Design of Low-Cost Fault-Tolerant Solutions based on Approximate Computing
Bastien Deveautour

To cite this version:

HAL Id: tel-03361321
https://tel.archives-ouvertes.fr/tel-03361321
Submitted on 1 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
THÈSE POUR OBTENIR LE GRADE DE DOCTEUR
DE L’UNIVERSITÉ DE MONTPELLIER

En SYAM — Systèmes Automatiques et Micro-Electroniques

École doctorale : I2S — Information, Structures et Systèmes

Unité de recherche LIRMM — Laboratoire d’Informatique, de Robotique et de Micro-électronique de Montpellier

CONCEPTION DE SOLUTIONS DE TOLERANCE AUX
FAUTES À FAIBLE COÛT BASEES SUR DES STRUCTURES
APPROXIMEES

DESIGN OF LOW-COST FAULT-TOLERANT SOLUTIONS BASED
ON APPROXIMATE COMPUTING

Présentée par Bastien DEVEAUTOUR
Le 11 décembre 2020

Sous la direction d’Arnaud Virazel
et de Patrick Girard

Devant le jury composé de

Daniel CHILLET, Professeur à l’Université de Rennes 1, IRISA, Lannion
Matteo SONZA REORDA, Professeur au Politecnico di Torino, Italie
Giorgio DI NATALE, Directeur de Recherche au CNRS, TIMA, Grenoble
Arnaud VIRAZEL, Maître de conférence à l’Université de Montpellier, LIRMM, Montpellier
Patrick GIRARD, Directeur de Recherche au CNRS, LIRMM, Montpellier

Rapporteur
Rapporteur
Président
Directeur de Thèse
Co-Directeur de Thèse
ACKNOWLEDGMENT

First of all, I would like to thank my thesis director, Dr. Arnaud Virazel for his priceless patience and support through all these years. My path in research has been greatly shaped by his guidance, advises and trust.

My grateful thanks to my co-director Dr. Patrick Girard whose advices and helpful criticism allowed me to improve the quality of my publications. I certainly look forward to put into practice your observations.

I also would like to thank Dr. Giorgio Di Natale for accepting to review this manuscript. Moreover, I would like to thank the rest of my thesis committee members, Prof. Daniel Chillet and Prof. Mateo Sonza Reorda for the review and positive feedback regarding this manuscript.

I would like to acknowledge to all researchers and LIRMM administrative staff that, during these years, spared me little bits of wisdom. I was all ears all the time.

I wish to thank the support received from my colleagues and friends Mathieu, Caroline, Marcello, Emanuele, Safa, Ilaria, Francesco, Clément and Linh. Thank you for the various discussions, brain storming sessions and coffee breaks. Also for making the last three years memorable.

A huge thank you to my family for the support and the encouragements. Your constant support and understanding was extremely important in this endeavor. I owe you so much of what makes me the person I am today.

Finally, I would like to give a very special thank the love of my life, my wonderful wife Katherine whose support and blind faith in me gave me the strength to keep going forward. Her unwavering patience and cheering meant everything to me and none of this would have been possible without it.
# Table of Contents

Introduction .......................................................................................................................... 7

Chapter I : Context and Motivation ..................................................................................... 10

I.1. Fifty years of semiconductor technology scaling ....................................................... 11
I.2. Reliability issues in nanometer technologies ............................................................ 12
  I.2.A. Manufacturing defects and variability ................................................................. 13
  I.2.B. Interferences ....................................................................................................... 13
  I.2.C. Wear-Out ........................................................................................................... 14
I.3. Errors in Integrated Circuits ....................................................................................... 14
  I.3.A. Soft Errors ........................................................................................................ 15
  I.3.B. Hard Errors ...................................................................................................... 18
  I.3.C. Timing Errors .................................................................................................... 19
I.4. Reliability Improvement Approaches ......................................................................... 19
  I.4.A. Fault Avoidance ............................................................................................... 19
  I.4.B. Fault Removal .................................................................................................. 20
  I.4.C. Fault Tolerance ................................................................................................. 20
I.5. Research Objectives .................................................................................................... 21

Conclusion ............................................................................................................................. 23

Chapter II : State of the Art ............................................................................................... 24

II.1. Fault Detection Techniques ....................................................................................... 25
  II.1.A. Duplication with Comparison ........................................................................... 25
  II.1.B. Error Detecting Codes ..................................................................................... 26
II.2. Fault Correction Techniques ..................................................................................... 27
  II.2.A. Rollback Error Recovery ................................................................................ 27
  II.2.B. Forward Error Recovery ................................................................................ 28
II.3. Fault-Tolerant Architectures ..................................................................................... 29
  II.3.B. Razor .............................................................................................................. 30
  II.3.C. STEM .............................................................................................................. 31
  II.3.D. CPipe ................................................................................................................ 32
  II.3.E. TMR .................................................................................................................. 32
  II.3.F. DARA-TMR ..................................................................................................... 34
  II.3.G. Architecture Comparison ............................................................................... 34
II.4. Selective Hardening Approaches .............................................................................. 35
INTRODUCTION

Technology scaling allows a greater integration of transistors on a single chip, which favors the design of systems more and more complex. This high level of integration leads to increased power and current densities, thus leading to early device and interconnect wear-out. As a result, Complementary Metal-Oxide Semiconductor (CMOS) technology induces shifts in electrical characteristics of circuit components and even permanent damages. Besides these wear-out or aging issues, increasing integration density makes the testing of complex systems very difficult. This testing complexity can lead defects to escape manufacturing test and manifest themselves, only later, during in-field operations. Moreover, beside failures caused by manufacturing defects, transistors are susceptible to high-energy particles from the atmosphere or the packaging. This susceptibility increases the system failure occurrence. These phenomena, also known as Single Event Effect (SEE), are more prone to occur on devices operating at reduced supply voltages and high frequencies. SEE causes particle-induced voltage transients usually called Single Event Transient (SET) or particle-induced bit-flips in memory elements also called Single Event Upset (SEU).

Technology scaling allows higher operating frequencies, higher logic complexity, increasing power density, and hence requires complex manufacturing processes. High-performance microprocessors are usually at the forefront of these technology advances and thus, are susceptible to errors. Errors can be hard when a circuit has a permanent defect or soft when caused by temporary effects called transient fault. For a long time, Soft Error Rate (SER) was considered to be exclusively a consequence of particle strikes in memory elements. However, it is now known that SEE in Combinational Logic (CL) plays an important role in the increase of SER. Thus, it becomes impossible for industry to ignore these issues as reliability is a bottleneck for the development of high-performance and low power microprocessors.

Techniques that deal with reliability issues inherent to nanometric circuits must consider threats at the technological, manufacturing and design levels. Although essential, existing techniques are often imperfect and fault occurrence remains because of test-escapes, particle strikes or aging. New Fault-Tolerant techniques are therefore necessary to lower the impact of in-field operation faults by using information, timing and hardware redundancies. They guarantee the proper system operation regardless of the presence of faults.
Usually, many of the fault-tolerant techniques found in the literature improve the robustness of systems by dealing with faults due to aging, wear-out or particle striking. Conversely, very few of these techniques can address both permanent and transient faults. Among these techniques, some of them rely on recovery mechanisms that imply a significant delay unsuitable for highly interactive applications. For example, the architecture introduced in [MEH2007] induces a little area overhead but has a severe impact on performance as it uses deep rollbacks and Built-In Self-Test (BIST) during periodic time intervals to detect the presence of permanent faults. Other fault-tolerant architectures like Razor [ERN2003], CPipe [SUB2008] and STEM [AVI2012] embed power saving and performance enhancement mechanisms like Dynamic Frequency Scaling (DFS) and Dynamic Voltage Scaling (DVS). However, these techniques usually allow to deal with timing errors but cannot handle permanent faults. Indeed, Razor only corrects timing faults and CPipe detects and corrects transient and timing fault but can only detect permanent faults without offering any correction solution. The outcome of these partial fault-tolerant techniques is an area and power resource saving at the expense of limited reliability.

The cost of fault-tolerant techniques is a rising concern in industry, especially power consumption. Even if fault-tolerance is necessary in mass products, excessive power consumption induced by these techniques is one of the key factors in digital design. Unfortunately, the above-mentioned partial fault-tolerant techniques cannot guarantee both power saving and full reliability as achieved by energy-consuming structures. This is the case of the well-known Triple Modular Redundancy (TMR). This solution is able to deal with timing, transient and even permanent faults but comes at a very high area and power overhead. Theoretically, the cost associated to fault tolerant techniques has a direct implication on the provided reliability.

In fault-tolerant techniques, an important factor is the reliability level required by the workload. Some safety-critical applications, like space or medical applications, cannot accept any error. Other applications that deliver signals to human senses can tolerate imprecisions that may alter the results of a precise computation as long as the inaccurate human perception does not notice it (e.g., typical applications are image processing or Recognition, Mining and Synthesis (RMS) applications. The resilience of these applications has led to new design approaches that willingly sacrifice accuracy to reduce area, power and timing costs. This concept is known as Approximate Computing (AxC) and it offers good trade-off between cost and reliability.

Selecting the ideal trade-off between reliability and cost associated with a fault-tolerant architecture generally involves an extensive design space exploration. This thesis proposes new fault-tolerant approaches that offer a better cost versus reliability
trade-off than conventional approaches existing in the literature. Another objective is to explore the possibility to develop fully reliable architectures at a cost lower than that usually found in safety-critical systems. Overall, this thesis highlights the interest of using the concept of AxC in fault-tolerant techniques based on hardware redundancy.

This manuscript is divided into four chapters:

- Chapter I describes the context and motivations of this thesis. First, it introduces the trends in semiconductor technology scaling and their impact on the reliability of nanometer circuits. Then, it reviews existing reliability improvement approaches. The chapter ends by a discussion about the objectives of the work accomplished during this thesis.
- Chapter II presents the state-of-the-art in the areas of fault-tolerant architectures and robustness assessment techniques. The discussion covers some basic concepts of error detection and correction, and then addresses fault-tolerant architectures existing in the literature. Next, we give an overview of selective hardening techniques and the trade-off between cost and reliability. The chapter ends with an introduction to AxC and its applications that show the growing interest of using AxC in fault-tolerant techniques.
- Chapter III demonstrates the usefulness of using AxC blocks to replace precise CL in a duplication and comparison fault detection structure for resilient applications. This solution is compared to three other scenarios: a full duplication comparison scheme with two precise CL as baseline and two selective duplication and comparison approaches. Experimental results with fault injection campaigns prove the advantages of using AxC in selective hardening.
- Chapter IV explores a novel design concept made exclusively of AxC blocks that reaches reliability levels equal to costly fault-tolerant architectures. First, we discuss the existing AxC based fault-tolerant architectures and their limitations. Next, we expose the concept of Quadruple Approximate Modular Redundancy (QAMR) and its goal. Experimental results show that QAMR is a good alternative to TMR structures, and that a deep design space exploration can lead to better cost-reliability trade-off.
CHAPTER I
CONTEXT AND MOTIVATION

Since the early 1970s, the demand in electronic components and the necessity to push the limits of manufactured circuits for increased performance and transistor density has never stopped. Consequently, each new generation of microprocessor suffers from reliability issues due to manufacturing defects, variability, interferences, and wear-out. These well-known drawbacks lead to the occurrence of faults that can finally cause system failures in integrated circuits. Several reliability improvement approaches exist and allow integrated circuits to work as intended.

This chapter is divided in five sections as follows. The first section briefly discusses the technological evolution of semiconductors and their scaling. The second section reviews reliability issues from different nature that can cause errors in integrated circuits at nanometer scale. The third section gives a classification of such errors and explains how they disturb the correct functioning of a circuit. The fourth section details various approaches to improve the reliability of integrated circuits and explains how to deal with errors. Finally, the last section defines the objectives of this thesis regarding cost effective ways to protect the combinational logic in microprocessors.
I.1. Fifty years of semiconductor technology scaling

In 1971, Intel proposed the first single chip microprocessor named 4004. With 2300 transistors, the 4004 was capable to run at a maximum clock speed of 740 kHz and perform between 46,250 and 92,500 instructions per second dissipating 0.5 Watts. From this first microprocessor, the next five decades showed a constant evolution following Moore’s law. This law is a result of the empirical experience acquired in production by the co-founder of Intel, Gordon Moore who observed that the number of transistors in a dense integrated circuit (IC) doubles about every two years. Thank to this growth, the computing power increased exponentially, allowing the emergence of various applications such as climate modelling, protein folding, electronic games and autonomous soft landings on extra-terrestrial bodies.

![Figure I.1 CMOS Technology scaling](image)

Nowadays, microprocessor chips embed billions of transistors and even multiple processor cores on a single silicon die. The clock speeds at which current microprocessors run is measured in gigahertz and allow to perform more than four million times the performance of the first 4004 Intel processor [AND2012]. Figure I.1 plots some trends of the technological advancements in microprocessors from 1971 until 2018. In 2019, the Wafer Scale Engine from Cerebras was developed with the highest transistor count in a non-memory chip, a deep learning engine that has 1.2 trillion MOSFET transistors manufactured using a TSMC’s 16 nm FinFET technology [WAF2019]. This huge advancement in semiconductor technology is the result of major contributions from the Electronic Design Automation (EDA), manufacturing lithography and advanced semiconductor material industries [NIS2007].
Figure I.1 also shows that around 2005, the frequency scaling trend slowed down by hitting a power wall around 3 GHz. By increasing the switching activity of transistors, the power consumption increases as well. Moreover, the frequency limitation is also due to the power density. Indeed, the million transistor density pauses heat dissipation challenges that further restrains frequency scaling to avoid breakdown of physical materials. Figure I.1 shows that these limitations were circumvented by designing multicore architectures.

The evolution of semiconductor technology would not be as it is today without the continuous high demand of new high performance, enhanced featured and low-power integrated circuits. Without this demand, the initial investment to build smaller physical structures would not be sustainable, as these microprocessors require expensive sophisticated equipment. To give an example, in November 2019, TSMC started the development of a 3nm plant in Taiwan with a cost estimated at $20B [HAM2019] [ZAF2020].

I.2. Reliability issues in nanometer technologies

The reliability of digital circuits and systems is kept high owing to several methods. These methods ensure that the designs achieve their function under defined conditions and during their estimated lifespan. They cover different aspects of the well-manufacturing and well-functioning of semiconductors. Cleanrooms control impurities, industrial control systems achieve production consistency. Burn-in, and testing before and after packaging, ensure the detection of design weaknesses and manufacturing defects after stressing the circuits. All these methods are necessary before introducing the semiconductors to the market but they are not fool proof.
Even though miniaturization offers many advantages, each new CMOS node faces reliability issues as the trend reaches physical limits of operation and manufacturing [ITR2011]. Digital systems can experience failures during the three phases of their lifespan depicted in the bathtub curve in figure 1.2. Early failures are labelled as infant mortality; random failures occur during the working life and wear-out failures happen at the end of the circuit’s lifespan.

1.2.A. Manufacturing defects and variability

Early failures during the infant mortality phase are mainly due to manufacturing issues. During the many steps that include implantation, etching, deposition, polarization, cleaning and lithography [ITR2013], imperfections can induce permanent defects in a chip. Variability of transistor characteristics due to variations in Process, Voltage and Temperature (PVT) has always been an issue in integrated circuit design [WIR2015]. PVT variations prevent the circuit from functioning correctly even though each individual transistor behaves correctly [ODA2015]. Indeed, in nanoscale CMOS technology, transistors are so small that printing errors below the wavelength of light and variations in the discrete number of dopant atoms have major effects on their performance.

Furthermore, even with the number of transistors doubling every 2 to 3 years, according to past microprocessor data, the die size remains relatively constant [HUA2010]. This scaling inevitably leads to an increase in chip power densities and inadequate heat sinking causes hot spots to appear. These fluctuations of ambient temperatures alter the timing characteristics of circuits [KUM2008].

Inevitably, the defect density increases continuously and so do the logic complexity that entails the emergence of defects each time subtler and thus, harder to detect [SAN2008, SEG2004]. Considering all these possible issues, the testing of potentially defective chips needs to be as accurate as possible before releasing any product to the market.

