A workload written in the C language has been compiled using a compiler (GCC) in this flow. As a result, a binary (Hex) file has been generated for the post-synthesis simulation using the QuestaSim tool [Que16]. The result of the simulation is a switching-activity file. Our synthesis tool (Design Compiler Synopsys [Kur12]) receives an HDL description of the circuit and generates a gate-level netlist in the target technology (here, TSMC 40 nm CMOS technology). The power-analyser tool calculates the power dissipation based on the switching-activity file and the logic gate-level netlist. A floorplanning tool such as Innovus Cadence generates a floorplan of the circuit. After the floorplanning, the location of each unit of design is determined. At the end, the temperature calculation is carried out based on the thermal model explained previously. The HotSpot tool [Ska04] can be used for this phase. In [Ska04], this tool has been verified using measurements via on-chip temperature monitors in a chip.

The processors are briefly explained next, and then the simulation results and the temperature profile of the processors are presented. It will be shown that the relation between workloads and their induced temperature on the processors can be exploited for thermal cycling and, as a result, IRF evoking.

6.3.1 The Plasma Processor

A MIPS processor design called Plasma [Pla15] has been used for thermal map simulations. The architecture of this processor is shown in Figure 6.3. The processor consists of different units: multiplication and division (Mult), memory controller (Mem_ctrl), opcode decoder (Decoder), arithmetic logic unit (ALU), bus multiplexer (Bus_mux), register bank (Reg_bank), shifter (Shifter) and the program counter (PC_next). The *PC_next* unit determines the address of the next instruction. The opcode is fetched from the memory by the *Mem_ctrl* unit. The *Decoder* receives this 32-bit opcode, converts it into a 60-bit VLIW opcode, and then sends the control signals to the other units. The units *Shifter*, *ALU*, and *Mult* receive input data from the *a_bus* and *b_bus*, and write result data to the *c_bus*. For more details on the Plasma architecture reader is referred to [Pla15].

This design has been synthesised by us with the Synopsys Design Compiler tool [Kur12] using TSMC 40 nm CMOS technology. The performance-driven synthesis yielded a clock frequency of 500 MHz. The synthesis results are listed in Table 6.1 regarding the area and power dissipation.

Figure 6.4 shows a floorplan of the Plasma processor. In this figure, each processor unit is shown as a rectangle.

This processor has been subjected to three typical workloads; Quick-sort (QS), Reed–Solomon error correction (RS), and PI value calculator (PI), as well as two custom workloads; W_{High} and W_{Low}. The total power dissipation of each execution unit regarding each workload is shown in Table 6.1. As can be seen, each workload imposes a different switching activity and, consequently, power dissipation in each processor unit. However, the power dissipation values in some units, such as the register bank, are relatively close for different workloads. The reason is the architecture of the MIPS. For example, more than 90 % of instructions in each workload require reading from or writing to some registers.

To investigate the possibility of increasing the temperature of the processor

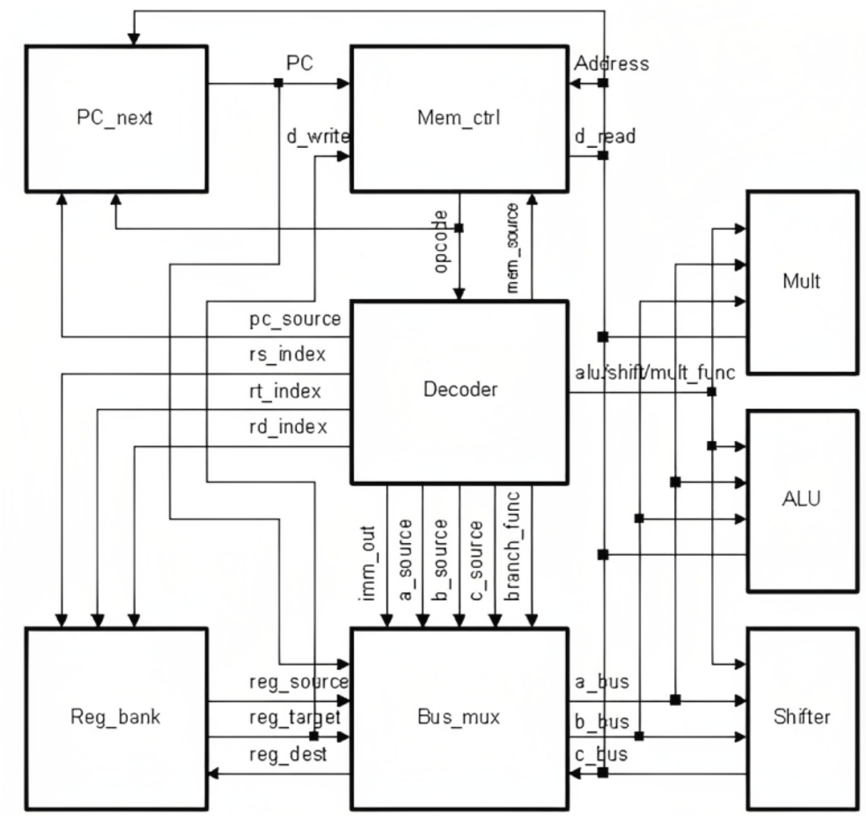


Figure 6.3: The architecture of the Plasma processor [Pla15].

Table 6.1: Dynamic power dissipation of each unit in the Plasma processor.

	Area $[\mu m^2]$	**Power dissipation**[mW]				
Units		**QS**	**RS**	**PI**	W_{High}	W_{Low}
Reg_bank	14192	**1.288**	**1.048**	**1.294**	**1.235**	**0.588**
Mult	3742	0.429	0.501	0.520	**0.771**	0.009
Mem_ctrl	2527	0.689	0.591	0.598	0.612	0.009
ALU	1968	0.456	0.238	0.245	**0.642**	0.007
Shifter	1939	0.064	0.044	0.037	**0.191**	0.003
Pc_next	1853	0.147	0.128	0.127	0.132	0.001
Bus_mux	1813	0.040	0.037	0.020	0.061	0.001
Decoder	878	0.017	0.015	0.015	0.017	0.001
Total power dissipation[mW]		3.13	2.602	2.856	3.661	0.619

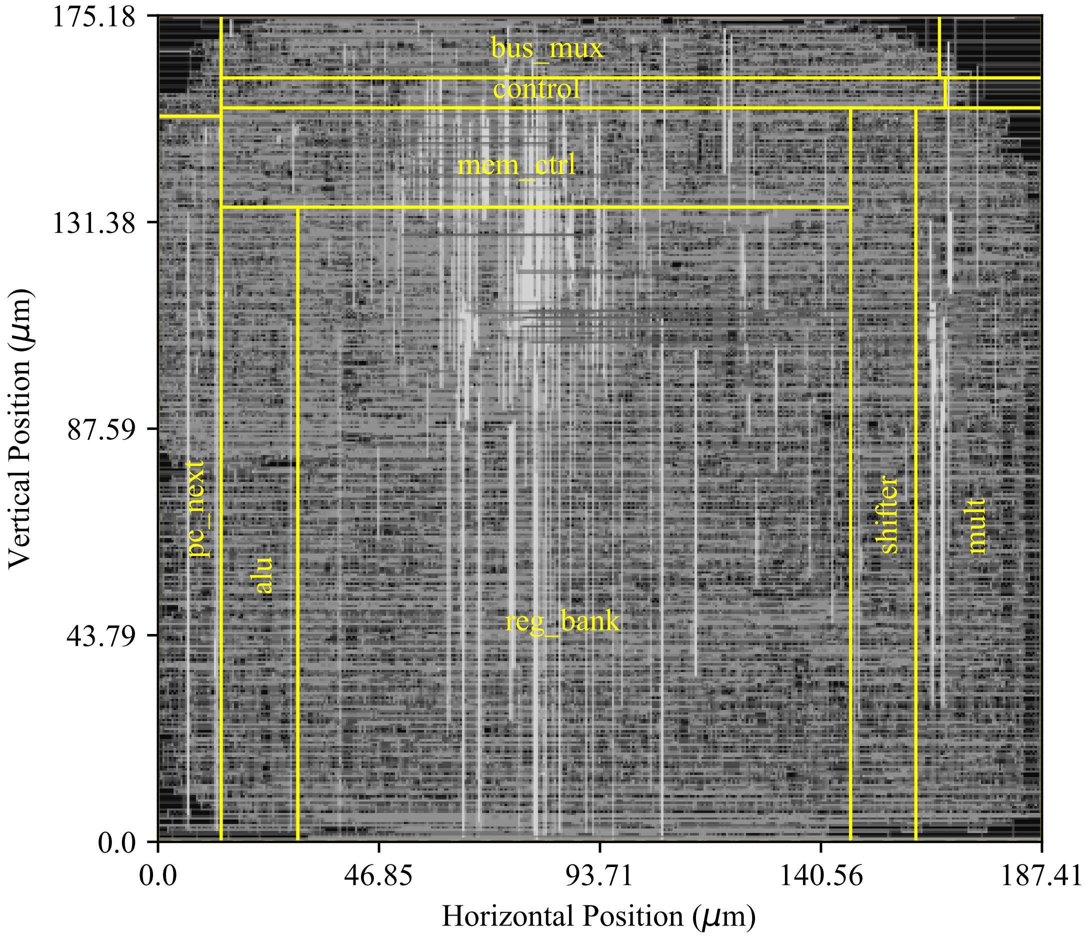


Figure 6.4: Our generated floorplan for the Plasma processor [Pla15].