1.2.B. Interferences

Variability generated by manufacturing imperfections may generate unexpected circuit behaviors during operation. Furthermore, as transistors become smaller, their supply voltage ($V_{dd}$) decreases. These conditions are favorable for the occurrence of temporary effects like transient or intermittent faults. These temporary effects can be the result of electromagnetic influences, alpha-particle emission or cosmic radiations. They are responsible for the greatest part of digital malfunctions and more than 90% of the total maintenance costs are credited to them [SAC2013].
Internal interferences can also be a cause of temporary malfunctioning. With the scaling of components, the scaling of the interconnect line thickness (width and separation) must also follow. In these conditions, a high crosstalk noise is becoming a major issue due to larger capacitive couplings between interconnects in a polluted environment. Additionally, supply voltage scaling lowers the noise sensitivity threshold and increases the transient fault sensitivity of new technology nodes due to high energy particle from environment or within the packaging.

I.2.C. Wear-Out

Although area scaling follows an exponential trend, supply voltage had a way slower scaling pace. There are two main reasons for such slow supply voltage scaling. The first one is the need to keep up with the competitive frequency growth. The second is to retain the basic noise immunity and cell stability [SRI2004-1]. As a result, the discrepancy between area and voltage scaling leads to high power density and elevated temperatures. The increase of temperature is responsible for four well-known wear-out mechanisms: Time Dependent Dielectric Breakdown (TDDB), Electromigration (EM), Thermal Cycling and stress migration.

Electromigration is the main cause of interconnect wear-out [HAL2020]. The high unidirectional current can reach a density which is high enough to drift the metal ions in the direction of the electron flow. This phenomenon leads to variations in the resistance of interconnects and causes modifications in the timing characteristics of the design. Electromigration can last until the extreme case where the metal runs out, creating a void and thus, an open in the metal line.

Wear-out failures appear in-field after a certain period of use and limit both performances and lifetime of modern microprocessors [SRI2004-2]. This is especially critical for applications which demand high throughput (e.g. data centers) or which technical support is expensive (e.g. space equipment)

I.3. Errors in Integrated Circuits

When a fault propagates through the logic, it can be captured by a memory cell (or flip-flop) and stored as faulty value. As seen previously, faults may have different causes: manufacturing defects, variability, interferences and aging. They can be classified according to their duration in three categories: transient, intermittent and permanent faults.
Transient faults randomly affect the correct functioning of the Integrated Circuit (IC) for a short time window. After this period, the device returns to a normal behavior. Variability and interferences are the main causes of transient faults.

Intermittent faults occur randomly like transient faults, but they never really disappear. In fact, their occurrence often precedes the occurrence of a permanent fault. Aging is the primary cause of intermittent faults.

Permanent faults are irreversible and are mostly due to manufacturing defects. They can also appear at the end of the circuit’s lifetime due to extreme wear-out effects.

Depending on their nature, faults can become hard or soft errors that may cause a subsequent failure if the error reaches the service interface and alters the service [AVI2001]. In integrated circuits, an error can be classified depending on its temporal characteristics [LEH2005], its severity, the product life cycle stage of its induction, etc. However, in the following sub-sections, the error classification is made according to their underlying fault type.

I.3.A. Soft Errors

Soft errors occur when particles like high-energy neutrons from cosmic rays or alpha particles generated from impurities in the packaging strike a sensitive zone of the microelectronic device.

Whenever a particle strikes the silicon, the fission of the ion shatters the silicon atoms forming a cylindrical track of electron hole pairs. If the ionization track is formed near the depletion node, the particle-induced charge can be very efficiently collected through drift processes and lead to a transient current at the junction contact. In tens of picoseconds, the collection is completed and the diffusion phase follows where carriers generated beyond the depletion zone can diffuse back toward it.

Figure I.2 illustrates a particle striking process where the junction can locally collapse when a charge is generated along the particle track due to its highly conductive nature and to the separation of charge by the depletion region field [BAU2005]. Figure I.2 also shows that the junction electric field can extend beyond the junction and reach deep into the substrate due to the increase charge collection at the strike node caused by the tunneling effect. The efficient drift process can collect such charge deposited away from the junction [DOD2003].
Following such mechanism, the soft error causes voltage glitches at struck nodes. Combinational logic parts of the IC propagate these glitches considered as Single-Event Transient (SET) according to the terminology formerly used in [BEN2004], [FER2013] and [DOD2004]. If the propagation of the SET reaches memory elements of the circuits and their value is captured, the value is changed (bit-flip). This phenomenon is called Single-Event Upset (SEU). However, for the value of a node to flip completely, the collection of a certain amount of charge is required. This quantity of charge relies on the gate capacitance and voltage of the node. The increase of soft errors and their growing impact on ICs during operation is related to the downscaling of the gate capacitance and the supply voltage. In brief, SEU is considered as the result of a fault that propagates through the logic, as the direct consequence of a particle strike, and can lead to a soft error.

Although this SEU definition gives a very precise description of the phenomenon, for practical reason, this manuscript will consider that given in [DOD2003], [SHI2002] and [GOE2008]. By considering SEU as a soft error in memory elements, the definition is more suitable for system level analysis since it ignores at which level the issue hails from. The definition is as follows: “Radiation-induced errors in microelectronic circuits caused when charged particles (usually from the radiation belts or from cosmic rays) lose energy by ionizing the medium through which they pass, leaving behind a wake of electron-hole pairs” [NAS2012].
Soft Error Rate

The failure rate induced by soft errors, or Soft Error Rate (SER), is reported in FIT (Failure In Time) or MTBF (Mean Time Between Failure). In terms of occurrence rate, SER will be many times higher than the hard failure rate of all other mechanisms combined. SER not only increases according to the shrinking of electronic devices. The environment can drastically affect it. Indeed, in avionics applications the neutron flux can be hundreds of times denser than ground-level applications. Shielding the ICs is a partial solution that cannot fully counter the particle strikes. There are no real standards on an acceptable SER. The SER is different for each application since it depends on how much memory is present, whether or not the memory is protected, in which environment the application is operating (e.g. ground-level, aviation, nuclear power plant, etc.), etc.

In the late 1990s, researches on Soft Error Rate (SER) limitation techniques caused by SEE in CL circuits emerged. The objective was to reduce the impact that SETs in CL that result in SEUs. Researchers realize that, with advanced technologies and effective detection and correction techniques, memory soft errors could be kept under control.

In 1994, according to Lidén et al., the SEUs originated from SETs impacting CL was minimal. In their study, they stated that only 2% of the bit flips were originated from SETs generated and propagated through CL. Particle strikes directly in latches were the main cause for the rest of the SEUs [LID1994]. Later in 2004, Shivakumar et al. established that the supply voltage and the transistor gate length downscaling were exposing memory arrays even more to high-energy particles but also noted that the increase on CL stages were more pronounced [SHI2002].

In 2011 a study used a probability model to estimate that susceptibility of CL to SETs nearly doubles as the technology nodes scale from 45 nm to 16 nm [VEL2011]. In another study from 2014, circuits with 40 nm, 28nm and 20 nm nodes were exposed to alpha particles to investigate the voltage and frequency dependence of combinational logic and flip-flop circuits. They state “At higher frequencies, the logic SER will certainly be comparable to the latch SER and could exceed it as well” [MAH2014].

Nowadays, designers implement detection and correction techniques in CL to prevent soft errors from altering the correct functioning of a device.

Soft Error masking categories

Previous statements describe how technology scaling has a negative impact on the susceptibility of CL nodes to particle strikes. However, it also lowered the natural
limitations of SET propagation through CL stages. Those limitations are known as masking effects that prevent an SET to become a soft error:

- **Electrical Masking**: it happens when the voltage transient resulting from a striking particle is attenuated by subsequent logic gates because of the electrical property of the logic gate [KAR2004]. If a pulse loses strength while propagating through a sensitized path or completely disappear before reaching a memory element, then the SET is referred to be electrically masked [MAS2008].
- **Latching-window Masking**: it means that the arrival of a transient pulse with enough amplitude to consider it as a valid logic level is outside of the latching window for the sequential element(s). This SET pulse will not affect the stored data due to the latching-window masking effect [GEO2011].
- **Logical Masking**: it happens when one of the other inputs (unaffected by the SET) of a gate is in controlling state (e.g., 0 for a NAND gate), so that the transient is blocked. For a SET to propagate through CL and result in a soft error, it is necessary that the path from the point of SEE generation to a memory element is functionally sensitized during the time of SEE propagation [GEO2011]. This depends on the input vector being applied at the time of the SEE propagation.

Once again, technology scaling favors SEU occurrence as it lowers the impact of the masking effects against SET propagation. Electrical masking tends to diminish because the SET attenuation effect is weaker within faster transistors. With high operating frequencies, the latching windows are more frequent. This fact increases the probability for a SET to be latched and become an SEU. The less affected masking effect is logical masking since its masking effect does not depend on scaling [SHI2002]. Consequently, research attention was drawn towards developing techniques to reduce the impact of SETs in CL. The provided efforts became comparable to efforts made in protecting state elements.

### 1.3.B. Hard Errors

Unlike soft errors, hard errors are due to permanent silicon defects. They may be due to imperfections in the manufacturing process as discussed in sub-section 1.2.A or to the wear-out effect discussed in sub-section 1.2.C. In the last decades, as transistor density increased, processor performances kept increasing as well. However, this trend also increases constantly likelihood of getting more hard errors in a given core. In addition, the high frequencies increase the switching activity rate that accelerate material aging due to temperature and voltage stress [CHE2015]. Furthermore, the connectivity complexity between the several different stages of high-performance processing cores increased to match the higher transistor integration. This
connectivity supports advanced features like: hazard detection, branch prediction or even data forwarding. These features and their connectivity are very challenging in terms of error confinement efforts [WAL2015].

The deteriorating effects of the previously introduced failure mechanisms like \textit{TDDB} in gate oxides and \textit{EM} in interconnects increase with transistor shrinking. Besides, the major risk for the reliability of the system is the degradation of device parameters through the lifespan of the IC. Circuits are particularly prone to wear-out from Negative Bias Temperature Instability (\textit{NBTI}) and Hot Carrier Injection (\textit{HCI}) that cause electrical parameter alteration such as an increase of the transistor threshold voltage. According to [KEA2011], \textit{NBTI} and \textit{HCI} are serious concerns with technology scaling as they could result in significant degradation of the circuit over its lifespan.

\section*{I.3.C. Timing Errors}

Unlike hard and soft errors, components that suffer from timing error still provide correct logic outputs. However, they have higher delays between input and output signal establishments. Faults induced by \textit{PVT} variability, manufacturing defects and aging phenomenon are responsible for this type of error. Timing behaviors are also less predictable in today’s ICs as technology keep scaling. This uncertainty translates in occasional timing errors on speed-paths, i.e., near-critical paths [MAT2014].

\section*{I.4. Reliability Improvement Approaches}

Previous section discussed about errors and some of them arise during the lifetime of circuits. This means that efficient Automatic Test Equipment (ATE) used after manufacturing are not sufficient to achieve reliability goals. Indeed, it is difficult to identify future sources of errors. To reduce the susceptibility to faults of ICs, every cycle of the design process must include reliability improvement practices. Each phase of the design process requires a deep understanding of reliability needs of the design. In [HEI1992], Heimerdinger et al. characterize the reliability improvement practices according to their chronology in the product development and life cycle as fault avoidance, fault removal and fault tolerance.

\section*{I.4.A. Fault Avoidance}

Fault avoidance aims at minimizing the sensibility of ICs to faults. To do so, specific tools and techniques assist the designers to specify, design and manufacture systems [SHI2007] by addressing the source of the mechanisms causing the failures as shown in Figure I.3. Use of meticulous methods in the specification phase may reduce the impact of those mechanisms and avoid faults [RUS1993]. Some non-exhaustive actions
can be taken to avoid faults. The resizing of transistors within critical gate during design phase helps to decrease the susceptibility of the circuit to soft errors [ZHO2006]. Including technology mitigation techniques to modify conventional manufacturing processes may also help. For example, using Silicon-On-Insulator (SOI) technology on modern chips to reduce significantly the susceptibility to soft errors due to the smaller volume for charge collection [HAR2001] is a well-known solution. Also, using radiation hardened components or adopting standards of quality during manufacturing process like ensuring high cleanroom standards is another possibility.

<table>
<thead>
<tr>
<th>Fault Avoidance</th>
<th>Fault Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Fault Tolerance</td>
<td>Dynamic Fault Tolerance</td>
</tr>
</tbody>
</table>

**Figure I.3**  Reliability improvement approaches across the fault to failure life cycle [CAS2015]

**I.4.B. Fault Removal**

Fault removal includes a large spectrum of approaches whose function is to detect and eliminate existing faults during specification and design. Fault removal also refers to removing faulty components during production and operational phases. Some of the various methods for fault removal include formal verification, design rule checking, signal integrity analysis, static timing analysis, etc. during sign-off to locate faults in specification or design. These methods expose the last necessary changes to be done before tape-out. Burn-in techniques can be used to discard defective chips after manufacturing so they do not end up in systems that have very low failure tolerance. Design For Test (DFT) structures like scan chains and online/offline tests, etc. are also embedded on chips to remove faults during the IC lifespan [SHI2007].

**I.4.C. Fault Tolerance**

Until now a good way to prevent permanent and transient faults in ICs is to improve manufacturing processes to reduce defects and variability and test the components to remove faulty parts that may jeopardize the design. These actions are not easy to apply and do not solve random failures. Even the best efforts and investments to avoid or remove faults cannot prevent them from appearing in any operational system. However, it is possible to prevent those faults using hardware fault tolerant techniques (see Figure I.3). Some of these techniques like masking are considered static whereas some other like Error Correcting codes are dynamic.
Both static and dynamic fault-tolerant techniques are further discussed in chapter II. Fault tolerance aims at guaranteeing the service provided by the product despite the presence or appearance of faults [HEI1992].

Note that some fault-tolerant designs are destined to resilient application. Resiliency being the ability to provide and maintain an acceptable level of service despite faults occurring in the process. In these cases, the fault-tolerant is designed to limit the faults under an established level of impact.

All these fault-tolerant techniques are all settled on the common ground of redundancy. A principle introduced by John von Neumann in 1950s [NEU1956]. The direct idea is to improve the reliability of the system by adding a redundancy that could be classified as structural, temporal or of information according to Mathew et. al. [MAT2014].

- **Structural Redundancy**: refers to techniques, which employ extra hardware treating the same information. The inclusion of a logic voting from the multiple redundant outputs to a single output allows the mitigation of transient or permanent faults effects. The extra hardware resources used to achieve reliability incurs area and power overheads [MAH2004].

- **Temporal Redundancy**: the principle of temporal redundancy is to repeat a computation or a transmission and compare them to the original one [DUB2013]. Spatial redundancy implies additional area and power costs. In some cases, it is preferable to avoid those expenses and spend some extra computation or transmission cycles to tolerate faults. Therefore, temporal redundancy generally sacrifices computing performance in order re-compute or retransmit data using the same hardware resources.

- **Information Redundancy**: Particularly used in memory devices, information redundancy uses detection and correction codes that are integrated with the original circuit data [SOS1994]. These extra information codes are generated from the original logic data to effectively identify the presence of one or more transient or permanent faults and possibly correct them.

I.5. Research Objectives

The work in this thesis aims to improve the transient, permanent and timing error reliability of future microprocessor systems adapted to the application resiliency. These works focus on error detection and correction in CL parts of logic circuits. As discussed in Section I.3.A, CL networks are becoming increasingly susceptible to SEE. In addition, pronounced variability and power densities either cause the electrical
characteristics of these node to change, resulting in timing or hard errors. As a result, the research attention drawn towards developing techniques to limit SER in CL is becoming comparable to effort made in protecting state elements. However, fault-tolerant techniques imply the use of redundancy and thus, require a non-negligible additional cost in computation cycles, area and or power.

The objective of the work discussed in this thesis is to investigate cost-adapted approaches of fault-tolerant architectures depending on the required reliability level. A special focus is made on the trade-off between reliability and its consequent cost. The first step is to develop an approach suitable for applications with different levels of resiliency. The second step is to design a fault-tolerant architecture satisfying a reliability level at a lower cost when compared to existing architectures that present a similar reliability level.
Conclusion

Fifty years of technology scaling clearly exposed transistors and interconnects to an increasing occurrence reliability hazards. Researchers have observed and classified the different phenomena that affect the correct functioning of integrated circuits. Also, they have defined different kind of errors that could result from those phenomena. A clear understanding of the nature of those errors is important as it allows using the correct reliability improvement approach that the circuit requires to operate correctly. One of these reliability improvement approaches is fault-tolerance and the work detailed in this thesis focuses on developing cost effective fault-tolerant approaches
CHAPTER II
STATE OF THE ART

Fault occurrence in high performance and low power systems represents a major problem during their development. To prevent these faults, design architects need to address these reliability issues by making sure the system embeds detection and correction capabilities. Fault-Tolerant techniques prevent faults from occurring during the normal activity of a device at system-level (e.g. microprocessor, System-On-Chip, etc.). These hardening techniques use combinational and sequential redundant logic to detect, correct or mask faults regardless of their transient or permanent nature [KOR2010]. The addition of this redundant logic implies a cost overhead according to three parameters: area, power and timing.

This chapter first discusses some fault detection techniques commonly used in fault-tolerance. The second section of this chapter focuses on relevant fault correction techniques and different approaches to recover from a fault. The third section reviews existing Fault-Tolerant architectures. The fourth section discusses about the trade-off between costs (area, power and timing) and reliability possible through selective hardening. The fifth section introduces the concept of Approximate Computing (AxC), quickly reviews the existing fault-tolerant architectures based on AxC, and discusses its advantages in selective hardening approaches. The next section summarizes the different fault-tolerant techniques. Finally, a section is dedicated to evaluation methods that guarantee the quality of a device even when it is subject to faults.
II.1. Fault Detection Techniques

Due to the increasing complexity of chips, exhaustive testing of defaults at the end of the production line is still dominant in the testing field. It allows the detection of the majority of manufacturing defects. However, some of these defects can escape manufacturing testing, and may appear at any time once the chip is in-field. In addition, transient faults can occur at unpredictable places and times during in-field operation. Similarly, the system will inexorably experiment failures due to aging effects in its lifespan. It is therefore imperative to detect faults during in-field operation to avoid a system failure or data corruption. Error detection during the lifetime functional operation of a system is called on-line detection [GOE2008].

II.1.A. Duplication with Comparison

Duplication with comparison is a fault detection technique commonly used. It is simple to implement and can prevent faults occurring in a digital circuit. Since it is hardware based, the redundancy implies the use of two identical copies of a circuit whose outputs are compared. In case an error affects one copy of the circuit, the fault is detected due to the inequality of the results computed by the two copies. An error flag coming from a comparator usually notifies inequality. Figure II.1 shows the simplest implementation of the duplication with comparison technique. The simplicity of its implementation combined with its ability to detect a wide variety of faults (permanent, transient and timing) is the main reason of its popularity.

The area cost of the duplication with comparison technique is usually twice the area of the original circuit plus the area of the comparator. This area overhead implies a proportional overhead in power consumption but no extra timing cost — when no errors are detected — since the original logical paths are not affected by the added logic.
II.1.B. Error Detecting Codes

Error detecting codes are another category of fault detection techniques widely used for digital circuits. As for duplication and comparison, this category makes use of hardware redundancy to detect faults. The redundancy is implemented through information representation to detect possible errors in that representation [LUS2004]. The result of an operation computed by a logic function is guaranteed to be correct if some special characteristics are respected. These characteristics (e.g. parity, checksum, etc.) are generally predicted by a smaller logic circuit and compared to the same characteristics extracted from the logic circuit’s output in a checker. Figure II.2 shows a logic circuit that performs a function $f$ on the $n$-bit input data $i$ and produces a $m$-bit output $f(i)$. From the data input, the predictor extracts the $k$-bit predicted characteristic $C(i)$. In addition, the checker computes the $k$-bit characteristic $C'(f(i))$ from the data output and compares both characteristics to ensure the integrity of the data.

![Figure II.2 General Architecture of Error Detection with Codes](MIT2000)

Error detecting codes are mostly used for the protection of memories since these techniques can provide protection against SEUs and permanent faults [PET1972]. The reason of the widespread use in memories is their regular structure, which allows an efficient insertion in already regular structures [DUT2008]. The opportunity of using error detection codes in logic circuits to detect malfunctioning relies on its reduced integration cost. However, this method implies a specific design of the original circuit to be effective [GOE2008]. In that configuration, error detection codes have lower area and power costs than duplication and comparison techniques and no timing impact when no faults are detected.
II.2. Fault Correction Techniques

Although fault detection is capable of notifying the system about a fault occurrence, it is useless—apart from being purely informative—without a proper method to recover from it. For this reason, Fault-Tolerant schemes include error recovery mechanisms that follow the detection of a fault. The recovery consists of the restoration of the last error-free state of the system and/or the prevention of the fault from occurring again. Fault masking is another option to prevent a fault from propagating and eventually cause an error.

II.2.A. Rollback Error Recovery

Rollback consists in repeating the last operation(s) by returning to a known fault-free state. Usually, architectures based on rollback recovery detect faults through physical redundancy and correct them by applying temporal redundancy. Periodical or occasional checkpoints save the state of the system. These fault-free checkpoints are the starting point from which the previously faulty operations are recomputed. According to Mehrara et al. in [MEH2007] these rollbacks can backtrack the system to several thousand previous states or can simply just recover one cycle deep [TRA2011].
Figure II.3a shows the example of a rollback recovery of one cycle deep. In this case, a concurrent fault detection mechanism checks for errors at the end of each cycle. If an error is detected, the next cycle will re-execute the instruction.

An example of a much deeper rollback recovery scheme is shown in figure II.3b. In this case, the computation takes several cycles. When no errors are detected, the state of the system is saved as checkpoint data. The state includes all the data necessary for the recover to happen (contents of pipeline register, control registers, memory update information, etc.). Whenever an error is detected, the system recovers the last saved checkpoint data to return to an error-free state and the previously affected computation is repeated.

On one hand, fine-grained rollback recovery schemes incur high-energy costs but their integration is less intrusive. They allow frequent error checks and need a minimal amount of stored check-pointing data. On the other hand, coarse-grained rollback recovery schemes demand a significant storage in order to store check-pointing data. This is especially true when the instructions involve memory updates. Additionally, these kinds of rollback imply intrusive software based recovery sequences and check-pointing procedures. Besides implementation cost, coarse-grained recovery schemes can have significant performance overhead, particularly in high error rate environments [YAO2009]. This technique implies lower area and power costs than the techniques using logic redundancy. However, in case of an error recovery, power and timing costs are proportional to the quantity of cycles that are recovered.

II.2.B. Forward Error Recovery

To avoid the delay resulting from timing redundancy, correction techniques can execute a forward error recovery in case of fault occurrence. In that case, the data is corrected and available at or before the time it is normally propagated through the pipeline. Error Masking is a perfect example of forward error recovery scheme. Contrasting with the rollback recovery that may require re-computing several previous cycles, the recovery mechanism takes action as soon as a fault is detected. Since these techniques do not rely on temporal redundancy, their use is efficient in tight deadline applications [SUR2014].

Forward Error recovery schemes comes with a significant amount of area cost due to the spatial redundancy applied generally for both detection and correction. This area cost is permanent and entails a proportional power cost addition since the redundancy is constantly active irrespective of fault occurrence. From the redundant
resource allocation point of view, rollback recovery schemes perform better since the temporal redundancy is only requested when a fault is detected [NAE2008].

II.3. Fault-Tolerant Architectures

In the last years, several fault-tolerant architectures dedicated to different computer architectures have been proposed to deal with reliability issues in logic part of ICs. To give a representative sample on each class of existing solutions, the following structures are discussed in the next subsections: PaS [NAE2008], Razor [ERN2003], STEM [AVI2012], CPipe [SUB2008], Partial-TMR, Full-TMR [LYO1962] and DARA-TMR [YAO2009, YAO2012].

II.3.A. Pair-and-A-Spare

Pair-and-A-Spare (PaS) uses a multiple duplication with comparison scheme. Having more than one copy allows to have at least one standby-spare copy while two others are active. Initially introduced in [JOH1989], the redundancy and fault detection principles of PaS are illustrated in Figure II.4. In this scheme, at least three copies are connected with their own Fault-Detection (FD) module that detects hardware mismatches within the corresponding module. The output comparator detects any mismatch between two active modules. Whenever a disagreement is detected, the switch selects a new active module based on the reports sent by each FD module. The FD module reporting a mismatch indicates that its corresponding module copy is faulty. The switch then deactivates the faulty module and the standby-spare module becomes active [DUB2013]. This structure is only capable to detect when the hardware becomes permanently faulty and lacks of protection against transient faults. Also, the logic that identifies the faulty modules entails a large area overhead.

![Figure II.4: Principle of PaS redundancy](image-url)
II.3.B. Razor

Razor was first introduced in [ERN2003] and is now well-known as a timing error resilient structure. It uses a technique called timing speculation. As shown in Figure II.5.a, the main idea of this architecture is to use a Razor flip-flop that double-samples pipeline stage values. The first sample is done with a fast clock and the second one with a time-borrowing delayed clock (\(clk_{-del}\)). A metastability-tolerant comparator then checks the values sampled with the fast clock. In case of a timing error, the shadow latches driven by the second clock will activate the recovery mechanism to restore the correct program state. The error is propagated through an OR tree to inform the system that an error correction needs to be carried out by either clock-gating or by rollback recovery. To optimize the energy versus error rate trade-off, Razor uses Dynamic Voltage Scaling (DVS). This method allows the scheme to tune the supply voltage by monitoring the error rate during circuit operation, thereby eliminating the need for voltage margins and exploiting the data dependence of circuit delay.

Razor adds an additional area and power cost due to the sequential logic redundancy. The \(CL\) cost remains the same. If no transient fault is detected, the additional delay added by the control logic is very low.

![a) RAZOR Flip Flop](image)
II.3.C. STEM

Soft and Timing Error Mitigation (STEM) uses the same approach as Razor in the sense that it deals with transient faults. It is shown in Figure II.6. STEM cell architecture was first introduced in [AVI2012]. It employs power saving and Dynamic Frequency Scaling (DFS) mechanisms to operate circuits beyond their worst-case limits. Similarly to Razor that uses shadow latches and a delayed clock, STEM executes triple sampling using two delayed clocks. Whenever the comparator detects a mismatch, the resulting error signal indicates which sample is the most likely to be corrected for a rollback.

The triplication of the sequential logic and its control logic in STEM generates an additional area and power cost of more than 200% for the sequential logic. No
additional cost is associated to the CL. The control logic adds a small delay in the pipeline.

II.3.D. CPipe

The Conjoined Pipeline (CPipe) architecture introduced in [SUB2008] is capable to detect and recover transient and timing errors using spatial and temporal redundancies. In this structure, the CL copy and its flip-flops are interlinked with their original copies through two pipelines as illustrated in Figure II.7. While the leading pipeline is overclocked for faster execution, the duplicated shadow pipeline is sufficiently delayed to detect timing errors. The comparators placed across the leading pipeline register detect any metastable state and eventual SETs reaching the registers during the latching window. It takes three cycles to complete the error recovery rollback by stalling the pipelines and using data from the shadow pipeline registers.

![Figure II.7 CPipe Architecture [SUB2008]](image)

The CPipe architecture does not add a significant delay to the signal propagation but the area and power costs are increased by more than 100% since the CL and the flip-flops are fully duplicated and synchronized by additional control logic.

II.3.E. TMR

Triple Modular Redundancy (TMR) is one of the most popular fault-tolerant architecture. Its first application to computing systems can be found in [LYO1962]. TMR exists in different versions, the simplest one being the Partial-TMR illustrated in Figure II.8a. Three exact copies of the CL are linked to a voting circuit. This means that a majority vote excludes any output that differs from the other two. By doing so, this configuration can mask any single fault occurring in the CL. However, any fault located in the input or output register causes a system failure. To address this issue, the Full-
TMR shown in Figure 2.8b triplicates the entire circuit—flip-flops included—and masks any single fault in the circuit with the exception of the voter and the data input pipeline. A fault occurring in any unprotected part of the circuit would result in a common-mode failure.

![Figure 2.8 Triple Modular Redundancy](image)

The costs associated to the TMR structure are considerably high. Since the CL is triplicated, the partial-TMR area and power cost are of at least 200%. The majority voter must be added to that cost. The additional timing cost to take into account is the delay added by the majority voter. The Full-TMR incurs the same delay but since the sequential logic is included in the triple redundancy, the area and power cost are fully tripled with the addition of the majority voter.
II.3.F. DARA-TMR

The Dynamic Adaptive Redundant Architecture TMR (DARA-TMR) has three complete pipelines but aims at saving power consumption by activating only two pipeline copies in faultless conditions. The architecture operates as a Dual Modular Redundancy (DMR) process. The third pipeline is only activated by power gating when a diagnosis is needed after the error frequency reported by a comparator has reached a threshold. Figure II.9 shows a simplified illustration of a DARA-TMR scheme. When all three pipelines are active, the double comparison allows the identification of the defective pipeline, which is set in off-mode. Then, the system returns to the DMR mode. The error recovery mechanism makes use of architectural components like those used in case of branch prediction errors. DARA-TMR treats permanent fault occurrence as a very rare phenomenon and undergoes a lengthy reconfiguration mechanism to isolate them [YAO2012].

The DARA-TMR has a very similar area cost when compared to a Full-TMR. If no error is detected, the timing is shorter since the comparison takes place during the same time window than the CL signal propagation. However, since only two out of three pipelines are active at the same time, the power cost only doubles.

Figure II.9 DARA-TMR [YAO2012]

II.3.G. Architecture Comparison

All the architectures discussed in previous subsections are reported in Table II.1 (column 2) to give a comparison. Column 1 states if the architecture has a partial fault tolerance or fully tolerates all types of faults. Columns 3 to 5 detail which of the three types of faults is detected and corrected by the architecture. Columns 6 to 9 specify
which components are replicated, and the area, power and timing overhead of each architecture.

Table II.1 SUMMARY OF COMPARISON OF DIFFERENT RELATED FAULT- TOLERANT ARCHITECTURES

<table>
<thead>
<tr>
<th></th>
<th>Partial Fault Tolerance</th>
<th>Transient Fault Tolerance</th>
<th>Timing Fault Tolerance</th>
<th>Hardware Redundancy</th>
<th>Area Overhead</th>
<th>Power Overhead</th>
<th>Error Recovery Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARTIAL</td>
<td>PaS</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>CL×3</td>
<td>No data in [JOH1989]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RAZOR</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>FF×2</td>
<td>1%-3%</td>
<td>3.1%</td>
</tr>
<tr>
<td></td>
<td>STEM</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>FF×3</td>
<td>14%-15%</td>
<td>No data in [AVI2012]</td>
</tr>
<tr>
<td></td>
<td>CPIPE</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>CL×2, FF×2</td>
<td>No data in [SUB2008]</td>
<td>1 cycle</td>
</tr>
<tr>
<td>FULL</td>
<td>Partial-TMR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>CL×3</td>
<td>155%</td>
<td>173%</td>
</tr>
<tr>
<td></td>
<td>Full-TMR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>CL×3, FF×3</td>
<td>207%</td>
<td>206%</td>
</tr>
<tr>
<td></td>
<td>DARA-TMR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>CL×3, FF×3</td>
<td>No data in [YAO2010 or YAO2012]</td>
<td>0 cycles</td>
</tr>
</tbody>
</table>

Architectures that provide protection against permanent, transient and timing faults are considered as full protection solution whereas partial protection architectures only provide fault tolerance for a subset of fault types. Partial-TMR, Full-TMR and DARA-TMR architectures fall in the first category. On the other hand, PaS architecture that only offers permanent faults protection, Razor architecture that only tolerates timing faults, and STEM and CPIPE architectures that deal with transient and timing faults, all falls under the second category. Both full and partial categories include architectures that require at least one full replicate of the component that the architecture protects or require a second full computation to mask or correct a fault.

II.4. Selective Hardening Approaches

As seen in the previous section, all hardening methods rely on some variants of redundancy. This redundancy and the error recovery mechanisms demand considerable resources to tolerate faults [KOR2010]. Moreover, these fault-tolerant solutions have most of the time a limited application range (e.g. ECC are primarily adapted for memory circuits). The most effective approach to deal with a wide spectrum of failures often leads to a massive hardware redundancy. Each redundant structure implies at least more than 100% area overhead. When used, this structural redundancy also implies a similar power overhead. TMR is an example of a structure that covers a wide range of faults. The triplication of the CL – and sometimes including the flip-flops –and the additional cost of the majority voter entails induces an area and power overhead of more than 200% [FAZ2009].
Researchers addressed this concern by proposing a selective hardening approach. The idea of selective hardening is straightforward. If hardening the whole circuit is too resource demanding, then only some chosen parts of the circuit are hardened. The choice of which parts need hardening depends on two main factors, i.e. their particular exposure to failure and their criticality for correct system functioning. Even if the fault coverage decreases due to selective hardening, the overall error rate is still greatly reduced and the redundancy costs are reduced. Selective hardening is then a reliability versus resources trade-off in which the designer tries to reduce costs without penalizing too much the correct functioning of the system.