using a workload, a customised workload (W_{High}) has been created to induce high temperatures in the processor. This workload increases switching activities in some execution units of the processor, such as the *Mult*, *ALU* and *Shifter* units. As can be seen in Table 6.1 (grey cells), the power dissipations in *Mult*, *ALU*, and *Shifter* are higher in comparison to the other workloads.

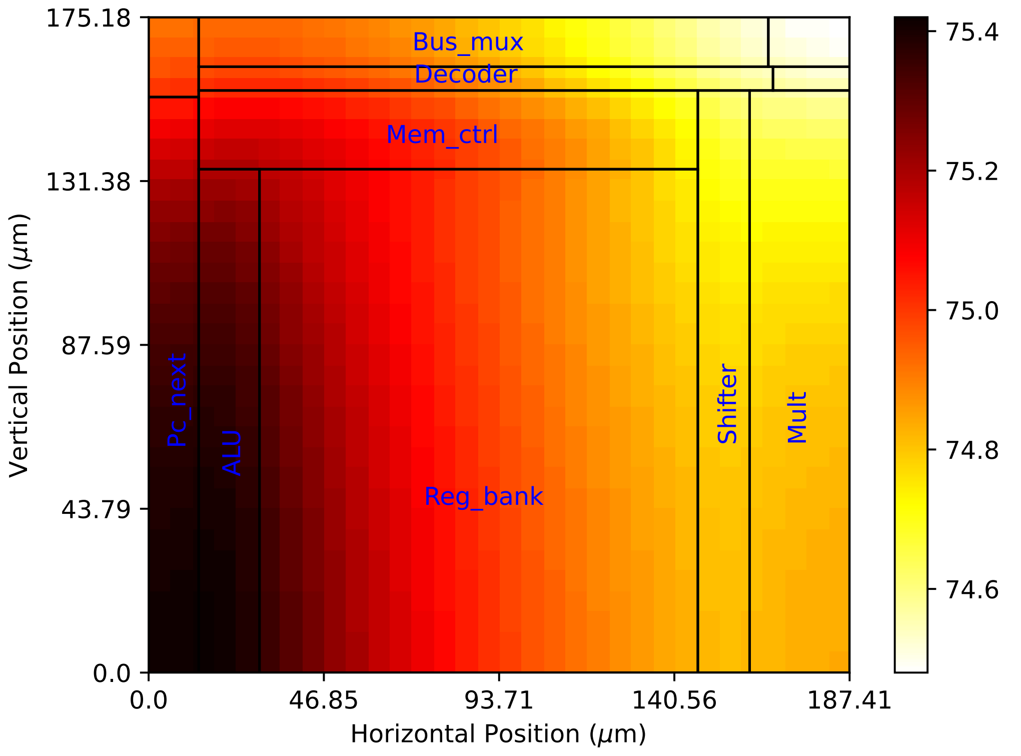
Similarly, a custom workload named W_{Low} has been designed to induce the lowest temperature on the processor by minimising the switching activities in functional blocks.

The workloads were applied to the Plasma processor, and based on the procedure depicted in Figure 6.2, the temperature values in each unit were extracted. For example, the thermal maps for the two workloads *QS* and W_{High} are depicted in Figure 6.5. The dark red areas indicate the highest temperature. The steady-state temperature for the *QS* and W_{High} workloads are 75.4 °C and 106.8 °C, respectively. As can be seen, the temperature difference between units is less than 1 °C. This is because the Plasma processor has a very small area ($175\mu m \times 187\mu m$), and the heat is conducted and distributed among adjacent units. it is not feasible to establish a local hotspot and temperature gradient for a small processor like Plasma. Subsequently, similar experiments have been conducted on a larger Xentium processor.

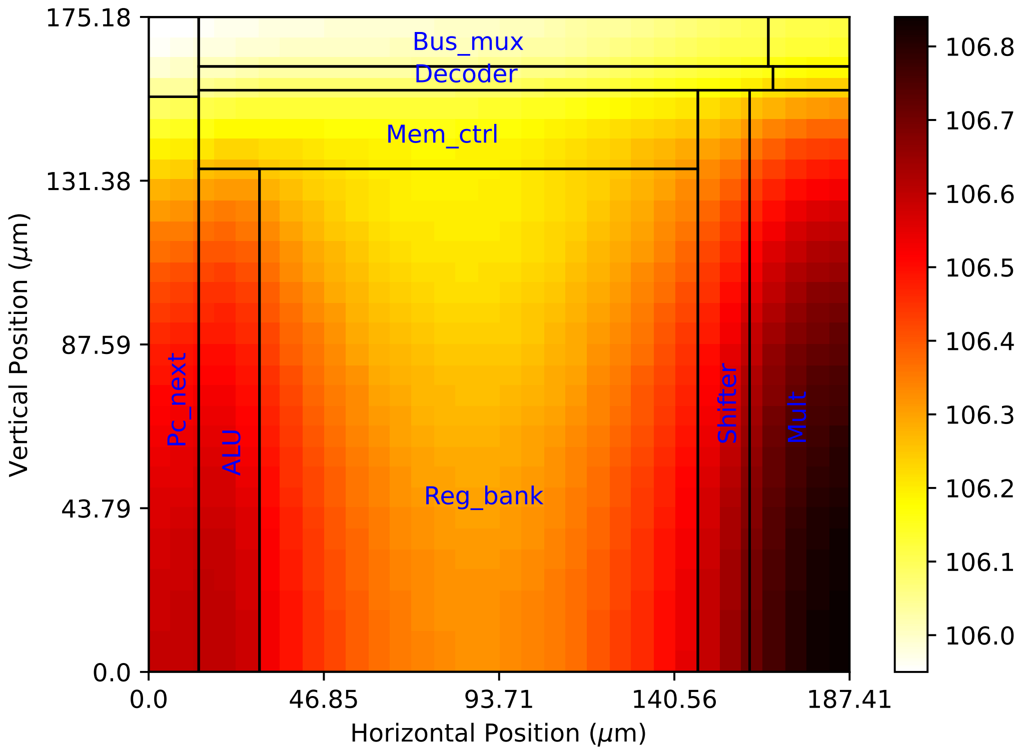
6.3.2 The Xentium Processor

In order to examine our approach on a larger and more complex design, we have chosen the single-core Xentium processor [Rec15, Wal11]. The Xentium processor [Rec15] is a high-performance processor designed for high-performance computing. Figure 6.6 shows the data-path architecture of this processor. The data path is based on a VLIW architecture that comprises ten execution units (*E0, E1, A0, A1, S0, S1, M0, M1, C0* and *P0*) as well as five register files (*RFA, RFB, RFC, RFD* and *RFE*). Each execution unit is responsible for a specific set of instructions. *E* units (*E0* and *E1*) perform load/store instructions while *A* and *S* units (*A0*, *A1*, *S0* and *S1*) execute arithmetic and logic operations. The *M* units (*M0* and *M1*) are multipliers. The *C* and *P* units (*C0* and *P0*) execute instructions that use the program counter. All five register files can be accessed simultaneously by all functional units. For more details on this processor, the reader is referred to [Rec15].

This large processor was an interesting candidate for investigating the feasibility of increasing temperature locally in each functional unit. Because each execution unit is dedicated to a specific instruction set, we may adjust the workload in order to impose a higher/lower temperature at the location of each execution unit. For example, rising temperatures in the *M* units imply a workload with many multiplication instructions.