At first glance, the first steps of the selective hardening process are to improve the vulnerability analysis methodology and use fault-tolerant architectures for hardening. The circuit element vulnerability estimation usually considers the three masking effect discussed in sub-section I.4.C. These effects prevent a fault from being latched in flip-flops. Unfortunately, the huge computation effort required to simulate models with all three masking effects, makes this estimation impractical. Thereby, some techniques rely on approximate abstract models like [FAZ2011]. [MOH2003] and [BOT2015] consider all three masking effects while other like [PAG2012], [POL2008], [MAN2010] and [ZOE2008] resort to only one or two of them to identify which circuit elements are more prone to suffer from and propagate a higher soft error rate.

In this context, Pagliarini et al. introduced in [PAG2012] a cost-aware methodology for selective hardening of combinational logic cells. This is based on Simultaneous Parameter Retrieval Algorithm (SPRA) algorithm and only calculates logical masking. This methodology is capable to perform an automatic trade-off between reliability improvements and associated costs. The algorithm provides a ranked list of the most effective candidates for hardening.

The work by Polian et al. in [POL2008] also uses error probability of circuit elements to estimate their contribution to soft error rate. However, unlike [PAG2012], instead of observing each logical gate and calculate its error propagation probability, it expresses the circuit error rate according to the observability of its outputs. It assumes that the error rate is a conditional probability for an error to be observed on the outputs. The error observability on the outputs is evaluated in the same way as a stuck-at-fault on the output of the evaluated gate would be. Advantages of this method are the very negligible runtime even for massive industrial gate integration. Additionally, this method allows the possibility to incorporate electrical and latching window masking approaches.
Maniatakos et al. in [MAN2010] focus on identifying the most vulnerable parts of a microprocessor for hardening. They also use the logical masking effect for hardening but they include a workload specific to the tested processor. In [BOT2015], Bottoni et al. propose a vulnerability analysis based on a workload specific fault injection method that considers logical and latching-window masking. This method requires a high computational effort due to the simulation-based fault injection.

Finally, some techniques consider all logical, latching-window and electrical masking together. This is the case of Fazeli et al. work in [FAZ2011] in which all three masking effects are included in their probabilistic models to identify fault-exposed spots. In order to alleviate the simulation efforts, Mohanram et al. in [MOH2003] propose an innovative heuristic to meet the trade-off between reliability and cost for the proposed partial duplication architecture.

Table II. 2  SUMMARY OF SELECTED WORKS IN THE AREA OF SELECTIVE HARDENING

<table>
<thead>
<tr>
<th></th>
<th>Masking effects considered</th>
<th>Circuit element of interest</th>
<th>Targeted faults</th>
<th>Fault tolerance technique employed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Logical</td>
<td>Latching-window</td>
<td>Electrical</td>
<td></td>
</tr>
<tr>
<td>Pagliarini et al. [72]</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>Standard cells</td>
</tr>
<tr>
<td>Polian et al. [76]</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>Gates</td>
</tr>
<tr>
<td>Maniatakos et al. [56]</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>FFs</td>
</tr>
<tr>
<td>Bottoni et al. [14]</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>FFs</td>
</tr>
<tr>
<td>Fazeli et al. [30]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>FFs</td>
</tr>
<tr>
<td>Mohanram et al. [65]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Gates</td>
</tr>
</tbody>
</table>

* Multiple Transient Faults  ** Multiple Event Transient

Table II.2 summarizes the selective hardening works discussed in this subsection. Each work (column 1) considers some masking effects (column 2 - 4), some circuit elements to be harden (column 5), the targeted faults (column 6) and eventually the used fault-tolerance technique (column 7).
II.5. Towards Approximate Computing based Fault-Tolerant Architectures

Approximate Computing (AxC) is a paradigm that can be used to deal with the efficiency and cost dilemma. A straightforward definition of AxC would be the relaxation of computational constraints like implementation, storage and/or result accuracy for performance or energy gains. AxC takes advantage of the gap between the accuracy required by an application or user and that delivered by the computing system [MIT2016]. However, and alike Selective Hardening, such relaxation must be judicious in order to preserve quality loss below a certain threshold. AxC has been used for resilient applications, e.g. speech recognition, image encoding, etc., where an approximate result is sufficient for their purpose [SAN2012]. From the hardware standpoint, AxC enables the creation of circuits whose output values may differ from the original circuit for a certain set of input values [MIT2016].

AxC is able to target different layers of computing systems, from hardware to software [XU2016]. In this thesis, the focus is set on Approximate Integrated Circuits (AxICs), which are the outcome of AxC application at hardware level, specifically on ICs. Some authors have developed different strategies to create approximate combinational hardware circuits. These strategies can be grouped into the three main approaches:

- **Ad-Hoc approximate circuits**, which usually involves a different handling for each approximation case. The Ad-Hoc approach is a necessary choice in the case the designer needs to implement or remove specific functionalities to the original circuit. For example, authors in [KAH2012] and [KUL2011] propose an accuracy-configurable adder and multiplier, respectively, to reduce power consumption if the application is resilient. Although they can be efficient, Ad-Hoc approaches are usually very resource-demanding when applied to large circuits.

- **Automatic approximate circuit synthesis methodologies**, which assist the designer in reducing the area of a circuit while minimizing the impact on the accuracy. Authors in [RAH2015] present an algorithm created to design general inexact circuits able to achieve a certain Quality of Resilience (QoR). A quality function determines if the circuit meets the QoR requirements. In [MRA2018], authors developed an evolutionary technique based on genetic codes to approximate circuits until they reach a state in which they are considered too far from their original circuit. Larger circuits can benefit from these synthesis methodologies.
- Hardware neural accelerators to implement approximate functions. Neural Networks (NNs) offer a significant parallelism capability and can be efficiently accelerated by dedicated hardware to gain in performance/energy at the expense of accuracy. For example, in [ELD2014], authors propose NN-based accelerators to approximate Transcendental Functions (i.e. cos, exp, log, pow, and sin).

In previous works presented in [WAL2016-2], the author proposed a very fast and low computational effort method that helps selecting the most sensitive parts of a logic design and identify the degree of hardening necessary to fulfill the design cost (in terms of area and power) and soft error reliability constraints [WAL2015]. Based on this very fast reliability analysis, called structural susceptibility analysis, he also proposed a selective hardening technique using the Hybrid Transient Fault-Tolerant (HyTFT) architecture [WAL2017]. By reducing the number of output nodes of the CL and comparing it with a full version of the circuit, this selective hardening approach not only reduces the size of the comparator but also significantly reduces the size of the duplicated CL copy in a vulnerability-aware manner. The use of the structural susceptibility analysis employed in the HyTFT architecture has proven to be more efficient in terms of area and power consumption with respect to a full duplication scheme. However, this analysis does not consider any error metrics like AxC evaluations usually do (e.g. Error Probability for error rate or Worst-Case Error for error magnitude).

AxC has already been used in the literature in the context of fault tolerance architecture. In [GOM2015] and [SAN2016], the authors presented the Approximate TMR (ATMR) and its extension as Full ATMR (FATMR). Like for the TMR, the ATMR scheme has three CL copies, two AxICs and a precise one. The FATMR goes deeper into AxC designing and uses three AxICs. In these implementations, only one AxIC delivers an erroneous response per input vector. The idea is that each approximate module has its own unique domain of approximation. Since the structure always delivers at least two correct outputs, it can mask any approximate responses coming from one of the AxICs. However, in case of a fault, the structure can only protect the circuit for a set of input vectors defined by the designer. Authors in [ALM2017] show the interest of AxC for fault tolerance in arithmetic circuits. They proposed a configurable-accuracy approximated adder embedding a correction technique. Although effective, this solution is workload dependent.

In this context, the emergence of architectures that bring a partial protection based on AxC has begun and it is important to make sure they are a good alternative to the classical partial protection architectures and what challenges they raise. Moreover, despite the imprecise nature of circuits based on AxC, is it possible to employ AxICs technics to achieve full protection architectures?
II.6. Evaluation

Evaluation methods guaranty the quality of a device by testing its robustness. ANSI and IEEE define robustness assessment testing as the degree to which a system or a component can run correctly in stressful environment conditions or in the presence of an invalid workload [ANSI1991]. The methods to assess these design merits can be divided in two groups: explicit methods and empirical methods.

II.6.A. Explicit methods

- Analytical assessment of robustness methods

They require specifying the behavior of the design with its potential faults according to the environment using logical or mathematical modelling. In addition, it is necessary to determine how well the fault tolerant mechanism works by producing analytic solution of the models [ARL2011]. The downside is the extreme difficulty to realize an exhaustive model of a very large and complex system in practice. Usually, assumptions are made to simplify the model that can greatly reduce the accuracy and usefulness of the results.

- Simulation-Based

Simulations offer a reasonable alternative to the analytic methods of robustness assessment. They give a good trade-off between computation time and modelisation. Simulations require setting up a stochastic model of the fault-tolerant system and the environment. Simulation of the model itself runs for a relevant period of time in which the data gathered is then used to characterize the model’s behavior. The fault occurrences can be forced in order to rapidly obtain an analysis of the system in terms of performance and reliability. Simulation-based fault injection environments require less time and effort to implement and offer better controllability and observability.

- Verification-Based

Industry requires proper verification techniques to provide the correctness of hardware designs and identify built-in robustness defects in fault-tolerant architectures for digital circuits. The first formal use of automated reasoning to check the fault tolerance of digital circuit was published in 1986, where Petri nets were used to verify the fault tolerance of a processor architecture in [CHI1986]. The definition this work gives about formal verification is “A vehicle for hierarchically structuring the verification process so that only few claims need to be proven and only a controllable amount of critical assumptions need to be generated”. In [FEY2011], Fey et al. gave an update to the interpretation and use formal verification in robustness analysis of digital circuits. This work discusses the dilemma of large state space and longer
observation time needed in simulation based approaches for robustness assessment with the use of formal techniques such as Boolean SATisfiability (SAT)-based bounded sequential equivalence checking.

These contributions and several others demonstrated that formal approaches to fault tolerance are rather exhaustive with respect to the complete input space compared to simulation-based approaches. However, these methods are not easily scalable with increasing complexity of digital electronics systems and suffer from runtime limitations [ARL2011].

II.6.B. Empirical Methods

- Field Experience based

Field experience based robustness methods rely on data collected from field to assess the robustness of designs. Making an expert judgment about reliability generally requires a long history of field data.

- Fault Injection based

Complex fault-tolerant systems usually give a hard time to analysis and field-based robustness assessment methods whose accuracy and applicability are considerably restricted. However, fault injection is a particularly attractive candidate and viable solution for such systems [KOO2014]. As mentioned in simulation-based methods, simulation-based fault injection environments require less time and effort to implement and offer better controllability and observability.

II.6.C. Fault Injection at Gate-level

Gate-level simulation provides a suitable model to perform fault-injection experiments. The high fidelity to model most of the physical defects and transient faults is an advantage that micro architectural-level simulation cannot offer. Also, it is much faster than transistor-level simulation. The automated fault-injection flow shown in Figure II.10 is divided in three parts described below.
Fault list generation

To generate a fault list, a parsing script extracts and exhaustively lists the site of each fault from Standard Delay Format (SDF) file or Value Change Dump (VCD) file according to the type of injected faults. Timing fault sites are extracted from the SDF file while transient and permanent faults are extracted from VCD file. The fact that VCD and SDF files are good gate-level representations of fault location and are easily interpreted by the gate-level logic simulator used for fault injection make them a perfect candidate for the task. Once the fault sites are listed, another script randomizes a selection of fault sites offering the possibility to select or exclude the fault sites located in specific design modules. Moreover, each random fault site is associated with a constrained-random timing value. The input parameters used to control the fault injection are listed and described in Table II.3.

Table II.3  FAULT INJECTION PARAMETERS [WAL2016-2]

<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault model</td>
<td>Specify the fault model to generate either permanent (stuck-at), transient (temporary stuck-at) or timing (interconnect delay) fault.</td>
</tr>
<tr>
<td>No. of injections</td>
<td>Set a number of fault site to be randomly listed in the fault list.</td>
</tr>
<tr>
<td>Injection time range</td>
<td>Specifies the range to constrain the random injection time generation process. Mainly used to ensure that none of the faults are injected during circuit initialization nor at the time too close to the end of the simulation.</td>
</tr>
<tr>
<td>Injection duration range</td>
<td>Used only for transient faults to specify their pulse width randomly between two defined values.</td>
</tr>
<tr>
<td>Timing error range</td>
<td>Specifies the range of additional random delay values to be used for timing errors.</td>
</tr>
<tr>
<td>Fault Location constraint</td>
<td>Constrains the fault list generation process to produce a faults list according to specified module(s).</td>
</tr>
<tr>
<td>Simulation duration</td>
<td>Gives the time of each fault simulation.</td>
</tr>
<tr>
<td>Injection type</td>
<td>Indicates whether single of multiple fault injections per simulation</td>
</tr>
</tbody>
</table>
- **Fault injection**

During the actual fault injection part of the flow, a script runs gate-level simulations in logic simulator and injects the faults. The time and location of such injected faults are specified by the fault list thanks to specific commands allowed by the simulator such as stopping the simulation or even modifying values of specified signals. A test bench produces the data necessary to generate logs for later analysis. The logs can store cycle-by-cycle information or only the final result depending the type of under test design. In addition, the test bench monitors which signal has an unexpected value and reports it together with the normally expected value. The *SDF* modifier script injects timing faults in the *SDF* file to be used in timing faults simulations. The fault injection campaign script generates a fault injection log file that can be interpreted in retrospect.

Gate-level simulation offers the possibility to inject faults according to the desired model being stuck-at fault for permanent faults, temporary stuck-at faults for *SETs* or interconnect delay faults for timing faults. The fault models enounced above and their fault injection mechanism are discussed in the subsections below:

- **Permanent fault injection**: The use of the standard stuck-at fault model perfectly represents permanent fault behavior. However, the stuck-at are not imposed at 0 or 1 on the circuit nodes. Instead, the logic state of the fault injected node is forced to its opposite logic value at the injection time. This fault injection model allows the simulation of several different permanent faults without altering the golden design. A permanent fault is defined by fault location (*l*) and fault injection time (*t*). Figure II.11 shows a permanent fault injection campaign that consist in a fault simulation amount defined in the fault list generation step. Each simulation starts and carries on until the time *t* is reached. At this time, the simulation is stopped and the logic value of the injection site is flipped. Then, the simulation is resumed until the end of the workload. Once the simulation is over, the information is stored in the fault injection log file.
Figure II.11 Permanent Fault Injection Campaign

- **Transient fault injection**: Transient faults are modelled as temporary stuck-at faults. This allows to represent *SETs* as a pulse width defined by three parameters: fault location \( l \), fault injection time \( t \) and pulse duration \( d \). As shown in Figure II.12, Transient faults are injected as permanent faults with the exception that the flipped value on injection location is flipped back at the end of the pulse duration \( d \). As for permanent faults, once the simulation is over, the information is stored in the fault injection log file.

Figure II.12 Transient Fault Injection Campaign

- **Timing fault injection**: This fault injection requires the modification of the *SDF* file in which the interconnect delays are adjusted to simulate timing faults. Synthesis tool generates the original *SDF* file with interconnect delays of duration \( l \) to which an amount \( \Delta t \) is added. Figure II.13 shows how at the beginning of each simulation, the original *SDF* file is modified based on \( l \) and \( \Delta t \). This modified *SDF* file is used for
gate-level simulation instead of the original one. As for permanent and transient faults, once the simulation is over, the information is stored in the fault injection log file.

Figure II.13  Timing Fault Injection Campaign

- **Fault classification**

  The fault injection log file gives the possibility to make an analysis of the injected faults and classify them into the five possible following outcomes:

  - **Silent Faults**: these are the faults that entail no effect on the execution of the workload that ends normally with no error detection. The computation result is correct and the data stored in registers and other memory elements are the same as those of a fault-free run.
  - **Latent Faults**: these faults are considered as latent when the workload ends normally with the exception of corrupted content of the pipeline registers, register-file or other memory elements. In this scenario, there is no error detection and the erroneous data will be used in later computations. Latent faults are considered critical as the erroneous data will potentially lead to wrong computations.
  - **Fail-silent Faults**: The workload terminates normally with no error detection and the result computed is wrong. These faults are the most critical as the result computed are wrong without any error indication.
  - **Corrected Faults**: The workload terminates normally with at least one error detected. The result is correct and the content of pipeline registers and register-file are the same as those of a fault-free run.
  - **Unclassifiable**: Some injected faults result in setup or hold violations and cause unknown logic value X to propagate. In real devices, hold violations may cause faulty value that may be stored in memory elements. This anomaly
can eventually be detected by a detection mechanism. Also, X value propagations are due to gate-level simulation restraints. In test cases involving a real device, these faults will actually result in the silent or corrected fault categories and thus, non-critical from the robustness point of view. Since gate-level simulation cannot make the distinction among them, they are considered as unclassifiable.

In this thesis, latent faults are considered as fail-silent faults for practical reason since both are critical in terms of fault-tolerant capability and both lead to the same outcome. Critical faults are the ones that escape the detection and lead to a failure. The ratio of these critical faults with respect to the total number of injected faults gives an efficiency performance to compare the fault-tolerant capability of different architectures. The detected faults in section III.4 have the same characteristics as corrected faults with the difference that there is no recovery mechanism once the fault is detected.
Conclusion

The answer to faults occurring during normal activity are usually fault-tolerant designs. Whether faults take their origin from a particle strike, interferences, design variations or wear-out, they can be detected, corrected or masked with corresponding design technics. This chapter reviewed an exhaustive list of the different approaches existing in the literature, discussed their mechanisms to deal with faults and gave an insight of their drawbacks in terms of costs. To ensure that a fault-tolerant design satisfies the required level of reliability, it is possible to use evaluation methods. Fault injection techniques are very useful to speed up fault tolerant design testing by modifying specific values at gate level so the behavior of the design can be observed.
CHAPTER III
SELECTIVE HARDENING BASED ON APPROXIMATE DUPLICATION

The selective hardening philosophy aims at minimizing the cost entailed by the fault-tolerant architecture while trying to minimize the reliability loss that comes with cost reductions. As previously mentioned in section II.5, AxC have a similar reasoning where the goal is to minimize the logic area cost at the expense of precision in the computation. This analogy awakes the interest in employing AxIC as a redundant module for a duplication and comparison scheme and observe its performance in the reliability cost trade-off. Most AxICs conceived are arithmetic circuits because their precision loss is easily measurable and it doesn't require to know the exact workload.

In this chapter, we analyze the impact of the selective hardening technique introduced in section II.4 and proposed in [WAL2017] by comparing different duplication techniques implemented in an error detection architecture suitable for arithmetic circuits. We explore four different scenarios of duplication i) a full duplication scheme, ii) a reduced duplication scheme based on the structural susceptibility analysis presented in [WAL2015], iii) a reduced duplication scheme based on the logical weights of the arithmetic circuit outputs and iv) a reduced duplication scheme based on an approximated structure from a public benchmark suite [MRA2017] which is composed of arithmetic circuits. Note that, all the considered scenarios are built independently of the workload. Experimental results achieved on adders and multipliers demonstrate the interest of using approximate structures as duplication scheme since both area overhead and power consumption are reduced compared to a full duplication scheme, while maintaining good levels on error metrics.

The study aims at highlighting the fact that approximate structures used for duplication offer interesting perspectives to build error detection schemes. In the proposed study, experiments have been done as fairly as possible, with faults injected in the combinational blocks only, thus assuming fault-free voters. Note that the arithmetic circuits used as case studies (8-bits adders and 8 to 16-bits multipliers) in our experiments are relatively small compared to the required comparator needed to build the duplication scheme. Consequently, considering area and power overhead of comparators would negatively affect the reliability comparisons between the four considered scenarios. For this reason, all experiments have been done without considering the area and power overhead due to the comparators. This may slightly biased the results from a quantitative point of view, but it does not jeopardize the main
conclusion about the interest of using approximate structures as duplication scheme. Moreover, to corroborate the experimental results, we run a set of simulation-based gate-level transient fault injections. They show that using approximate structures as duplication scheme offers a better reliability level compared to the other considered duplication scenarios.

In this chapter, we first review the structural susceptibility analysis in subsection III.1 which is one of the three scenarios detailed in subsection III.2. Then, we go through the scenarios comparison and discussion over various metrics in subsection III.3. In subsection III.4, we validate the analytical results with fault injection results. Finally, a summary gives a closure on the AxC-based hardening selective performances.
III.1. Structural Susceptibility Analysis

The structural susceptibility analysis methodology proposed in [WAL2015] is based on the fact that not all outputs of a CL block have the same susceptibility to SET (Single Event Transient) effects and assumes that their susceptibility is a function of the number of nodes in their fan-in logic cone. It exploits the structural properties of the output fan-in cone to get their relative susceptibility estimates. The outputs are ranked on the basis of their relative susceptibility and the best candidates are selected for error detection.

Algorithm III.1 shows the pseudo-code of the susceptibility analysis. The algorithm starts by reading the pre-place-and-route netlist of the design. Then it forms groups $F_j$ of all fan-in cells for each CL output $S_j$. Once the groups are formed, the weight $W_j$ of each fan-in cone is calculated by adding the weights of all the cells in the corresponding fan-in cone group. According to the assumption that forms the basis of this method, the cell weight is the number of inputs and outputs of that cell. Ranks are assigned to each output on the basis of their fan-in cone weight using a sort function shown in line 15 of Algorithm III.1.

```
1 read(netlist);
2 // Group all fan-in cone cells together for each output node
3 foreach $O_j$ do
4     $F_j$ ← $O_j$.get_fanin();
5 end
6 // Get weight of fanin cone of each output
7 foreach $O_j$ do
8     foreach $C_i$ do
9         if $C_i$ ∈ $F_j$ then
10            $W_j$ ← $W_j + C_i$.get_pim();
11        end
12     end
13 // Sort outputs on the basis of their fanin cone weight
14 sort($O_j,F_j,W_j$);
```

Algorithm III.1 Structural susceptibility analysis

The algorithm is further explained by its application to a simple example circuit shown in Figure III.1. The shaded regions mark the boundaries of the two output fan-in cones. On the top of each gate, $W_i$ indicates their respective weight. The addition of all the $W_i$ gives the preliminary fan-in cone weight ($S_j$). In this example, 14 and 12 are the respective fan-in cone weights of $O1$ and $O2$. According to these figures, output $O1$ is more susceptible to SETs than output $O2$. In other words, having a SET detection mechanism placed on $O1$ can better improve the reliability of the circuit when compared to having it placed on $O2$. With this acknowledged, we consider that cells within several output fan-in cones only belong to the most weighted one. Until all the
overlaps are erased, the ranking procedure is reiterated and, each time, we recalculate the weight of the output fan-in cones.

Figure III.1 Application of the structural susceptibility analysis

In order to prove the effectiveness of the proposed structural susceptibility analysis, we have compared it with fault injection experiments. Figure III.2 gives these comparisons for the b03 circuit from the ITC’99 benchmark suite. The red colored plot represents the normalized distribution of the output fan-in cone weight \( S_j \) (i.e. the structural susceptibility) and the blue plot represents the distribution of the average number of soft error failures observed on the output nodes of the circuit during a fault injection campaign. Deeper analyses and validations of this structural susceptibility analysis can be found in [WAL2015].

Figure III.2 Output susceptibility analysis results on b03

III.2. Selective Error Detection Architecture for Arithmetic Circuits

An error detection architecture must be capable of detecting transient, permanent and timing faults that may occur in an arithmetic circuit. The error detection scheme we evaluate employs duplication and comparison to detect the occurrence of faults.
Since the architecture relies on duplication of the arithmetic block and the use of a comparator, its implementation incurs an overhead of more than 100% in terms of area and power.

A practicable way of providing the designer the freedom to control the area/power overhead and the reliability improvement of an error detection architecture implementation is to cleverly select the functions to be duplicated. Figure III.3 shows a simplified scheme of the considered error detection architecture. It can be seen that the Reduced Copy Block (RCB) only implements a part of the arithmetic functions of the original Arithmetic Block (AB). A comparator, represented by the block labeled as ‘==?’, generates an Error flag signal when its inputs are different, thus allowing the fault detection. Moreover, the comparator must be adapted to the various duplication scenarios.

Next sub-sections address the different duplication scenarios we have considered:

A: a full duplication scheme;

B: a reduced duplication scheme based on the susceptibility analysis;

C: a reduced duplication scheme based on the logical weight of the arithmetic circuit outputs;

D: a reduced duplication scheme based on an approximated structure.

### III.2.A. Scenario 1 (S1) – Full duplication scheme

This scenario represents the ideal case of the error detection architecture. In fact, when full duplication is used, the error detection architecture is able to detect all faults (transient, permanent and timing faults) that may occur in the arithmetic circuit. For this scenario, the comparator is a full comparator able to produce an error signal when it receives different binary values on its inputs.
III.2.B. Scenario 2 (S2) – Reduced duplication scheme based on the structural susceptibility analysis

Here, we use the structural susceptibility analysis to build a number of reduced copies of an arithmetic circuit. As illustrated in Figure III.4, each copy is created by selecting a set of outputs ranked by descending order of their weight $S_j$ obtained by Algorithm III.1. Consequently, the smallest copy corresponds to the logic cone driving the output having the highest weight $S_j$. Conversely, the biggest copy corresponds to a copy of the circuit truncated from its logic cone driving the output having the lowest weight $S_j$. For this scenario, the comparator is reduced compared to a full comparator since $RCB$ has fewer outputs to compare to the original $AB$.

The use of $S2$ to build the duplication scheme leads to an error detection architecture able to detect only faults affecting the common (structural/functional) area between $AB$ and $RCB$ generated by $S2$. Hence, a set of faults, with a size depending on the duplication ratio, will be not detected by these duplication schemes. These faults will affect the function of the arithmetic circuit by providing wrong answers. Consequently, we must formalize the impact of the undetected faults on the application in order to determine if the outputs are still acceptable or not by the user. This characterization is usually done in the AxC context.

The AxC paradigm is based on the intuitive observation that rather than a perfect result, inner operations of a computing system can be selectively inaccurate for providing gains in efficiency (i.e., less power consumption, less area, higher manufacturing yield) [WAL2017, MRA2017 and TRA2018]. An AxC structure is generally qualified by error metrics. In this chapter, we use the Error Probability ($EP$) and the Worst-Case Error ($WCE$) metrics to evaluate the impact of the different duplication scenarios on the correctness of the arithmetic outputs. Those metrics are commonly used in the AxC field. For $S2$, the $WCE$ is defined as the largest arithmetic difference between $AB$ and $RCB$ (Equation III.1).

$$WCE = |AB_{\max} - RCB_{\max}| = \sum_{i \in B} 2^i, \quad 0 \leq WCE \leq 2^n - 1$$ (III.1)

where $B$ (with $0 \leq B \leq n - 1$) indicates the position of the outputs in $AB$ that are truncated in $RCB$ and $n$ is the number of outputs of $AB$. 
Reduced combinational blocks versions ranked by the structural susceptibility analysis

The $EP$ error metric (Equation III.2), in case of $S2$, is always 100% since every $RCB$ version has at least less outputs compared to $AB$. Consequently, there is always a difference between the responses of $RCB$ compared to the ones of $AB$.

$$EP = 100\%$$ \hspace{1cm} (III.2)

III.2.C. Scenario 3 ($S3$) – Reduced duplication scheme based on the logical weight

Since the structural susceptibility analysis only considers the circuit structure, here we consider the possibility to duplicate the arithmetic circuit by using a functional metric. Indeed, we consider that the outputs of the arithmetic circuit can be ranked from $LSB$ (Least Significant Bit) to $MSB$ (Most Significant Bit). Note that, $S3$ partial duplication scheme may be considered as an Unequal Error Protection ($UEP$) scheme. As shown in Figure III.5, the idea is to build the reduced copies of the arithmetic circuit based on logic cones driving the $MSB$ down to the $LSB$. In this case, the smallest copy corresponds to the logic cone driving the $MSB$ output while the biggest duplication corresponds to a reduction of the arithmetic circuit truncated from its logic cone driving the $LSB$ output. As for $S2$, the comparator is also reduced since the duplication has fewer outputs.
Figure III.5 Reduced combinational blocks versions ranked by the logical weight analysis

S3 creates reduced copies by truncating output cones from the original circuit in the same way S2 does. In that case, S3 can be characterized using AxC metrics ($WCE$ and $EP$) as well. Hence, for S3 the $WCE$ and $EP$ are calculated using Equations III.1 and III.2 respectively.

### III.2.D. Scenario 4 (S4) – Reduced duplication scheme based on an approximate structure

S4 consists in using as RCB an approximate version from a public benchmark suite [MRA2017] of the arithmetic circuit. The approximate version is selected based on its reduced area and timing properties compared to the original precise version. For this last duplication scenario, the comparator must provide an error signal when the precise arithmetic circuit processes a response for which the difference with the AxC version used as RCB is larger than the selected $WCE$ value. Note that, in the considered benchmark suite we worked with, all approximate circuits have the same number of outputs as the precise arithmetic circuits. More details on the design of such a comparator can be found in [TRA2018]. For this scenario, $WCE$ and $EP$ are defined by Equations III.3 and III.4 respectively. These equations are commonly used in the AxC field.

\[
WCE = \max_i \left| O^{(i)}_{\text{approx}} - O^{(i)}_{\text{prec}} \right|, \quad 0 \leq WCE \leq 2^n - 1 \quad (\text{III.3})
\]

\[
EP = \frac{\# \text{Faulty response}}{2^n} \quad (\text{III.4})
\]

where $O^{(i)}_{\text{approx}} (O^{(i)}_{\text{prec}})$ is the ith output of the approximate (precise) implementation. The number of faulty responses is obtained by running an exhaustive simulation of $AB$ and $RCB$. The number of responses that are different for the two modules is the number of faulty responses.
III.3. Experimental Results

In this section, we aim to demonstrate the interest of using AxC circuits as reduced duplication in error detection scheme. To do that, we have selected simple arithmetic circuits and their corresponding AxC metrics, i.e. \( WCE \) and \( EP \). The extension of this study to general purpose circuits would require the knowledge of i) the workload of the circuit and ii) the set of constraints the designer can relax on the precision. Then, AxC versions could be generated and evaluated with our partial duplication scheme.

III.3.A. Experimental Setup

The four duplication scenarios are compared using four case studies based on precise arithmetic circuits: an 8-bits carry look-ahead adder, an 8-bits carry look-ahead multiplier, a 12-bits array multiplier and a 16-bits array multiplier. Table III.1 provides an overview of the specifications of these four arithmetic circuits in terms of function and number of available AxC versions in the public benchmark suite [MRA2017]. The versions of each kind of arithmetic circuit we selected fulfil two criteria. The first one is to make sure the circuit’s area is equal or lower than that of the precise version. The second one is that the timing of the longest path of the AxC is equal or shorter than that of the precise version. The last column of Table III.1 gives the number of AxC versions we considered in our experiments.

<table>
<thead>
<tr>
<th>Circuit name</th>
<th>Functionality</th>
<th># of AxC versions available</th>
<th># of AxC versions considered</th>
</tr>
</thead>
<tbody>
<tr>
<td>add8</td>
<td>8-bits adder</td>
<td>504</td>
<td>449</td>
</tr>
<tr>
<td>mult8</td>
<td>8-bits multiplier</td>
<td>521</td>
<td>471</td>
</tr>
<tr>
<td>mult12</td>
<td>12-bits multiplier</td>
<td>153</td>
<td>147</td>
</tr>
<tr>
<td>mult16</td>
<td>16-bits multiplier</td>
<td>60</td>
<td>60</td>
</tr>
</tbody>
</table>

To compare the different scenarios (S1, S2, S3 and S4), we present the results achieved in terms of area and power consumption overhead with respect to S1, as well as \( EP \) and \( WCE \) metric values. For S2 and S3, \( WCE \) and \( EP \) are calculated with Equations III.1 and III.2 For S4, \( WCE \) and \( EP \) values are obtained with Equations III.3 and III.4 and the application of an exhaustive workload. All netlists were synthesized using a commercial RTL synthesis tool [SYN] with the NanGate 45nm Open Cell Library [SIL]. Note that, all synthesis runs are in the same magnitude order (i.e., about few seconds). Moreover, we did not observe a significant difference of CPU run time between a precise and a reduced arithmetic circuit, even with the largest considered case study (i.e., a 16 bits multiplier). Power consumption values are estimated values (rather than exact values) given by the commercial synthesis tool. Nonetheless, they are relative
between every single version of each scenario we explored, making so that a fair comparison of the power consumption is finally obtained. Moreover, note that $S_1$ is implicitly shown in every figure where the area overhead is 100% as it corresponds to a full duplication scheme.