(a) QS workload,



(b) Customised workload W_{High},

Figure 6.5: Temperature map of the Plasma processor induced by (a) QS workload and (b) a customised workload W_{High}.

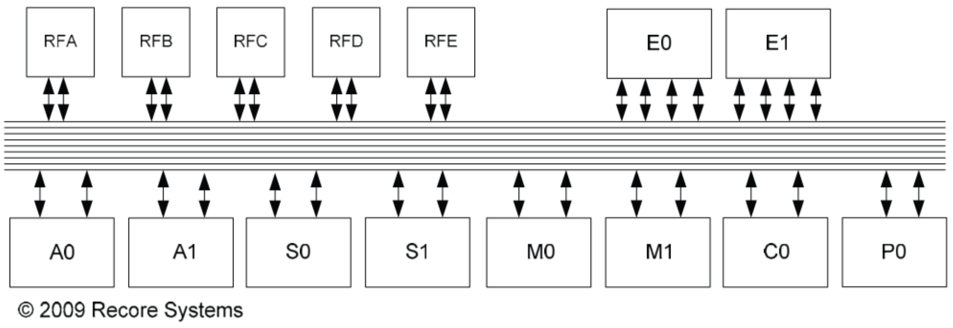


Figure 6.6: The Xentium data-path architecture [Rec15].

Table 6.2: Switching activity ratio in each execution unit of the Xentium processor.

	Switching activity ratio (%)					
Workloads	E0/E1	M0/M1	A0/A1	S0/S1	C	P
W1	80	10	10	10	50	40
W2	10	10	90	90	40	40
W3	20	20	20	20	90	80
W4	10	90	10	10	40	40

Four different workloads have been applied to the Xentium processor. Table 6.2 shows the switching activity ratio of each execution unit with regard to each workload. The switching activity ratio for each unit is the ratio of the switching activity induced by the workload to the maximum switching activity feasible for that unit. As can be seen, each workload imposes a different switching activity in each execution unit. For example, *W1* generates 80 % switching activity in units *E0* and *E1*, and 10 % in units *M0, M1, A0, A1, S0* and *S1*. The switching activity of units *C* and *control* are 50 % and 40 %, respectively.

The temperature value in each unit has been calculated using the procedure depicted in Figure 6.2. Table 6.3 shows the normalised temperature for different execution units. The lowest temperature is considered to be 0 °C. The hottest unit in each workload is highlighted in **bold**. As can be seen, different workloads impose different temperature distributions among the execution units of the processor. For example, in the case of the *W3* workload, a local hotspot with 11 °C difference between the coldest unit (E) and the hottest unit (P) was achieved.

Figure 6.7 depicts a more detailed heat expansion in different processing units. The dark red colour indicates the highest temperature.

The first observation from these results is that the temperature increment of each unit does not follow a linear dependency with the delivered dynamic powers. Other factors, however, such as the temperature of adjacent units and the positioning of

Table 6.3: Temperature difference distribution [°C] and the hotspot for each workload.

Unit	**W1**	**W2**	**W3**	**W4**
C	3	**5**	1	**5**
M	0	5	**7**	0
P	2	7	**11**	5
S	4	**10**	5	1
A	4	**10**	3	1
E	**9**	0	0	0

the unit inside the chip package, are critical. For instance, the *W1* workload imposes more dynamic power on the *C* and *P* units compared to the *A* and *S* units, yet the temperature elevation on the *A* and *S* units are higher. The reason is that the *A* and *S* units are located closer to the hottest units (*E* units). In the case of other workloads, there are also no linear correlations between the dynamic power and the local temperature of that unit.

Another observation is that ambient temperature conduction significantly impacts the temperature increment of each unit. For example, in the case of *W2* and *W3*, the *E1* unit has the lowest temperature, although its switching activity is in the same range as other units. A detailed investigation showed that *E1* unit loses much heat to the neighbouring area since it has two adjacent sides with the area with low temperature, while this is not the case for *E0* unit.

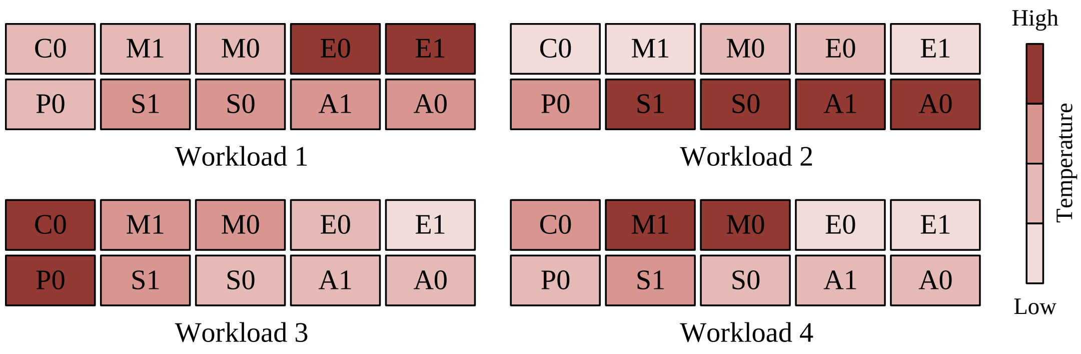


Figure 6.7: Heat distribution for various workloads (The darker colour a unit is, the higher the local temperature becomes for that workload).

6.4 IRF Evoking

The extracted temperature profile allows design and test engineers to have a more accurate view of the impact of a workload on heat generation in a processor.

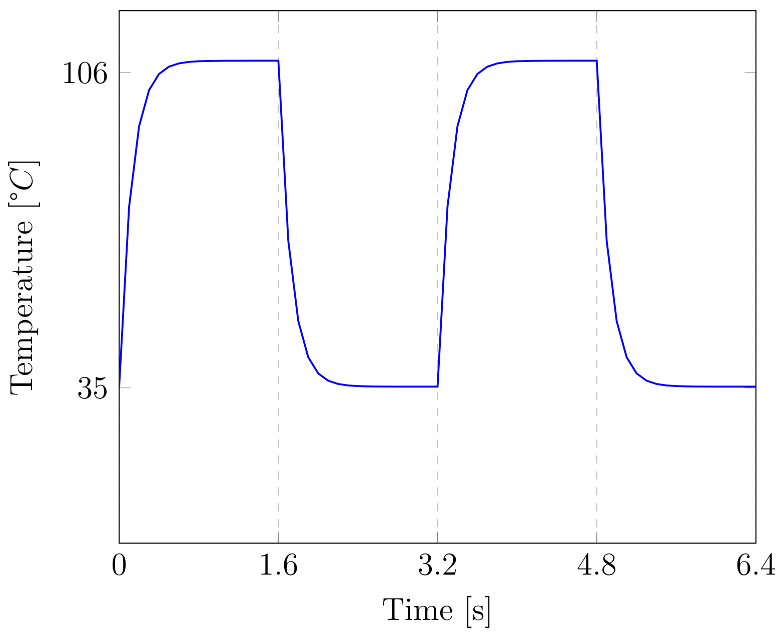


Figure 6.8: A simulated thermal cycling sequence for evoking IRF in the Plasma processor.

Moreover, understanding the influence of each workload on the temperature of a given processor can be utilised to develop a thermal cycling procedure for IRF evoking.

As demonstrated by the experimental results in Chapter 3, thermal cycling can be utilised to evoke IRFs at the board level. Knowing this fact motivates the employment of thermal cycling to facilitate IRF evoking at the chip level.

Thermal cycling inside a processor can be achieved by alternating between a workload that imposes a high temperature and a workload that induces a low temperature.

An example of thermal cycling applied to the Plasma processor is shown in Figure 6.8. Two workloads W_{high} and W_{low} were executed successively to achieve this thermal cycling. The W_{high} and W_{low} workloads induce a high and a low temperature in the Plasma processor, respectively. The impacts of the W_{high} and W_{low} workloads on the power dissipation are listed in Table 6.1. By applying the W_{high} workload, the temperature of the Plasma processor increases to 106 °C. The temperature drops to 35 °C when the W_{low} workload runs on the processor. Our previous test in Chapter 3 revealed that this temperature range is sufficient for IRF evoking. The simulation was performed using HotSpot [Ska04]. This tool has been verified using measurements via on-chip temperature monitors in [Ska04].

The simulation results show that custom workloads can be employed to generate and apply thermal cycling, effectively accelerating the process of IRF evoking at the chip level. The IRF detection techniques outlined in Chapter 4 can be used to detect the IRFs that become active during thermal cycling.