**III.3.B. Selective Susceptibility versus Selective Arithmetical Hardening**

Before any comparison that would show how $S_4$ is performing, it is important to get a good look at $S_2$ and $S_3$ performances before to compare $S_4$ to these two scenarios. Figures III.6a, III.6b, III.6c and III.6d present the results (i.e. the area overhead, the power overhead and the WCE values) of $S_2$ with each possible reduced duplication applied on the 8-bits adder and the 8/12/16-bits multipliers respectively. The more outputs and their connected logic are removed, the lower the area overhead.

For example, let us consider the highlighted data shown in Figure III.6a. The resulting area overhead of the duplication scheme for a selected version of the 8-bits adder is 94.89% (i.e. $RCB$ represents 94.89% of the total area of $AB$). This is represented by the dashed vertical line in Figure III.6a. For this duplication ratio, we obtain a power consumption overhead of 92.87% and a $WCE$ of 97. As stated in Equation II, the $EP$ value is 100% for every reduced possible duplication cases.
Figure III.6  Comparison of Scenario 2 with respect to Scenario 1

b) 8-bits multiplier

c) 12-bits multiplier

d) 16-bits multiplier
In the same way, Figures III.7a, III.7b, III.7c and III.7d present the same results achieved when using S3 as duplication scenario.

\[ a) \text{8-bit adder} \]

\[ b) \text{8-bit multiplier} \]

\[ c) \text{12-bit multiplier} \]
As first general comment on these results, we can highlight the fact that when reducing the area of RCB, the power overhead is also reduced while WCE values increase. Moreover, these results show that while the power overhead of S2 and S3 have a similar behavior, the WCE is lower for most of the versions of the duplications using S3. This behavior is explained by the fact that S3 duplicates the arithmetic circuit with functional constraints (the last output fan-in cones to be removed are the arithmetically most significant) while S2 only considers structural constraints. Both, S2 and S3 have the same smaller RCB version as they both share the same remaining output logic cone. This is explained easily enough since for S2, the most susceptible output logic cone is the one with more logic gates. This output logic cone usually corresponds to the one with most arithmetic value as well since it coincides with the carry value. This fact also explains the reason why area overhead values have difficulties to reach very interesting area cost reductions with respect to satisfactory reliability metrics.

III.3.C. Approximate Redundancy Performance

For better visualization and comparison of the previous results with respect to S4, we illustrate power consumption overhead, EP and WCE values of each scenario separately. In the approximate benchmark suite, and for a fair comparison, we selected all the circuits available whose area and timing values do not exceed the values of the circuit used as precise arithmetic block.
• **Power and Area Overheads**

For each scenario, Figures III.8a, III.8b, III.8c and III.8d show the power consumption overhead of the 8-bits adder and the 8/12/16-bits multipliers respectively. Results show that the power overhead in every scenario has the same behavior, i.e. it decreases proportionally with respect to the hardware cost. Smaller circuits (i.e. 8-bits adder and multiplier) have a lower power overhead for S4. For the bigger circuits, the power consumption of approximate circuits is nearly 10% more than for the S2 and S3 scenarios with a few exceptions for the 12-bits multiplier. Even if the power consumption cost is sometimes higher for S4, the wide choice of AxC versions allows the designer to choose one with a lower power consumption.

![Graph showing power overhead vs area overhead for different scenarios](image)

**a) 8-bits adder**

![Graph showing power overhead vs area overhead for different scenarios](image)

**b) 8-bits multiplier**
Figure III.8 Power overhead comparison of all scenarios with respect to Scenario 1

- Error Probability

Figures III.9a, III.9b, III.9c and III.9d show the EP values of the 8-bits adder and the 8/12/16-bits multipliers respectively. The EP values achieved with an AxC version of the arithmetic circuit (S4) are lower than those obtained with the other scenarios. This is true for the considered case studies of arithmetic circuits, but also true for any other case since EP of S2 and S3 is constant at 100% by definition. The EP rates reach most of the time 100% (or close to 100%) when decreasing the area overhead, meaning that the architecture will mostly always have different responses between the precise version and the reduced version. Nevertheless, a high EP level is not always problematic. If a circuit has a high EP only affecting LSB for example, the impact on the
application will be low. On the other hand, we could have a low $EP$ that could impact the $MSB$ and thus have a higher impact on the application. This statement is analyzed with the help of the $WCE$ metric.

![Graph](image-url)

**Area Overhead (%)**

- **S2 EP**
- **S3 EP**
- **S4 EP**

a) 8-bits adder

![Graph](image-url)

**Area Overhead (%)**

- **S2 EP**
- **S3 EP**
- **S4 EP**

b) 8-bits multiplier
Figure III.9  EP comparison of all scenarios with respect to Scenario 1
- **Worst-Case Error**

Figures III.10a, III.10b, III.10c and III.10d show the WCE values of the 8-bits adder and the 8/12/16-bits multipliers respectively. From these results, we can say that S3 performs better (i.e. provides a lower WCE level) than S2, but more importantly, that S4 performs better than S3 for any duplication versions. This is understandable as the main purpose of AxC is to produce outputs as close as possible to the precise circuit outputs but with a reduced cost (i.e. area overhead and power consumption).
III.3.D. Results Summary

These comparisons between the four scenarios, summarized in Table II, show that the use of $AxC$ circuit as reduced copy to build a duplication scheme seems to be a good alternative to build an error detection scheme for arithmetic circuits. In fact, this duplication scenario ($S4$) offers better values in terms of area and power overhead while reducing drastically the error metrics (i.e. $EP$ and $WCE$) compared to the more conventional $S2$ and $S3$ duplication scenarios. $EP$ and $WCE$ ranges were measured exhaustively for $S2$ and $S3$. $EP$ and $WCE$ values for $S4$ were taken from the public benchmark suite [MRA2017].
The discussion on the fact that a high EP could not be problematic as long as the WCE only affects LSB outputs is quite self-explanatory when comparing S2 with S4. While S2 and S4 share a similar EP for the RCB versions with lower area, S2 WCE rapidly reaches its maximum value. It is clear that S4 offers a better trade-off as well as a wider choice of area and power costs.

### Table III.2 Ranges of Metrics (Area, Power, EP and WCE) for each Scenario (S1, S2, S3, and S4)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Any</td>
<td>100%</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>S2</td>
<td>add8</td>
<td>76.14% - 98.30%</td>
<td>71.71% - 98.30%</td>
<td>100%</td>
<td>0%</td>
<td>1 - 255</td>
</tr>
<tr>
<td></td>
<td>mult8</td>
<td>94.21% - 99.61%</td>
<td>90.08% - 100%</td>
<td>100%</td>
<td>0%</td>
<td>1 - 32767</td>
</tr>
<tr>
<td></td>
<td>mult12</td>
<td>97.67% - 99.94%</td>
<td>95.18% - 99.86%</td>
<td>100%</td>
<td>0%</td>
<td>1 - 8e6</td>
</tr>
<tr>
<td></td>
<td>mult16</td>
<td>98.22% - 99.97%</td>
<td>96.21% - 99.90%</td>
<td>100%</td>
<td>0%</td>
<td>65503 - 2e8</td>
</tr>
<tr>
<td>S3</td>
<td>add8</td>
<td>76.14% - 98.30%</td>
<td>71.71% - 98.30%</td>
<td>100%</td>
<td>0%</td>
<td>1 - 255</td>
</tr>
<tr>
<td></td>
<td>mult8</td>
<td>94.21% - 99.61%</td>
<td>93.44% - 100%</td>
<td>100%</td>
<td>0%</td>
<td>1 - 32767</td>
</tr>
<tr>
<td></td>
<td>mult12</td>
<td>97.367% - 99.94%</td>
<td>95.18% - 99.95%</td>
<td>100%</td>
<td>0%</td>
<td>1 - 8e6</td>
</tr>
<tr>
<td></td>
<td>mult16</td>
<td>98.22% - 99.97%</td>
<td>96.21% - 99.97%</td>
<td>100%</td>
<td>0%</td>
<td>1 - 2e6</td>
</tr>
<tr>
<td>S4</td>
<td>add8</td>
<td>69.6% - 99.43%</td>
<td>62.04% - 83.30%</td>
<td>0.4%</td>
<td>3.81%</td>
<td>1 - 168</td>
</tr>
<tr>
<td></td>
<td>mul8</td>
<td>65.74% - 99.81%</td>
<td>60.83% - 96.78%</td>
<td>0.1%</td>
<td>3.90%</td>
<td>1 - 3204</td>
</tr>
<tr>
<td></td>
<td>mult12</td>
<td>50.07% - 99.73%</td>
<td>50.02% - 97.58%</td>
<td>25%</td>
<td>1.53%</td>
<td>1 - 2e6</td>
</tr>
<tr>
<td></td>
<td>mult16</td>
<td>50.51% - 100%</td>
<td>50.27% - 111%</td>
<td>96.9%</td>
<td>0.01%</td>
<td>38804 - 855e6</td>
</tr>
</tbody>
</table>

### III.4. Fault Injection Validation

#### III.4.A. Fault Injection Campaign Setup

In the experimental results detailed in Section III.3, we compared all scenarios in terms of area/power overhead, EP and WCE. These results show that the use of an AxC arithmetic structure as RCB in a duplication/comparison scheme offers a better trade-off than the reduced versions of the original circuit proposed by S2 and S3. In fact, S4 offers better error metric values together with lower area and power overheads. To assess and compare the error-detection capability of the different duplication scenarios, we performed simulation-based gate-level fault-injection experiments using our ad-hoc fully automated fault-injection framework presented in [WAL2016-1]. Gate-level simulation provides a suitable paradigm to perform fault-injection experiments since, unlike micro architectural-level simulation, it faithfully models most of the physical defects and transient faults, and is much faster than transistor-level simulation.
The fault injection campaign consists in injecting single transient faults randomly chosen in the duplication scheme, excluding the comparator structure since any fault inside it cannot be detected. The number of transient faults we injected is defined by the approach proposed by [LEV2009] with 1% margin of error and 95% confidence. With these values, the number of injected faults is large enough to achieve a good distribution between fault categories (i.e., masked faults, detected faults and fail silent faults). We did not choose more rigorous confidence and error percentage values due to the scalability issues. Transient faults are modelled as digital pulses using three parameters: fault location \(l\), fault-injection time \(t\) and duration \(d\) that represents the SET pulse width. Particle induced SET pulse widths vary depending on factors like type of radiations, capacitance of the impacted net and process technology [WIR2008]. We randomly selected pulse widths from a range between 0.25ns and 1.25ns. The selection of this range was made considering the typically anticipated SET pulse widths in 45nm technology. \(l\) and \(t\) were selected randomly. Also, the polarity of the injected pulse was selected to be opposite to the signal state of \(l\) at the time of the fault injection. Note that, we added a delay on the circuit outputs so that the comparator can finish its work before the capture is done. The test bench we applied as workload is a test sequence detecting all stuck-at faults in the duplication scheme, still excluding the comparator structure. The advantage of using such a sequence is that it guarantees that all nets are activated. We use as case studies an 8-bits adder and an 8-bits multiplier to build the different duplication scenarios. Moreover, for a fair comparison between scenarios, we selected comparable duplication ratio in terms of area.

Table III.3 gives details (Area Overhead, Power Overhead, WCE and EP values) on the RCB version used in each scenario as well as the number of faults injected (# Injected Faults) for each injection campaign. One can notice that the different area overhead of S2, S3 and S4 were chosen closest to each other with about 80% for 8-bits adder, 95% for the 8-bits multiplier, 97% for the 12-bits multiplier and 98% for the 16-bits multiplier. As a result, the number of injected faults also remains very similar between the different compared scenarios. Note that, if we selected the best case of each scenario the resulting comparison would not be fair since the area overhead of S4 will be close to 50% with regards to 80% for S2 and S3. In that case, the fault injection analysis would not fairly reflect the interest of S4.
Table III.3  Case Study Specifications

<table>
<thead>
<tr>
<th>Scenario</th>
<th>8-bits adder</th>
<th>8-bits mult</th>
<th>12-bits mult</th>
<th>16-bits mult</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RCB</td>
<td>Area Overhead</td>
<td>Power Overhead</td>
<td>WCE</td>
</tr>
<tr>
<td>S1</td>
<td>Full 8-bits adder</td>
<td>100%</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>Reduced 8-bits adder based on the structural susceptibility analysis</td>
<td>81.25%</td>
<td>78.76%</td>
<td>247</td>
</tr>
<tr>
<td>S3</td>
<td>Reduced 8-bits adder based on the logical weight</td>
<td>80.68%</td>
<td>76.54%</td>
<td>63</td>
</tr>
<tr>
<td>S4</td>
<td>An approximate version of the 8-bits adder: Add8_142</td>
<td>80.68%</td>
<td>79.24%</td>
<td>7</td>
</tr>
<tr>
<td>S1</td>
<td>Full 8-bits multiplier</td>
<td>100%</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>Reduced 8-bits mult based on the structural susceptibility analysis</td>
<td>95.00%</td>
<td>94.53%</td>
<td>28671</td>
</tr>
<tr>
<td>S3</td>
<td>Reduced 8-bits mult based on the logical weight</td>
<td>95.00%</td>
<td>94.53%</td>
<td>8191</td>
</tr>
<tr>
<td>S4</td>
<td>An approximate version of the 8-bits mult: Mul8_268</td>
<td>95.42%</td>
<td>88.68%</td>
<td>2292</td>
</tr>
<tr>
<td>S1</td>
<td>Full 12-bits multiplier</td>
<td>100%</td>
<td>100%</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>Reduced 12-bits mult based on the structural susceptibility analysis</td>
<td>97.67%</td>
<td>95.18%</td>
<td>8e9</td>
</tr>
<tr>
<td>S3</td>
<td>Reduced 12-bits mult based on the logical weight</td>
<td>97.67%</td>
<td>95.18%</td>
<td>8e9</td>
</tr>
<tr>
<td>S4</td>
<td>An approximate version of the 12-bits mult: Mul12_070</td>
<td>95.90%</td>
<td>98.54%</td>
<td>16765</td>
</tr>
</tbody>
</table>

Since the fault injection is done by simulation on a gate-level model of the duplication scheme, we have access to all nets of the circuits and hence we can compare the outputs of $AB$ with their corresponding golden responses. After the fault injection, faults can be classified into three categories as presented in sec:

- **Silent Faults**: With transient fault injection model, such $SET$ are filtered/masked by the circuit. For $S4$ when a failure induces an effect on circuit’s outputs below the $WCE$, it is classified as Silent Fault. In that case, the difference between the original circuit and its approximate version does not exceed the numerical $WCE$ value. On the other hand, $S2$ and $S3$ correspond to duplication scenarios where only non-zero differences are considered as masked-faults. Here, the reduced copy only allows the detection of possible failures affecting the common duplicated logic.

- **Fail-silent Faults**: During workload application, the outputs are corrupted at data capture time —like a memory element would— with no error detection.

- **Detected Faults**: The faults result in an error that is detected by the fault-tolerant architecture.
Based on this fault classification and on the different possibilities of responses of the AB outputs and comparator (OK or FAIL) we obtain the four cases detailed in Table III.4. We consider Silent faults as those that are not detected by the comparator and for which AB outputs remain correct. They are masked.filtered by the logic of the arithmetic circuit or they propagate an erroneous response in the range of the considered WCE in the case of scenario S4. Detected Faults are those detected by the comparator (FAIL flag). Finally, Fail-Silent Faults occur when the comparator does not detect that AB outputs are wrong. This happens when the SET affects an unprotected part of AB with respect to the reduced duplication RCB.