6.5 Summary

The investigation of IRF evoking at the chip level was the focus of this chapter, which was devoted to answering research question Q6. A method has been proposed for creating and applying thermal cycling in a chip using specific workloads.

First, the relation between dynamic power dissipation and executing workloads has been extracted. A thermal map related to the workload has been constructed using the chip layout information and the dynamic power dissipation induced by executing a specific workload.

The simulation results of the Plasma and Xentium processors revealed that executing various workloads results in different thermal maps. Moreover, the temperature distribution across the entire core of a small processor like the Plasma processor was nearly uniform, with a temperature difference of only 1 °C. However, in a larger-sized processor, such as the Xentium processor, it was possible to observe a local hotspot with a few degrees of the temperature gradient between functional units. The results have shown a maximum temperature gradient of 9 °C for the Xentium processor.

The simulation results have shown that the proposed method can be utilised to create thermal cycling at the chip level to accelerate IRF evoking. Furthermore, compared to thermal cycling at the board level, thermal cycling at the chip level can be carried out more precisely and cheaply using workloads without requiring an external oven.

Chapter 7
Conclusions and Future Work

Abstract– *In this chapter, The original problem statement and research questions are summarised. Then, the first general conclusions of this research are given. The original thesis contributions are stated and discussed. Finally, the limitations of this research and suggestions for future work are provided.*

7.1 Introduction

In the previous chapters, IRF modelling, simulation and hardware emulation were discussed. Moreover, IRF detection at the chip and board levels has been investigated. At the end, IRF evoking at the chip level has been addressed.

In this chapter, first, the problem statement and associated research questions from Chapter 1 are summarised in Section 7.2. Section 7.3 provides the final conclusions with regard to the research questions. Subsequently, all original contributions of our research are presented in Section 7.4. There are always items not included in this research due to time and resource limitations, resulting in recommendations for future work. The limitations and recommendations are provided in Section 7.5. This chapter concludes with a list of all papers published by the author.

7.2 Problem Statement and Associated Research Questions

As stated in Chapter 1, intermittent faults like IRFs, result in significant maintenance costs, especially in safety-critical boards and chips. From this problem, the following research questions have to be answered.

Q1. How do IRFs manifest in real life, and what is their physical origin?

Q2. At the board and chip levels, how can IRFs be modelled? How may IRFs be emulated (or simulated) to evaluate IRF detection techniques? How can the susceptibility to IRFs of a design be measured?

Q3. How may IRFs be detected at the chip level? Is it possible to build a fully digital, scalable, and IJTAG-compatible embedded instrument for IRF detection?

Q4. How may embedded instruments be located in a design to detect IRF effectively? How many embedded instruments are required, and where should they be placed?

Q5. On the board level, how can IRFs be detected? Is it possible to detect IRF in PCBs using on-chip embedded instruments?

Q6. Is it possible to speed up IRF detection while testing? How can IRFs be evolved and activated at the chip level?

7.3 Conclusions from This Research

From our experiment and simulation results, we can conclude as follows. IRFs can be detected at early stages before causing a logical error in the system. IRF detection can be carried out at the chip and board level.

Moreover, to investigate the behaviour of circuits under IRF effects, IRF generators have been developed. Simulation-based IRF generators can be used to simulate IRFs at the transistor level. To evaluate designs against IRFs at the board level, IRFs can be emulated using hardware-based IRF generators.

Embedded test instruments have been developed to detect IRFs at the chip level. To facilitate fault localisation, the ETIs can be designed to be compatible with the IEEE IJTAG standard. To increase the chance of IRF detection, the number of ETIs should be increased inside the chip. As a result, fully digital and scalable ETIs, with low hardware costs are required. Moreover, an insertion algorithm has been designed to find the best location inside the chip for inserting the ETIs.

At the board level, the IEEE 1149.4 standard can be enhanced for IRF detection for analogue as well as digital signals. To reduce hardware costs for IRF detection at the board level, on-chip ETIs can be reused for detecting IRFs in PCBs. In addition, due to the random occurrence of IRFs over time, periodic testing can increase the likelihood of IRF detection by retesting the system regularly.

Thermal cycling at the chip level can be used to accelerate IRF evoking during testing. Custom workloads can be used to perform thermal cycling at the chip to activate IRFs to speed up IRF detection.

To summarise, from our simulation and experimental results, IRF-ETIs are the recommended approach for detecting IRFs at early stages.

7.4 Our Contributions

This thesis contains several original contributions. These contributions respond to the research questions raised in Chapter 1 and Section 7.2. They are listed and briefly described below.

7.4.1 IRF Measurements and Modelling (Q1)

The thermal cycling approach was used in Chapter 3 to investigate the behaviour of intermittent resistive faults (IRFs). Many cold solder joints and loose connectors were produced for measurements.

The results of our experiments indicate that local temperature fluctuations on a PCB can activate IRFs. The measurements confirmed that IRFs occur randomly over time. Additionally, if an IRF occurred at a specific location, there is a possibility of its reappearance there at a later time Furthermore, in some cases, long-term exposure to temperature stress has exacerbated IRFs, leading to more severe faults and eventual failures Based on our measurements, a representative model of IRFs has been introduced.

7.4.2 IRF Simulation and Hardware Emulation (Q2)

In Chapter 3, we utilised our IRF model to design an IRF generator, enabling us to examine how circuits respond to the influence of IRFs at both the transistor and board levels. For IRF injection at the transistor level, a simulation-based IRF generator was used. To speed up IRF injection experiments, a hardware-based IRF generator was proposed [Ebr16a]. Once compared to the simulation-based IRF generator, the experimental findings revealed that the hardware-based IRF generator could speed up the IRF injection by approximately seven orders of magnitude. Furthermore, both simulation and experimental results revealed the vulnerability of digital systems to IRFs. Moreover, IRFs can create random timing errors in data paths and transmission lines. In addition, IRFs can cause single and multiple bit-flips, as well as frame errors in UART data transmission [Ebr18].

Moreover, the IRF generator can be used to evaluate the sensitivity of new designs regarding IRFs. We utilised both software and hardware-based IRF generators to validate our new IRF embedded test instruments at both the chip and board levels

7.4.3 IRF Detection at Chip Level (Q3)

In Chapter 4, two embedded test instruments for IRF monitoring and detection at the chip level have been proposed by us [Ebr16b, Ebr21]. Scalable, completely digital, IJTAG compatible, and low-cost hardware requirements are all features of our ETIs. Simulation-based fault injections were used to test the capability of our embedded test instruments for IRF detection. The simulation results have shown that the embedded test instruments successfully detect IRFs. It has been shown that ETIs can be utilised to differentiate between intermittent, ageing, and permanent faults [Ebr16b].

In addition, a new periodic testing method based on our embedded test instrument has been proposed for the on-line detection of IRFs at the chip level. As a case study, an implementation of our method based on our IRF embedded test instruments in a NoC has been studied using post-synthesis simulations. According to the simulation results, IRFs can affect data and control signals in a NoC. About half of the injected IRFs resulted in logic errors. Furthermore, simulations have been used to verify the ability of our testing method to identify IRFs at the chip level. The findings revealed that if the testing execution cycles are 10 % of the total cycles, the IRF coverage can be as high as 90 % [Ebr21].

7.4.4 Insertion Algorithm for IRF Monitoring (Q4)

A node-selection algorithm was proposed for IRF monitoring at the chip level, in Chapter 4. Based on the propagation probability and an estimate of the number of vias for each net in the design, the proposed algorithm aims to locate the optimum sites that give maximum IRF coverage. Our approach can offer 100 % IRF coverage for input ports at the cost of around 3.5 % area overhead, according to simulation findings on the ISCAS 89 benchmarks.

Furthermore, the findings revealed that having full IRF coverage for the entire design interconnection is very costly in terms of area overhead. However, our algorithm can provide 50 % IRF coverage for the entire design interconnections with an average of 9.2 % area overhead [Ebr20].

7.4.5 IRF Detection at Board Level (Q5)

In Chapter 5, IRF detection at the board level was explored, and several on-line testing approaches to tackle IRFs were presented.

First, the capability of the IEEE 1149.4 standard for IRF detection was investigated. Then, we proposed an enhanced version of the IEEE 1149.4 standard to tackle IRF detection at the board level. It was demonstrated that the enhanced version of the IEEE 1149.4 standard can detect IRFs in the interconnections used for analogue as well as digital signals [Ker15].

Second, the impact of IRFs on data transmission on the board has been investigated. The on-chip ETIs have been reused for IRF detection at the board level. As case studies, two widely used on-board transmission protocols, the Universal Asynchronous Receiver Transmitter (UART) and the Serial Peripheral Interface bus (SPI), have been chosen for our IRF injection experiments. Even with parity and frame checking, IRFs have been found to cause logic errors during serial data transmission. For Hamming and the single-parity technique, injected IRFs resulted in undetected logic errors in 5 % and 34 % of the total transmitted data, respectively. According to hardware-based fault injections, reusing our on-chip IRF-ETIs allows the serial receivers to detect IRFs at early stages before they lead to undetected logic errors [Ebr19].

Finally, we investigated the reuse of our on-chip embedded test instruments for IRF detection at the board level by employing hardware fault injections [Ebr21]. Moreover, the ability of IRF detection using a combination of periodic testing and the ETIs was validated by evoking actual IRFs induced by cold solder joints under thermal cycling. The experimental results show that our on-chip ETIs can be reused periodically to detect IRFs at the board level. It was shown that the IRF-ETIs could detect IRFs in cold-solder joints with a minimum resistance of 3.86 Ω and a few nanoseconds activation time. The periodic testing is relatively fast and requires 20 ns to be carried out.

7.4.6 Evoking IRFs at Chip Level (Q6)

IRF invoking at the chip level was examined in Chapter 6. A method has been proposed for creating and applying thermal cycling in a processor using different workloads.

First, the relationship between dynamic power dissipation and workload execution has been determined. A thermal map relevant to the workload was created using the chip layout information and the dynamic power dissipation caused by executing a workload. The simulation results on two CPUs, the Plasma and Xentium processors, demonstrated that different workloads produce different thermal maps. Moreover, the temperature distribution across the entire core of a small processor like the Plasma processor is almost uniform. However, a local hotspot with a few degrees of centigrade temperature variation between functional units can be obtained in a larger processor, such as the Xentium processor.

According to the simulation results, our method can produce thermal cycling at the chip level to accelerate IRF evoking. Furthermore, compared to thermal cycling at the board level, thermal cycling at the chip level may be conducted more accurately and at a lower cost by employing workloads instead of an external oven [Roh16].

7.5 Future Work

Due to time and resource limitations, there are currently some shortcomings that can be improved in future work.

First, the IRF models provided in this thesis are based on a limited number of measurements. However, more measurements on different interconnections are required to drive a more precise model. In addition, the proposed model consists of variable resistance and fixed capacitors. Knowing that IRFs occur due to physical defects, having a model with variable resistance and capacitors would be more accurate for representing an IRF.

Second, the IRF-ETIs presented in Chapter 4 are simulated against the ageing effect. The IRF-ETIs can differentiate between IRF faults and ageing faults. However, the simulations were based on the 65 nm NBTI model, which is relatively old technology.

Third, the IRF-ETIs presented in this thesis were verified by real hardware for IRF detection at the board level. However, IRF detection at the chip level has not been verified by real hardware.

Fourth, the thermal cycling procedure was proposed in Chapter 6 for evoking IRFs in processors. The procedure was evaluated using software simulation. However, the procedure should be verified on a real processor equipped with temperature sensors.

Based on the limitations mentioned above, for future research, it is recommended to focus on the following areas:

- The first area of interest is IRF modelling. Clearly, having an accurate IRF model is the first step in order to develop better IRF detection methods. More measurements are required to construct an accurate IRF model that can represent variations in capacitance in addition to a variable resistance. In addition, more measurements are necessary to model the ageing effect of IRFs.
- The simulation for verifying IRF-ETI capability in ageing fault detection should be carried out based on more recent technology than 65 nm.
- One can employ the proposed algorithm in Chapter 4 to place our IRF-ETIs in an actual design to evaluate the efficiency of the IRF-ETIs for IRF detection at the chip level using real hardware.
- The efficiency of the procedure presented in Chapter 6 can be verified on a real processor equipped with local temperature sensors.

No Fault Found and IRFs are among the most challenging dependability issues nowadays. NFFs threaten the dependability of highly-dependable systems in the avionic and car industries. NFFs grow as the number and density of interconnections on boards and chips grow in future technologies. Therefore, more research is required to tackle NFFs in the future.

List of Our Publications

[Ebr16a] Hassan Ebrahimi and Hans G. Kerkhoff. “Testing for intermittent resistive faults in cmos integrated systems”. In *IEEE Euromicro Conference on Digital System Design (DSD)*, pages 703–707, 2016.

[Ebr16b] Hassan Ebrahimi, Alireza Rohani, and Hans G. Kerkhoff. “Detecting intermittent resistive faults in digital cmos circuits”. In *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 87–90, 2016.

[Ebr18] Hassan Ebrahimi and Hans G. Kerkhoff. “Intermittent resistance fault detection at board level”. In *IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 135–140, 2018.

[Ebr19] Hassan Ebrahimi and Hans G. Kerkhoff. “A digital on-line monitor for detecting intermittent resistance faults at board level”. In *World Scientific Journal of circuits, systems and computers (JCSC)*, volume 28, no. 01, (2019), pages 1940003–1–1940003–15.

[Ebr20] Hassan Ebrahimi and Hans G. Kerkhoff. “A new monitor insertion algorithm for intermittent fault detection”. In *IEEE European Test Symposium (ETS)*, pages 1–6, 2020.

[Ebr21] Hassan Ebrahimi and Hans G. Kerkhoff. “Embedded Test Instrument for Intermittent Resistive Fault Detection at Chip Level and Its Reuse at Board Level”. In *IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 75–80, 2021.

[Ker15a] Hans G. Kerkhoff and Hassan Ebrahimi. “Detection of intermittent resistive faults in electronic systems based on the mixed-signal boundary-scan standard”. In *IEEE Asia Symposium on Quality Electronic Design (ASQED)*, pages 77–82, 2015.

[Ker15b] Hans G. Kerkhoff and Hassan Ebrahimi. “Intermittent resistive faults in digital cmos circuits”. In *IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 211–216, 2015.

[Ker16] Hans G. Kerkhoff and Hassan Ebrahimi. “Investigation of intermittent resistive faults in digital cmos circuits”. In *World Scientific Journal of circuits, systems and computers (JCSC)*, volume 25, no. 03, (2016), pages 1640023–1–1640023–17.

[Ker17] Hans G. Kerkhoff, Ghazanfar Ali, and Hassan Ebrahimi. “An automotive mp-soc featuring an advanced embedded instrument infrastructure for high dependability”. In *IEEE International Test Conference in Asia (ITC-Asia)*, pages 65–70, 2017.

[Roh16] Alireza Rohani, Hassan Ebrahimi, and Hans G. Kerkhoff. “A software framework to calculate local temperatures in cmos processors”. In *IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 183–188, 2016.

List of References

[AES19] AES-128, "Advanced encryption standard AES-128," `http://www.opencores.org/`, 2019.

[Agh15] N. Aghaee, Z. Peng, and P. Eles, "Temperature-gradient-based burn-in and test scheduling for 3-d stacked ics," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, volume 23, no. 12, pp. 2992–3005, 2015.

[Ali19] G. Ali, J. Pathrose, and H. G. Kerkhoff, "IJTAG compatible timing monitor with robust self-calibration for environmental and aging variation," in IEEE European Test Symposium (ETS), pp. 1–6, IEEE, 2019.

[And12] K. Anderson and U. Synaptics, "Intermittent fault detection & isolation system (IFDIS)," in CTMA Symposium White Paper Synaptics, 2012.

[And14] A. F. Anderson, "Intermittent electrical contact resistance as a contributory factor in the loss of automobile speed control functional integrity," in IEEE Access, volume 2, pp. 258–289, 2014.

[Ben02] I. Beniaminy and D. Joseph, "Reducing the "no fault found" problem: Contributions from expert-system methods," in IEEE Aerospace Conference, volume 6, pp. 2971–2973, 2002.

[Ble11] H. Bleeker, P. van Den Eijnden, and F. de Jong, Boundary-scan test: a practical approach, Springer Science & Business Media, 2011, ISBN 978-1-4613-6371-2.

[Bow09] K. A. Bowman, J. W. Tschanz, N. S. Kim, et al., "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," in IEEE Journal of Solid-State Circuits, volume 44, no. 1, pp. 49–63, 2009.