<table>
<thead>
<tr>
<th>AB Output values</th>
<th>Comparator output</th>
<th>Fault Status</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>Silent</td>
<td>The SET injected is filtered/masked.</td>
</tr>
<tr>
<td>OK</td>
<td>FAIL</td>
<td>Detected</td>
<td>The SET affects RCB and is detected by the comparator.</td>
</tr>
<tr>
<td>FAIL</td>
<td>OK</td>
<td>Fail-Silent</td>
<td>The SET affects an unprotected area of AB. The fault corrupts AB outputs but is not detected by the comparator.</td>
</tr>
<tr>
<td>FAIL</td>
<td>FAIL</td>
<td>Detected</td>
<td>The SET affects AB outputs and is detected by the comparator.</td>
</tr>
</tbody>
</table>

III.4.B. Fault Injection Analysis

Results of the fault injection campaign on the four scenarios as case studies are reported in Figure III.13. S1 injection campaign shows that 79% of the faults for the 8-bits adder (77% for the 8-bits multiplier; 87% for the 12-bits multiplier; 90% for the 16-bits multiplier, respectively) are masked.filtered and thus are classified as Silent Faults; 21% (23%; 12.5%; 10%, respectively) of the faults trigger the comparator and thus are classified as Detected Faults; there are no Fail-Silent Faults since the comparator covers every outputs of both precise circuits.
a) 8-bits Adder

b) 8-bits Multiplier
Fault classification for a) an 8-bits adder and b) an 8-bits multiplier c) a 12-bits multiplier d) a 16-bits multiplier

Figure III.13
With respect to S1, we compare the other three scenarios. S2 and S3 injection campaigns show a huge increase of Fail-Silent Faults for both case studies. A part of Detected Faults becomes Fail-Silent Faults. This is due to the fact that when S2 or S3 is used, not all the structure/function of AB is duplicated. When the SET corrupts an unprotected part of AB, it cannot be detected and hence is classified as Fail-Silent Fault. The percentage of Fail-Silent Faults is related to the duplication ratio. S4 injection campaign presents an interesting profile. In fact, the number of Fail-Silent Faults remains very low and below 50% of the Fail-Silent Faults rate of S2 and S3. This is explained by the fact that when a SET corrupts AB outputs, the corruption must be large enough (i.e. above the considered WCE) to be detected by the comparator. Results are different for the case studies. For the 8-bits adder, since the WCE is 7 (see Table III.3) most of SET corrupting AB outputs are detected and are thus classified as Detected Faults. In the case of the 8-bits multiplier, the WCE is 2292 (see Table III.3) So, most of the injected SET does not corrupt AB outputs sufficiently. Faults are then classified as Silent Faults. This phenomenon is even more noticeable for 12 and 16-bits multiplier as their WCE is even higher (16765 and 79154 respectively). Thus, we observed very few Detected and Fail-Silent Faults for these two cases. This is due to the fact that it gets very difficult to corrupt the circuit without a masking effect. Half of the faults that corrupt the circuit in a way that its output show an erroneous response are Fail-Silent Faults. This is because the impact is higher than the WCE allowed by the structure.

In addition to experimental results in terms of area/power overhead and error metric values shown in Section III.3, validations using fault injection presented in this section demonstrate the interest of using AxC structures to build a duplication scheme for error detection.

Partial duplication as explored in the late 90s and early 00s is not very interesting since the designer has to sacrifice too much accuracy with very low cost reduction. We can observe this downside effect from results of S2 and S3. The new approach in which the duplication is an AxC circuit (S4), however, offers new perspectives. From our results on S4 shown in Figure III.10, we observe that with an approximate copy, the area overhead goes from nearly 100% down to almost 50%. Even with approximate circuits with half the area cost of a precise circuit, the precision loss is limited. For example, in the 12-bits multiplier case study, where the area overhead of some AxC circuits is nearly 50%, the WCE is set at 1/16th of the maximum value it can deliver. Additionally, the injection campaign proves that with S4, the ratio of fail silent faults (undetected faults for which arithmetical value is above the established detecting threshold) remains very low in comparison with more traditional partial duplication methods like S2 and S3.
Conclusion

In this chapter, we have addressed the challenges related to selective hardening of arithmetic circuits. We have considered a duplication/comparison scheme as error detection architecture with different duplication scenarios, and we have discussed the different trade-offs achievable with each scenario. Experimental results have shown the interest of using approximate structures as duplication elements. Both area overhead and power consumption of the studied circuits are reduced compared to a full duplication scheme, while maintaining good levels on error metric values. Moreover, the interest of using approximate structures as duplication elements has been validated with the help of a fault injection campaign. We have shown that when approximate structures are used, the number of fail silent faults is less compared to other duplication scenarios.

The results that compare the different scenario performances were published in three workshops [8], [9] and [10] as well as an international conference [5]. Finally the fault injection validation was presented in a workshop [7] and published as an article in an international journal [1].
CHAPTER IV
QAMR: FULL RELIABILITY BASED ON QUADRUPLE APPROXIMATE REDUNDANCY

IV.1. Introduction

During the lifespan of a system used in harsh (e.g. radiative) environment, its hardware is subject to various physical phenomena that may alter its performance or provoke errors [WEI2016]. Moreover, some systems demand a high level of reliability since failures would imply catastrophic outcomes. Aerospace systems, submarine telecom or even medical instruments cannot risk particle strikes, wear-out or aging. However, high levels of reliability usually require heavy fault tolerant designs to reach such high requirements.

Several structures have been designed to maintain the accuracy of these safety-critical applications. A well-known existing structure capable of tolerating soft and hard errors is the Triple Modular Redundancy (TMR) presented in subsection II.3.E. A tripling of the circuit with a majority voter ensures an extreme logic error masking at a cost of a 200% area and power overhead. As previously stated, a TMR masks (tolerates) permanent or transient faults occurring in one or several modules (provided that they do not impact the same outputs if several modules are faulty) for any vector applied to its inputs.

On a totally opposite reliability point of view, AxC philosophy does not seem to be compatible in serving the interests of systems designed for safety-critical applications. AxC has been applied to resilient applications, e.g. speech recognition, image encoding, etc., where an approximate result is sufficient for their purpose [SAN2012]. From the hardware standpoint, AxC enables the creation of circuits whose output values may differ from the original circuit for a certain set of input values [MIT2016].

In [IUR2015-1], AxC was applied to TMR, where two or even three of the modules are different approximations of the original circuit. Other proposals of a low cost TMR based on approximate computing were presented in [SIE2006] and more recently in [SAN2012] and [IUR2015-2]. Such AxC applications to TMR lead to both lower area overhead and power consumption. However, such advantages come at the expense of a reduced error-masking capability, which makes approximate TMR not suitable in safety-critical scenarios.

To overcome the above issue, we propose the Quadruple Approximate Modular Redundancy (QAMR). QAMR is a novel scheme to ensure a full logic masking (tolerance)
of transient and permanent faults. Like \textit{TMR}, \textit{QAMR} masks all faults occurring in the modules and for which the voter still has a majority of correct responses. It achieves the same accuracy than the \textit{TMR} while still benefiting from approximation advantages (i.e., smaller area and power overhead). To implement the \textit{QAMR}, we use four approximate circuit replicas. The fundamental condition to respect is that, at a given time, at least three precise responses (i.e., non-approximated) must be delivered by the \textit{QAMR} structure. In other words, the four Approximate Integrated Circuits (\textit{AxIC}) must be approximated in a complementary manner.

In this chapter, we present the \textit{QAMR} approach and a simple circuit approximation method developed to demonstrate its advantages. This approximation method is based on complementarily cutting outputs (and the related fan-in internal logic) from each circuit replica composing the \textit{QAMR}. This is done in such a way that three replicas of the same output are always available. Consequently, we are able to use the same majority voter as in \textit{TMR} schemes. To validate our approach experimentally, we used publicly available combinational circuits to implement the \textit{QAMR} scheme. Experimental results show promising results that encourage a deeper exploration. Indeed, for several benchmarks, \textit{QAMR} achieves a smaller area overhead than the \textit{TMR}, while still providing the same reliability level.

This chapter first introduce the concept of Approximate \textit{TMR} and discusses its limitations towards safety-critical applications. Next section presents the novel concept of Quadruple Modular Redundancy and its advantages. Section IV.3 exposes the design flow of the \textit{QAMR} approach. The following section shows the experimental setup and discuss the \textit{QAMR} performance obtained in terms of area, power and timing cost with respect to the \textit{TMR}. Last section summarizes the chapter and gives a conclusion on the use of \textit{AxIC} for full reliability designs.

\section*{IV.2. State-of-the-Art on AxC Based Fault Tolerance}

\textit{TMR} is a fault-tolerant scheme made of three identical instances of a circuit connected to a majority voter. \textit{TMR} protects against faults (permanent or transient) occurring in one or several modules (provided that they do not impact the same outputs if several modules are faulty), for any input vector. This fault-tolerant solution requires a 200\% area overhead due to the two extra circuit instances. Moreover, we must add the voter area that depends on the number of circuit outputs.

In the literature, several proposals have been made to reduce the \textit{TMR} area overhead by using \textit{AxC}. This scheme is known as \textit{Approximate TMR (ATMR)} [IUR2015-1] and its extension as \textit{Full ATMR (FATMR)}. The \textit{ATMR} scheme uses two \textit{AxIC}s and a precise one as replicas, while \textit{FATMR} uses three \textit{AxIC}s. In these implementations, only
one AxIC can give an erroneous answer at a time. In other words, each approximate module has its own unique domain of approximation. However, producing such a low cost TMR may suffer from severe limitations in term of reliability.

Let us resort to Figure IV.1 to illustrate the above mentioned issue. A green block in the figure refers to an original or approximate replica of the circuit that provides a correct output when the vector $x$ is applied at its input. Conversely, a yellow block refers to an approximate replica of the circuit that provides a wrong output (because of the approximation) when the vector $x$ is applied at its input. The figure shows the different possible scenarios in the case of a single fault. Figures IV.1a and IV.1b show the ATMR scheme in two different scenarios. In both scenarios, a Single Event Transient (SET) occurs in one of the three replicas. The outcome, however, is very different depending on the nature of the faulty replica. In the case of Figure IV.1a, the SET occurs in the replica that gives an approximate wrong response for the input vector $x$. In this case, the voter will deliver a precise output because the two remaining replicas are giving a precise response for the same input vector $x$. Conversely, in the case of Figure IV.1b, the SET occurs in one replica that might have delivered a correct response. In this case, the voter will deliver an incorrect output. Figures IV.1c and IV.1d show the same scenarios. The only difference is that the FATMR scheme has only approximate replicas. If the SET occurs in a replica delivering a wrong approximate response for the input vector $x$ (Figure IV.1c), the voter will deliver a correct output. However, if the SET occurs in a replica that should deliver a correct response for the input vector $x$, the voter will deliver an incorrect output. In summary, input vectors when only two out of three replicas compute correctly are vulnerable to SET. The authors refer to those vectors as unprotected.
Figure IV.1  ATMR and FATMR SET scenarios
Hardware design of fault tolerance circuits for safety-critical applications is a crucial task. Realizing it by using AxC-based schemes raises some important challenges. Specifically, it is mandatory to know the workload of such application. For FATMR schemes, it implies that input vectors that are not protected by the structure must not be critical for the application. Such design requirements can be challenging and not always achievable, even for resilient applications.

IV.3. Proposed QAMR Scheme

The goal of our approach is to achieve the TMR reliability level while reducing area and power costs. We propose to make use of AxC in a quadruple duplication scheme. As with a classic TMR, the goal is to protect the whole structure function for all input vectors against permanent and transient faults occurring in one or several modules (provided that they do not impact the same outputs if several modules are faulty). Such fault-masking coverage will be suitable for safety-critical applications even when the workload is unknown.

![TMR, FATMR and QAMR single fault masking](image)

Figure IV.2 TMR, FATMR and QAMR single fault masking

Figure IV.2 shows the principle of the proposed QAMR scheme. In this case, if we consider the precise modules in TMR having precision domains (D1, D2, D3 and D4) triplicated, we cover each precision domain against any single fault scenario. FATMR covers most of their precision domains but a single fault may affect the outcome. For example, the structure would mask a fault on D1 (as D1 appears in all three modules of FATMR structure) whereas a fault on D2, D3 or D4 could not be masked. Instead, QAMR offers a complete coverage since we triplicate all the precision domains. By doing so,
we can reach the same TMR reliability level. At the same time, the four AxICs enable the opportunity to achieve efficiency gains in terms of reduced area and power consumption. The underlying insight is that a good AxC technique achieves more gains than it reduces the system accuracy.

Hereafter, we formalize the conditions required to generate a QAMR structure: i) the circuit must have at least four outputs in order to obtain four AxICs, ii) for each AxIC with missing accuracy for a specific set of input vectors, all other AxICs must tolerate this deficiency and provide a correct output response.

To implement our QAMR approach, we developed a circuit approximation method to produce the AxICs respecting the conditions mentioned above. The goal of this work is to simply demonstrate the QAMR feasibility and the opportunities it creates for fault-tolerant architectures. The design of optimal approximation techniques is left out for future works. Our preliminary method is illustrated in Figure IV.3.

For a given circuit, we remove one group of outputs from the original circuit to obtain a first AxC module, and we repeat this process as many times as needed to form four different AxICs. Meanwhile, we also remove the fan-in logic belonging to the group of outputs removed for each different AxIC. To respect the above mentioned conditions (complete coverage of the precision domains), it is important that an output is removed from only one of the four approximate replicas. The advantage of such a structure is the use of the same voter than the TMR scheme. Indeed, for any input vector, the voter will always deal with three bits for each output of the circuit. For better clarity, Figure 3 shows an example of a 4-bits output circuit in the QAMR scheme. Since each AxIC has
only one missing output, the voter is able to execute the majority vote just like in a classic TMR scheme.

**IV.4. QAMR Design Flow**

Exploring all possibilities of iteratively removing one group of outputs for each circuit is not possible when using circuit with a large number of outputs. We applied our method to benchmarks with up to 245 outputs. The number of complementary groups of outputs for QAMR for a circuit having 245 outputs is $3.19 \times 10^{63}$. To avoid such a huge and unpractical exploration, we arbitrarily generate complementary groups of outputs in a random manner to obtain one QAMR version, then evaluate the area variance between the generated QAMR version and the previous ones, and finally iterate the process until the variance becomes lower than a threshold defined by the user. QAMR version with the lowest area is the final (best) QAMR.

Figure IV.4 sketches the flow of the proposed circuit approximation method. Starting from the netlist of the original circuit, a direct synthesis allows creating TMR by adding a majority voter for further comparison with QAMR. In parallel, we arbitrarily generate complementary groups of outputs in a random manner to create four distinct approximate versions of the original netlist.

![Diagram of QAMR design flow](image-url)
Once the algorithm has provided four approximate versions with the corresponding complementary groups of outputs, we perform a logic synthesis and obtain the four AxICs (modules) of the QAMR structure. Each module lacks of one group of outputs and their respective fan-in logic. We create the QAMR by adding a majority voter.

Then, if the area of the QAMR is smaller than the area of the TMR, we compare the new synthesized QAMR to a former best QAMR version stored in a database. Each time the area of a new QAMR is smaller than the area of the best QAMR version, it is saved and the former best QAMR candidate is overwritten in the database. If the area of the QAMR is larger than the area of the TMR or of the best QAMR, the QAMR version is discarded. Besides, each time a new QAMR version is synthesized, an area report is generated to update the area variance calculated by using Equation IV.1:

\[
V_A = \frac{1}{n} \sum_{i=1}^{n} (A_i - \bar{A})^2
\]

where \(V_A\) represents the area variance, \(\bar{A}\) is the mean area value of all the QAMR versions created so far and \(A_i\) is the area of the QAMR version obtained during the \(i^{th}\) iteration of the process described in figure IV.4, and \(n\) is the total number of iterations. Once a new QAMR version has been generated, compared, and saved or discarded, the variance is compared to a threshold defined by the user. If it is higher than the threshold, the process is re-run and an additional iteration will provide a new QAMR version. Otherwise, the algorithm stops and the best QAMR version saved in the database is considered as the final QAMR.

Note that, the two values used during the first process iteration to calculate the variance are the following: i) the area of the first QAMR generated version and ii) the TMR area to which we compare our QAMR area.

To also select the best QAMR versions in terms of power and timing gains, the same design flow was applied two additional times. An additional QAMR selection was made by comparing the power consumption of each new versions of the QAMR with respect to the TMR power consumption. The last selection was made by comparing the maximum delays of each new version with respect to the maximum delay of the TMR.