[Brg89] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in IEEE International Symposium on Circuits and Systems, pp. 1929–1934, 1989.

[Bus04] M. Bushnell and V. Agrawal, Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits, volume 17, Springer Science & Business Media, 2004, ISBN 0-306-47040-3.

[Cad14] Cadence Design Systems Inc., "Virtuoso Spectre simulator user guide," `https://www.cadence.com/`, 2014.

[Che13] L. Chen, M. Ebrahimi, and M. B. Tahoori, "Cep: Correlated error propagation for hierarchical soft error analysis," in Journal of Electronic Testing, volume 29, no. 2, pp. 143–158, 2013.

[Che15] H. B. Chen, S. X. D. Tan, V. Sukharev, et al., "Interconnect reliability modeling and analysis for multi-branch interconnect trees," in IEEE Annual Design Automation Conference, pp. 1–6, 2015.

[Che16] P. Chen, Y. Y. Hsiao, Y. S. Chung, et al., "A 2.5-ps bin size and 6.7-ps resolution FPGA time-to-digital converter based on delay wrapping and averaging," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, volume 25, no. 1, pp. 114–124, 2016.

[Che18a] H. Chen and D. D. Li, "Multichannel, low nonlinearity time-to-digital converters based on 20 and 28 nm FPGAs," in IEEE Transactions on Industrial Electronics, volume 66, no. 4, pp. 3265–3274, 2018.

[Che18b] W. K. Chen, The VLSI handbook, CRC press, 2018, ISBN 978-0-849-34199-1.

[Con03] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," in IEEE Micro, volume 23, no. 4, pp. 14–19, 2003.

[Con07a] C. Constantinescu, "Impact of intermittent faults on nanocomputing devices," in IEEE Workshop on Dependable and Secure Nanocomputing, 2007.

[Con07b] C. Constantinescu, "Intermittent faults in VLSI circuits," in IEEE Workshop on Silicon Errors in Logic-System Effects, 2007.

[Con08] C. Constantinescu, "Intermittent faults and effects on reliability of integrated circuits," in IEEE Reliability and Maintainability Symposium (RAMS), pp. 370–374, 2008.

[Das09] S. Das, C. Tokunaga, S. Pant, et al., "RazorII: In situ error detection and correction for PVT and SER tolerance," in IEEE Journal of Solid-State Circuits, volume 44, no. 1, pp. 32–48, 2009.

[Dav05a] S. Davidson, “Towards an understanding of no trouble found devices,” in IEEE VLSI Test Symposium (VTS), pp. 147–152, 2005.

[Dav05b] S. Davidson, “Understanding NTF components from the field,” in IEEE International Conference on Test, pp. 1–10, 2005.

[Den05] X. Deng, H. Feng, G. Li, et al., “A ptas for semiconductor burn-in scheduling,” in Springer Journal of Combinatorial Optimization, volume 9, no. 1, pp. 5–17, 2005.

[Ebr16a] H. Ebrahimi and H. G. Kerkhoff, “Testing for intermittent resistive faults in CMOS integrated systems,” in IEEE Euromicro Conference on Digital System Design (DSD), pp. 703–707, 2016.

[Ebr16b] H. Ebrahimi, A. Rohani, and H. G. Kerkhoff, “Detecting intermittent resistive faults in digital CMOS circuits,” in IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 87–90, 2016.

[Ebr16c] M. Ebrahimi, Z. Ghaderi, E. Bozorgzadeh, et al., “Path selection and sensor insertion flow for age monitoring in FPGAs,” in IEEE Design, Automation and Test in Europe (DATE), pp. 792–797, 2016.

[Ebr18] H. Ebrahimi and H. G. Kerkhoff, “Intermittent resistance fault detection at board level,” in IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), pp. 135–140, 2018.

[Ebr19] H. Ebrahimi and H. G. Kerkhoff, “A digital on-line monitor for detecting intermittent resistance faults at board level,” in World Scientific Journal of Circuits, Systems and Computers (JCSC), volume 28, no. 01, pp. 1940003–1–1940003–15, 2019.

[Ebr20] H. Ebrahimi and H. G. Kerkhoff, “A new monitor insertion algorithm for intermittent fault detection,” in IEEE European Test Symposium (ETS), pp. 1–6, 2020.

[Ebr21] H. Ebrahimi and H. G. Kerkhoff, “Embedded test instrument for intermittent resistive fault detection at chip level and its reuse at board level,” in IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), pp. 75–80, 2021.

[Eno18] R. Enomoto, T. Iizuka, T. Koga, et al., “A 16-bit 2.0-ps resolution two-step TDC in 180nm CMOS utilizing pulse-shrinking fine stage with built-in coarse gain calibration,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, volume 27, no. 1, pp. 11–19, 2018.

[Fan11] Y. Fang and X. Chen, “Design and simulation of UART serial communication module based on VHDL,” in IEEE International Workshop on Intelligent Systems and Applications, pp. 1–4, 2011.

[Fly12] C. Flynt, Tcl/Tk a developers guide, Elsevier, 2012, ISBN 978-0-12-384717-1.

[Gar13] J. GarciaMoran, D. Gil Tomas, L. J. Saiz Adalid, et al., “Defining a representative and low cost fault model set for intermittent faults in microprocessor buses,” in IEEE Latin-American Symposium on Dependable Computing, pp. 98–103, 2013.

[Gar14] J. GarciaMoran, J. C. Baraza Calvo, D. Gil Tomas, et al., “Effects of intermittent faults on the reliability of a reduced instruction set computing (RISC) microprocessor,” in IEEE Transactions on Reliability, volume 63, no. 1, pp. 144–153, 2014.

[Gha18] B. Ghavami, M. Raji, K. Saremi, et al., “An incremental algorithm for soft error rate estimation of combinational circuits,” in IEEE Transactions on Device and Materials Reliability, volume 18, no. 3, pp. 463–473, 2018.

[Gil12] D. Gil Tomás, J. Gracia Morán, J. C. Baraza Calvo, et al., “Studying the effects of intermittent faults on a microcontroller,” in Elsevier Microelectronics Reliability, volume 52, no. 11, pp. 2837–2846, 2012.

[Gil15] D. Gil Tomás, J. Gracia Morán, J. C. Baraza Calvo, et al., “Injecting intermittent faults for the dependability assessment of a fault-tolerant microcomputer system,” in IEEE Transactions on Reliability, volume 65, no. 2, pp. 648–661, 2015.

[Gom18] A. F. Gomez and V. Champac, “Selection of critical paths for reliable frequency scaling under BTI-aging considering workload uncertainty and process variations effects,” in ACM Transactions on Design Automation of Electronic Systems (TODAES), volume 23, no. 3, pp. 1–21, 2018.

[Han12] J. Hannu, J. Häkkinen, J. V. Voutilainen, et al., “Current state of the mixed-signal test bus 1149.4,” in Springer Journal of Electronic Testing, volume 28, no. 6, pp. 857–863, 2012.

[Han19] C. Han, S. Park, and H. Lee, “Intermittent failure in electrical interconnection of avionics system,” in Elsevier Reliability Engineering & System Safety, volume 185, pp. 61–71, 2019.

[Hof08] J. P. Hofmeister, P. Lall, D. Panchagade, et al., “Ball grid array (bga) solder joint intermittency detection: SJ BIST,” in IEEE Aerospace Conference, pp. 1–11, 2008.

[Hof10] J. P. Hofmeister, S. Vohnout, C. Mitchell, et al., “Halt evaluation of SJ BIST technology for electronic prognostics,” in IEEE AUTOTESTCON, pp. 1–7, 2010.

[Hsi15] M. Hsieh, Y. Huang, T. Yew, et al., “The impact and implication of BTI/HCI decoupling on ring oscillator,” in IEEE International Reliability Physics Symposium (IRPS), pp. 6A–4, 2015.

[IEE14] IEEE1687, “1687-2014—IEEE standard for access and control of instrumentation embedded within a semiconductor device,” in IEEE Standard, pp. 1–283, 2014.

[Ish01] M. Ishida, D. S. Ha, T. Yamaguchi, et al., “I_{DDT} testing: an efficient method for detecting delay faults and open defects,” in IEEE International Workshop on Defect Based Testing, pp. 23–28, 2001.

[Jah19] H. Jahanirad, “CC-SPRA: Correlation coefficients approach for signal probability-based reliability analysis,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, volume 27, no. 4, pp. 927–939, 2019.

[Jam03] I. James, D. Lumbard, I. Willis, et al., “Investigating no fault found in the aerospace industry,” in IEEE Annual Reliability and Maintainability Symposium, pp. 441–446, 2003.

[Jef05] C. Jeffrey, R. Cutajar, S. Prosser, et al., “The integration of on-line monitoring and reconfiguration functions using IEEE1149. 4 into a safety critical automotive electronic control unit,” in IEEE Design, Automation and Test in Europe (DATE), pp. 153–158, 2005.

[Jia19] Y. Jiao, K. Jermsittiparsert, A. Y. Krasnopevtsev, et al., “Interaction of thermal cycling and electric current on reliability of solder joints in different solder balls,” in Materials Research Express, volume 6, no. 10, pp. 1063021–10630210, 2019.

[Jun12] M. Jung, J. Mitra, D. Z. Pan, et al., “TSV stress-aware full-chip mechanical reliability analysis and optimization for 3D IC,” in IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, volume 31, no. 8, pp. 1194–1207, 2012.

[Kan20] M. Kanda, M. Hashizume, F. A. B. Ali, et al., “Open defect detection not utilizing boundary scan flip-flops in assembled circuit boards,” in IEEE Transactions on Components, Packaging and Manufacturing Technology, volume 10, no. 5, pp. 895–907, 2020.

[Kat15] K. Katoh and K. Namba, “A low area calibration technique of TDC using variable clock generator for accurate on-line delay measurement,” in IEEE Asia Symposium on Quality Electronic Design (ISQED), pp. 430–434, 2015.

[Ker15] H. G. Kerkhoff and H. Ebrahimi, “Detection of intermittent resistive faults in electronic systems based on the mixed-signal boundary-scan standard,” in IEEE Asia Symposium on Quality Electronic Design (ASQED), pp. 77–82, 2015.

[Ker17] H. G. Kerkhoff, G. Ali, and H. Ebrahimi, “An automotive MP-SoC featuring an advanced embedded instrument infrastructure for high dependability,” in IEEE International Test Conference in Asia (ITC-Asia), pp. 65–70, 2017.

[Kha12] S. Khan, P. Phillips, C. Hockley, et al., “No fault found, retest-ok, cannot duplicate or fault not found?—towards a standardised taxonomy,” in International Through-life Engineering Services Conference, pp. 246–253, 2012.

[Kha14] S. Khan, P. Phillips, I. Jennions, et al., “No fault found events in maintenance engineering part 1: Current trends, implications and organizational practices,” in Elsevier Reliability Engineering & System Safety, volume 123, pp. 183–195, 2014.

[Kim15] T. T. H. Kim, P. F. Lu, K. A. Jenkins, et al., "A ring-oscillator-based reliability monitor for isolated measurement of NBTI and PBTI in high-k/metal gate technology," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, volume 23, no. 7, pp. 1360–1364, 2015.

[Kir04] L. V. Kirkland, T. Pombo, K. Nelson, et al., "Avionics health management: Searching for the prognostics grail," in IEEE Aerospace Conference Proceedings, volume 5, pp. 3448–3454, 2004.

[Kod05] M. Kodera, S. I. Uekusa, H. Nagano, et al., "Stress corrosion cracking of Cu interconnects during CMP with a Cu/porous low-k structure," in Journal of the Electrochemical Society, volume 152, no. 6, pp. 506–510, 2005.

[Kra06] N. Kranitis, A. Merentitis, N. Laoutaris, et al., "Optimal periodic testing of intermittent faults in embedded pipelined processor applications," in IEEE Design, Automation and Test in Europe (DATE), pp. 1–6, 2006.

[Kur12] P. Kurup and T. Abbasi, Logic synthesis using Synopsys, Springer Science & Business Media, 2012, ISBN 978-1-4757-2370-0.

[Kwo11] D. Kwon, M. H. Azarian, and M. Pecht, "Nondestructive sensing of interconnect failure mechanisms using time-domain reflectometry," in IEEE Sensors Journal, volume 11, no. 5, pp. 1236–1241, 2011.

[Kwo15] D. Kwon, M. H. Azarian, and M. Pecht, "Remaining-life prediction of solder joints using RF impedance analysis and gaussian process regression," in IEEE Transactions on Components, Packaging and Manufacturing Technology, volume 5, no. 11, pp. 1602–1609, 2015.

[Lai14] L. Lai, V. Chandra, R. C. Aitken, et al., "SlackProbe: A flexible and efficient in situ timing slack monitoring methodology," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume 33, no. 8, pp. 1168–1179, 2014.

[Lee14] J. Lee and H. Jeong, "Fatigue life prediction of solder joints with consideration of frequency, temperature and cracking energy density," in Elsevier International Journal of Fatigue, volume 61, pp. 264–270, 2014.

[Li05] J. C. M. Li and E. J. McCluskey, "Diagnosis of resistive-open and stuck-open defects in digital CMOS ICs," in IEEE Transactions on Computer-Aided

Design of Integrated Circuits and Systems, volume 24, no. 11, pp. 1748–1759, 2005.

[Lin14] S. Lin and G. W. Roberts, "Towards a general purpose mixed-signal instrumentation layer in the die stack of a 3D-SIC," in IEEE European Test Symposium (ETS), pp. 1–2, 2014.

[Liu15] C. Liu, M. A. Kochte, and H. Wunderlich, "Efficient observation point selection for aging monitoring," in IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 176–181, 2015.

[Liu16] B. Liu, Y. Tian, J. Qin, et al., "Degradation behaviors of micro ball grid array (μbga) solder joints under the coupled effects of electromigration and thermal stress," in Springer Journal of Materials Science: Materials in Electronics, volume 27, no. 11, pp. 11583–11592, 2016.

[Loe12] F. Loete and C. Gilbert, "Diagnostic of connector's degradation level by frequency domain reflectometry," in IEEE Conference on Electrical Contacts (Holm), pp. 1–4, 2012.

[Mar10] E. J. Marinissen, "Testing TSV-based three-dimensional stacked ICs," in IEEE Design, Automation and Test in Europe (DATE), pp. 1689–1694, 2010.

[Mon02] R. R. Montañés, J. P. De Gyvez, and P. Volf, "Resistance characterization for weak open defects," in IEEE Design & Test of Computers, volume 19, no. 5, pp. 18–26, 2002.

[Mön06] L. Mönch, R. Unbehaun, and Y. I. Choung, "Minimizing earliness–tardiness on a single burn-in oven with a common due date and maximum allowable tardiness constraint," in Springer OR Spectrum, volume 28, no. 2, pp. 177–198, 2006.

[Nan08] Nangate, "Nangate 45nm open cell library," `http://www.nangate.com/`, 2008.

[Nek92] M. Nekili and Y. Savaria, "Optimal methods of driving interconnections in VLSI circuits," in IEEE International Symposium on Circuits and Systems, pp. 21–24, 1992.

[Nic98] M. Nicolaidis, Y. Zorian, and D. K. Pradan, On-line Testing for VLSI, Springer, 1998, ISBN 978-0-79-238132-7.

[NoC20] NoCem, “Network on Chip emulator,” `http://www.opencores.org/`, 2020.

[Nov08] F. Novak, F. Azais, P. Nouet, et al., “Implementation of an experimental IEEE 1149.4 mixed- signal test chip,” in IEEE Board Test Workshop (BTW), pp. 1–4, 2008.

[Pan01] J. H. L. Pang, D. Y. R. Chong, and T. H. Low, “Thermal cycling analysis of flip-chip solder joint reliability,” in IEEE Transactions on Components and Packaging Technologies, volume 24, no. 4, pp. 705–712, 2001.

[Pan11] J. Pan and J. Silk, “A study of solder joint failure criteria,” in International Symposium on Microelectronics, pp. 694–702, 2011.

[Pan12] S. Pan, Y. Hu, and X. Li, “IVF: Characterizing the vulnerability of microprocessor structures to intermittent faults,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, volume 20, no. 5, pp. 777–790, 2012.

[Pan14] J. Pan, “A control-chart-based method for solder joint crack detection,” in Journal of Microelectronics and Electronic Packaging, volume 11, no. 3, pp. 94–103, 2014.

[Par08] Y. W. Park, T. S. Narayanan, and K. Y. Lee, “Fretting corrosion of tin-plated contacts,” in Elsevier Tribology International, volume 41, no. 7, pp. 616–628, 2008.

[Par16] K. P. Parker, IEEE 1149.4 Analog Boundary-Scan, Springer, 2016, ISBN 978-3-319-01173-8.