**IV.5. Experimental Results**

**IV.5.A. Experimental Setup**

Experiments have been done by using the Combinational Multi-Level and Two-Level circuits from the publicly available LGSynth’91 benchmark suite [YAN1991]. For each circuit, we obtained the classic TMR by using three precise versions of the circuit and a voter. In a same way, we composed the QAMR with four approximate versions of the
circuit and the same voter. We used the principle described in subsection IV.3 to create the four approximate versions for each circuit. We used Design Compiler of Synopsys [SYN] for circuit synthesis, using the NanGate 45nm Open Cell Library [SIL].

From the LGSynth'91 benchmark suite, we select only circuits that have five or more outputs. This is important, as the first step after identifying the logic function of a given circuit is to form random groups of outputs. In the case a circuit has less than four outputs, the creation of four approximate modules is impossible. Note that with four outputs, the random selection of output groups would have always give the same combinations.

### IV.5.B. Area Results Analysis

To fairly compare our results with those obtained with the TMR scheme, we use a Relative Area Gain (RAG) metric. Note that results do not consider the voter area since it is the same in both schemes. This means that we consider the TMR as our baseline with 0% of area gain. Thus, the higher the RAG, the better the QAMR area performance with respect to the TMR. Equation IV.2 shows how RAG is calculated. \(A_p\) represents the area of each precise circuit in the TMR. \(A_{xn}\) represents the area of each approximate module in the QAMR.

\[
RAG = \frac{3A_p - (A_{x1} + A_{x2} + A_{x3} + A_{x4})}{3A_p} \quad (IV.2)
\]

Figure IV.5 shows RAG achieved for all benchmark circuits used in our experiments. Results indicate that 19 out of 52 circuits have a lower RAG when using the QAMR scheme. Some circuits like Apex1 or e64 have a RAG above 20%. On the contrary, RAG from circuits presenting area loss is generally staying above -10%. Peculiarly, results for Alu4, however, are really poor with our QAMR approach, with an additional area cost of 40%.

At first glance, the simple fact that a good proportion of circuits cost less area for the QAMR approach with respect to the TMR allow us to state that full reliability is not only achievable using AxC but also circumstantially cost-effective.
In addition to area gain results, we also obtained Relative Power Gains (RPG) of each circuits. This metric, defined in equation IV.3, is obtained with estimated values (rather than exact values) given by the commercial synthesis tool. $P_p$ represents the estimated power of each precise circuit in the TMR. $P_{xn}$ represents the estimated power of each approximate module in the QAMR.

$$RPG = \frac{3P_p - (P_{X1} + P_{X2} + P_{X3} + P_{X4})}{3P_p}$$  \hspace{1cm} (IV.3)

Even though this metric is an estimation, they are relative between every TMR and QAMR we explored, making so that a fair comparison of the power consumption is finally obtained. As shown in figure IV.6, Relative Power Gain (RPG) obtained is positive for nearly 60% of the studied circuits with 14% of the circuits above 20% RPG. The best case is the circuit K2 that performed very well with 46% RPG. On the contrary, RPG for circuits presenting power loss is generally above -10% RPG and represents 24% of the circuits (with a worst case of -12% RPG for alu4 circuit). Note that 7% of the circuits have a 0% RPG.

In general, circuits that have a positive RPG tend to perform with more magnitude than the underperformance of circuits with negative RPG. Although power results are very encouraging, they also confirm the interest of the performing a deeper design exploration.
**Figure IV.6**  Power gained by QAMR compared with TMR

### IV.5.D. Timing Results Analysis

The last metric we considered in terms of costs to compare with the TMR is the Relative Timing Gain (RTG). RTG, defined in equation IV.4, give an indication of the potential performance enhancements possible to accomplish in terms of system frequency. $T_p$ represents the timing of each precise circuit in the TMR. $T_{xn}$ represents the estimated timing of each approximate module in the QAMR.

\[
RTG = \frac{3T_p - (T_{x1} + T_{x2} + T_{x3} + T_{x4})}{3T_p} \quad (IV.4)
\]

Timing values are based on the timing of the longest path in TMR and QAMR schemes. As shown in figure IV.7, the RTG obtained is positive for nearly 63% of the circuit have a positive RTG, with 30% above 10% RTG (with a best case of 75% RTG for e64 circuit). Only 5% of the experimented circuits have a negative RTG and 19% present 0% RTG.

Those results also confirm the interest of the presented design exploration as they are overall positive and the negative impact, on the few circuits that have more delay increase, is not meaningful.
IV.5.E. Shared Logic Rate

With principally area results having such heterogeneity, we analyzed which parameters could help in determining what type of circuit would be suitable for a QAMR approach. We came to the following observation. Our approximation method consists in removing outputs and their associated fan-in logic without removing the logic shared with the other (preserved) outputs. One key characteristic of such method is therefore the number of nodes in the circuit that lead to more than one output. This number divided by the total number of nodes gives the so-called Shared Logic Rate (SLR) of the circuit.

Let us assume that the success of the QAMR approach remains in removing output’s cones that share most of their logic with other cones. The removal of a cone with such a high SLR will allow the synthesis tool to remap the circuit in a configuration that was not necessarily interesting before. Indeed, synthesis tools rely on heuristics to perform the best possible technology mapping. It is reasonable to assume that a synthesis tool can perform better with a circuit that has been simplified by removing logic from it. On the contrary, the synthesis tool will not be able to remap and optimize the remaining logic that was shared with a low SLR output cone since the remapping options are more limited.

Figure IV.7 shows results obtained previously, this time ranked from circuits with the highest SLR to circuits with the lowest. We observe that below 20% of SLR, our QAMR scheme underperforms the TMR most of the time. We can see that only 6 out of 23 circuits with an SLR lower than 20% have a positive RAG. On the other hand, 12 out of the 29 circuits with an SLR higher than 20% outperform the TMR versions by achieving a positive RAG.
In addition to area gain results, we obtained relative power and timing gains as well. Power consumption values are estimated values (rather than exact values) given by the commercial synthesis tool. Nonetheless, they are relative between every TMR and QAMR we explored, making so that a fair comparison of the power consumption is finally obtained. Timing values are based on the timing of the longest path in TMR and QAMR schemes. Relative Power Gain (RPG) obtained is positive for nearly 60% of the studied circuits with 14% of the circuits above 20% RPG (with a best case of 46% RPG for K2 circuit). On the contrary, RPG for circuits presenting power loss is generally above -10% RPG and represents 24% of the circuits (with a worst case of -95% RPG for ex5 circuit). Note that 7% of the circuits have a 0% RPG. Regarding the Relative Timing Gain (RTG), 63% of the circuit have a positive RTG, with 30% above 10% RTG (with a best case of 75% RTG for e64 circuit). Only 5% of the experimented circuits have a negative RTG and 19% present 0% RTG. Those results also confirm the interest of the presented design exploration.
Conclusion

In the context of error-tolerant applications, approximate computing trades off some computing accuracy with increased performance, decreased area footprint and/or power efficiency. In this context, studies in the literature proposed to relax reliability constraints to achieve gains in circuit area and power consumption. Despite the efficiency optimization opportunities brought by this kind of techniques, reliability still represents a key requirement in most advanced safety-critical computing systems: sacrificing reliability could result in the production of more cost-efficient systems, but also in endangering human lives. In particular, previous works on approximation-based TMRs presented the advantage of reducing its area cost compared to the standard TMR. However, such advantage comes at the expense of a reduced fault tolerance, preventing the Approximate TMR to be used in safety-critical applications. This chapter introduced the first solution to profit from the benefits brought by AxC, without sacrificing the reliability requirements. We proposed the novel Quadruple Approximate Modular Redundancy (QAMR) to reduce the standard TMR area cost without sacrificing the offered QoR.

To investigate the feasibility of the approach, we used a simplistic method based on the removal of a random portion of output’s cones for each one of the AxC. Despite that, we managed to obtain very promising results showing that it is possible to use AxC to reduce area costs without sacrificing reliability requirements. Obtained results published in an international conference [7] clearly indicate that QAMR offers a cheaper alternative to the standard TMR scheme for safety-critical applications.

The case study results obtained from a bundle of circuits show that not only QAMR is feasible, but it is also far from being an anecdotic achievement. Although results are far from unanimous, the gains obtained are significant and demonstrate that there is a genuine interest in pursuing the trail of QAMR and the use of AxC for safety-critical applications.

Further studies now are needed to establish enhanced approximation techniques to fully exploit AxC opportunities in safety-critical scenarios. Although the SLR gave a hint on how to classify circuits more prone to give good area gains for the QAMR approach, there are many other criteria to encompass. A first possibility is to enhance the approach used in this work, by smartly selecting the output groups to remove for each AxC. Understanding which cones are determinant to the simplification process should allow new uneven combinations of outputs and thus, enhance the area gains.

The works presented in this chapter were published in two international conferences [3] and [4].
CONCLUSIONS AND PERPECTIVES

Regardless the field of application, i.e. trading, health-care, satellite telecoms, civilian transports, military equipment, data centers, etc., there is a performance growth demand on electronics to execute an infinity of complex operations. Most of these operations demand a high degree of reliability, availability and safety. However, the increasing vulnerability of transistors and interconnects require electronic system designs to overcome challenges for every new emergent generation of the CMOS technology. Furthermore, the complexity of modern electronic systems renders error detection, recovery, masking, etc. a difficult task. A few other aspects to take into account are area and power limitations as well as high performance demands. These requirements force the industry to limit the overheads in reliability enhancements or come up with more adaptive designs that respond to the reliability problematic of the targeted field of application.

The objective of this thesis was to provide new cost-effective fault-tolerant methods to achieve a better trade-off than traditional approaches. An important parameter was to adapt the developed methods to the level of reliability required by the application that would run on such fault-tolerant approaches. This thesis explored of two approaches, one that would be suitable for resilient applications and another that would meet the requirements of safety-critical applications.

In this work, we explored a new approach in designing fault-tolerant architectures that deals with faults in combinational logic before the corrupted computation reaches memory elements and lead to system failure. Given the context of power and area limitations as well as performances needs, we concentrated on an emerging topic known as Approximate Computing. Even though AxC does not align with the direction of Quality of Resilience that many systems require, we believed that a judicious use of it could be profitable.

To prove so, we first performed a low-cost reliability estimation comparison of AxC designs with respect to more traditional selective hardening methods in a duplication and comparison configuration. Our approach of using approximate-based CL redundancy showed the benefits that it brings to the cost versus reliability trade-off. The fault injection campaign confirmed that, for resilient applications, the use of AxC to optimize the cost-reliability trade-off could be an interesting asset.

Secondly, we proposed to develop a structure based on approximate CL that accomplishes the same functions as the TMR structure at a reduced area, power and timing cost. To do so, we developed a simple method to create four approximate copies
of a CL circuit in a way that each copy would be approximate and complementary between them. The QAMR resulting structure achieves full reliability by using only approximate copies. Despite our very simplistic approximation approach, our QAMR performed very well in terms of costs, even better than the TMR in many cases for an equivalent reliability level.

The interest and advantages of AxC in fault-tolerant structures for two different fields of applications (resilient and safety-critical) have proven to be possible all along this thesis work. However, in order to establish reliable design protocols, more characterization is needed as well as a deeper exploration of the mechanisms that can be used to join AxC to the fault-tolerance designs. The next two subsections discuss our perspectives to extend the work presented in this manuscript.

**Towards an Approximate aging aware Fault-Tolerant Architecture**

Considering the scenarios from chapter III, with S1 any failure will be detected while with S2-S4 scenarios, a failure is detected only if it is hard enough to be sensed by the comparator. This is the case of any fault-tolerant approach masking any failures that are below the threshold that the schemes are able to tolerate. To open future developments in this direction, this section presents the lifespan of the circuit and warnings sent to the user in the following cases: i) the circuit presents tolerable faults and is not precise anymore and ii) the circuit presents faults that are not tolerable anymore and the circuit has been considered as faulty. In other words, we intend to use the S4 duplication scheme as indicator of the circuit aging and wear out (i.e. from the precise domain to the approximate one and to the approximate domain to the failing one).

*Figure V.1*  Lifespan of the circuit and warnings to the user
Figure V.1 illustrates the lifespan of a typical circuit: brand new and precise (green); approximate but with tolerable responses (orange); aged and permanently failing (pink). At this moment, with the above discussed scenarios, the error detection architecture is able to inform the user that the circuit is failing beyond the defined WCE threshold. However, to inform the user that the circuit is not precise anymore and is entering the approximate domain, it becomes necessary to investigate the new field of Approximate Fault-Tolerant Architectures with the development of specific comparator structures able to detect the different (approximate, non-approximate) domains. This is the purpose of our future work.

Future developments are required to provide a comparator suitable for S4 and hence take its area and power overheads into account in the full comparative study. It can be forecasted that such comparator will add area and power costs. So, the outcome of these further developments will be to demonstrate that this additional cost will not affect the gains obtained by using approximate large sized combinational circuits.

**QAMR: Functional Approach Perspective**

Although we developed the QAMR scheme by using a structural approach to prove its feasibility, other approaches must be explored. Among them, we can exploit the fact that circuits can also be approximated from a functional point of view. For example, representing a circuit as Sums of Products (SOP) could allow a designer to remove specific and different minterms for each AxIC. In that case, each AxIC would have a precision domain depending on input vectors rather than output cones. The challenge of such a functional approach resides in the design of a majority voting logic. Regardless of the functional technique utilized, it must discriminate an input vector from another to determine which AxIC will deliver an approximate response. This approach will be explored in the near future.

In general, more sophisticated and efficient logic synthesis techniques are required to fully profit from AxC opportunities, when it comes to safety-critical scenarios. In particular, advanced mathematical model are needed to turn an abstract specification of a desired QAMR behavior into an actual gate-level implementation. Synthesis tools based on such models will offer to designers a cheap alternative to the TMR scheme, still perfectly suitable in safety-critical contexts.
REFERENCES


SCIENTIFIC CONTRIBUTIONS

Journal


International Conferences


Seminars and Workshops


ABSTRACT

Transistor downscaling allows processors to continuously increase transistor density and to operate at higher frequencies. Although downscaling leads to higher performance and lower power consumption, each new CMOS technology node is facing reliability issues due to increasing rate of faults and errors that occur in electronic devices despite careful design and manufacturing processes. Consequently, most of systems today include fault-tolerant techniques that ensure correct operations of digital parts. These techniques employ redundancy to ensure that faults cannot cause system failures. This thesis studies the cost/reliability trade-off on fault-tolerant architectures based on structural redundancy exploiting the low-cost advantages of the approximate computing paradigm. The first contribution of the thesis is a selective hardening scheme that achieves fault detection with a duplication scheme that compares a precise and approximate version of the circuit. The second contribution of the thesis is a fault-tolerant architecture, called QAMR, which is able to mask the same faults that a TMR would, but at lower cost. Results of these works show that an appropriate use of approximate computing in redundancy schemes can achieve, at lower cost, the same reliability level than traditional techniques.

RESUMÉ

Les processeurs intègrent un nombre sans cesse croissant de transistors et opèrent à des fréquences de plus en plus élevées grâce à la miniaturisation de leurs composants élémentaires. Bien que cette miniaturisation permette de meilleures performances pour une consommation de puissance réduite, chaque nouveau nœud technologique est confronté à des problèmes de fiabilité accrus. En effet, malgré une conception et un processus de fabrication maîtrisés pour empêcher l’apparition de fautes, garantir un taux d’erreurs nul devient de plus en plus difficile. En conséquence, les architectures sont conçues en incluant des techniques de tolérance aux fautes qui assurent leur bon fonctionnement. Ces techniques reposent sur des principes de redondance pour garantir que les fautes qui apparaissent ne causent pas de défaillances du système. Dans cette thèse, une étude est faite sur le compromis coût/fiabilité sur des architectures tolérantes aux fautes inspirées du paradigme des structures de calcul approximées. La première contribution de cette thèse porte sur la conception de schémas de durcissement sélectif qui permettent la détection de fautes en comparant la version précise et la version approximée d’un circuit. La seconde contribution de cette thèse est une architecture de tolérance aux fautes appelée QAMR qui est capable de masquer les mêmes fautes qu’un TMR tolère mais à coûts réduits en surface et consommation. Les résultats de ces travaux démontrent qu’un usage approprié de structures de calcul approximées pour des schémas de redondance permet d’atteindre, à moindre coûts, un niveau de fiabilité égal à celui des techniques traditionnelles.