[Pla15] Plasma, “Open cores plasma cpu project,” `http://www.opencores.org/project,plasma`, 2015.

[Qi08] H. Qi, S. Ganesan, and M. Pecht, “No-fault-found and intermittent failures in electronic products,” in Elsevier Microelectronics Reliability, volume 48, no. 5, pp. 663–674, 2008.

[Que16] QuestaSim, “QuestaSim User Guide,” 2016.

[Rah14] A. Rahimi, L. Benini, and R. K. Gupta, “Application-adaptive guardbanding to mitigate static and dynamic variability,” in IEEE Transactions on Computers, volume 63, no. 9, pp. 2160–2173, 2014.

[Raz16] A. Raza and V. Ulanskyi, “Assessing the impact of intermittent failures on the cost of digital avionics maintenance,” in IEEE Aerospace Conference, pp. 1–16, 2016.

[Reb11] B. Rebaud, M. Belleville, E. Beigné, et al., “Timing slack monitoring under process and environmental variations: Application to a DSP performance optimization,” in Elsevier Microelectronics Journal, volume 42, no. 5, pp. 718–732, 2011.

[Rec15] Recore, “Recoresystems,” `http://www.recoresystems.com/`, 2015.

[Roh16] A. Rohani, H. Ebrahimi, and H. G. Kerkhoff, “A software framework to calculate local temperatures in CMOS processors,” in IEEE International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), pp. 183–188, 2016.

[Sad15] M. Sadi, L. Winemberg, and M. Tehranipoor, “A robust digital sensor IP and sensor insertion flow for in-situ path timing slack monitoring in SoCs,” in IEEE VLSI Test Symposium (VTS), pp. 1–6, 2015.

[Sad18] S. Sadeghi Kohan, A. Vafaei, and Z. Navabi, “Near-optimal node selection procedure for aging monitor placement,” in IEEE International Symposium on On-Line Testing And Robust System Design (IOLTS), pp. 6–11, 2018.

[Sal15] M. Saliva, F. Cacho, V. Huard, et al., “Digital circuits reliability with in-situ monitors in 28nm fully depleted SOI,” in IEEE Design, Automation and Test in Europe (DATE), pp. 441–446, 2015.

[Sam18] V. Samavatian, A. Masoumian, M. Mafi, et al., “Influence of directional random vibration on the fatigue life of solder joints in a power module,” in IEEE Transactions on Components, Packaging and Manufacturing Technology, volume 9, no. 2, pp. 262–268, 2018.

[San09] W. SanUm and M. Tachibana, “An on-chip analog mixed-signal testing compliant with IEEE 1149.4 standard using fault signature characterization technique,” in IEEE International Conference on Electrical Engineering/-Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), volume 1, pp. 592–595, 2009.

[Sem03] O. Semenov, A. Vassighi, M. Sachdev, et al., “Effect of cmos technology scaling on thermal management during burn-in,” in IEEE Transactions on Semiconductor Manufacturing, volume 16, no. 4, pp. 686–695, 2003.

[Sha17] W. Shan, L. Shi, and J. Yang, “In-situ timing monitor-based adaptive voltage scaling system for wide-voltage-range applications,” in IEEE Access, volume 5, pp. 15831–15838, 2017.

[Shi14] K. Shibin, S. Devadze, and A. Jutman, “Asynchronous fault detection in IEEE P1687 instrument network,” in IEEE North Atlantic Test Workshop (NATW), pp. 73–78, 2014.

[Ska03] K. Skadron, M. Stan, W. Huang, et al., “Temperature-aware micro-architecture,” in IEEE International Symposium on Computer Architecture (ISCA), pp. 2–13, 2003.

[Ska04] K. Skadron, M. Stan, K. Sankaranarayanan, et al., “Temperature-aware microarchitecture: Modeling and implementation,” in ACM Transactions on Architecture and Code Optimization (TACO), volume 1, no. 1, pp. 94–125, 2004.

[Söd07] P. Söderholm, “A system view of the No Fault Found (NFF) phenomenon,” in Elsevier Reliability Engineering & System Safety, volume 92, no. 1, pp. 1–14, 2007.

[Sor10] B. Sorensen, C. Chambers, and K. Andersen, “The right stuff for aging electronics, intermittence/no fault found,” in White Paper Synaptics Corporation, 2010.

[SPI03] SPI, “SPI block guide v03.06, freescale semiconductor,” 2003.

[Ste02] B. Steadman, T. Pombo, I. Madison, et al., “Reducing no fault found using statistical processing and an expert system,” in IEEE AUTOTESTCON, pp. 872–878, 2002.

[Ste05] B. Steadman, S. Sievert, B. Sorensen, et al., "Attacking bad actor and no fault found electronic boxes," in IEEE Autotestcon, pp. 821–824, 2005.

[Ste08] B. Steadman, F. Berghout, N. Olsen, et al., "Intermittent fault detection and isolation system," in IEEE AUTOTESTCON, pp. 37–40, 2008.

[Sto10] E. A. Stott, J. S. Wong, P. Sedcole, et al., "Degradation in FPGAs: measurement and modelling," in ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), pp. 229–238, 2010.

[Sui18] T. Sui, Z. Zhao, S. Xie, et al., "A 2.3-ps RMS resolution time-to-digital converter implemented in a low-cost Cyclone V FPGA," in IEEE Transactions on Instrumentation and Measurement, volume 68, no. 10, pp. 3647–3660, 2018.

[Sun19] J. Sun, C. Li, X. J. Wu, et al., "An effective method of weld defect detection and classification based on machine vision," in IEEE Transactions on Industrial Informatics, volume 15, no. 12, pp. 6322–6333, 2019.

[Sur20] A. Surendar, M. Kavitha, M. Arun, et al., "Reliability assessment of solder joints in electronic devices under extreme thermal fluctuations," in IEEE Transactions on Components, Packaging and Manufacturing Technology, volume 10, no. 8, pp. 1394–1400, 2020.

[Tho02] D. A. Thomas, K. Ayers, and M. Pecht, "The trouble not identified phenomenon in automotive electronics," in Elsevier Microelectronics Reliability, volume 42, no. 4-5, pp. 641–651, 2002.

[Tre20] Trenz, "TEB0745 carrier board," `https://wiki.trenz-electronic.de/display/PD/TEB0745`, 2020.

[Uns06] O. S. Unsal, J. W. Tschanz, K. Bowman, et al., "Impact of parameter variations on circuits and microarchitecture," in IEEE Micro, volume 26, no. 6, pp. 30–39, 2006.

[Wal11] K. H. Walters, S. Gerez, G. Smit, et al., "Multicore soc for on-board payload signal processing," in NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 17–21, 2011.

[Wan14] J. Wan and H. G. Kerkhoff, “The influence of no fault found in analogue CMOS circuits,” in IEEE International Mixed-Signals, Sensors and Systems Test Workshop (IMS3TW), pp. 1–6, 2014.

[Wan17] Y. Wang, Q. Cao, and C. Liu, “A multi-chain merged tapped delay line for high precision time-to-digital converters in FPGAs,” in IEEE Transactions on Circuits and Systems II: Express Briefs, volume 65, no. 1, pp. 96–100, 2017.

[Wil00] B. Wilkins, “IEEE standard for a mixed-signal test bus,” in IEEE Std 1149.4-1999, 2000.

[Xil14] Xilinx, “7 series FPGAs configurable logic block: User guide,” `http://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf`, 2014.

[Yan10] J. S. Yang, K. Athikulwongse, Y. J. Lee, et al., “TSV stress aware timing analysis with applications to 3D-IC layout optimization,” in IEEE Design Automation Conference, pp. 803–806, 2010.

[Yar15] A. Yaramasu, Y. Cao, G. Liu, et al., “Aircraft electric system intermittent arc fault detection and location,” in IEEE Trans. on Aerospace and Electronic Systems, volume 51, no. 1, pp. 40–51, 2015.

[Yi08] H. Yi, J. Song, and S. Park, “Interconnect delay fault test on boards and SoCs with multiple clock domains,” in Electronics and Telecommunications Research Institute Journal, volume 30, no. 3, pp. 403–411, 2008.

[Zha15] H. Zhang, Y. Liu, J. Wang, et al., “Failure study of solder joints subjected to random vibration loading at different temperatures,” in Springer Journal of Materials Science: Materials in Electronics, volume 26, no. 4, pp. 2374–2379, 2015.