Cross-layer Soft Error Analysis and Mitigation at Nanoscale Technologies

for obtaining the academic degree of

Doctor of Engineering

Approved

Dissertation

Department of Informatics
Karlsruhe Institute of Technology (KIT)

by

Mojtaba Ebrahimi

from Khoy, Iran

Oral Exam Date: 11 February 2016

Adviser: Prof. Dr. Mehdi Baradaran Tahoori, KIT
Co-adviser: Dr. Dan Alexandrescu, iRoC Technologies

To The People of Middle East

Acknowledgments

First of all, I would like to sincerely thank my adviser, Prof. Mehdi Tahoori, who encouraged me to explore new directions, kept me motivated, involved me in various exciting research projects, and guided me throughout my PhD journey. During these years, I have learned lots of lessons from him which will help me in my future carrier.

My gratitude extends to my wife, Mahnaz, for her love. She encouraged me to follow my dreams while sacrificing most of her wishes to take care of our children. I owe her a lot and I hope that I can compensate her effort in near future.

During my PhD study, I had the chance to collaborate with many people from academia and industry. Without their support and help, I could never have achieved what I did. In this context, I would like to specially thank iRoC experts, Dr. Dan Alexandrescu and Dr. Adrian Evan, for their help during my internship and also for their constructive comments throughout my PhD. I also sincerely thank Dr. Vikas Chandra from ARM, Dr. Austin Lesea from Xilinx, and Prof. Hossein Asadi from Sharif University of Technology for their fruitful insights.

I would like to thank former and current members of chair of dependable nano computing for their thoughts, helps, and companionship. I would like to especially thank my friends, Liang Chen, Saman Kiamehr, and Rajendra Bishnoi which I have learned a lot from them, not only in research, but also in other aspects of life.

It was my fortune to work with several hardworking bachelor/master students who contributed to this thesis. I want like to thank Razi Seyyedi, Nour Sayed, Maryam Rashvand, Firas Kaddachi, Dominik Maszczyk, Mohammad Hadi Moshrefpour, and Wen Huang Lin for their extensive help in the implementation phase of various projects.

I am also grateful to the DFG Priority Program SPP 1500 (Dependable Embedded Systems) and Karlsruhe House of Young Scientist (KHYS) for the financial support of my work.

Mojtaba Ebrahimi
Lessingstr. 36
76135 Karlsruhe

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben haben und dass ich die Stellen der Arbeit - einschlielich Tabellen, Karten und Abbildungen - die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Karlsruhe, December 2015
Mojtaba Ebrahimi

ABSTRACT

The aggressive technology scaling over the last several decades revolutionized the semiconductor industry and made the computing systems indispensable part of our daily life. The rapid downscaling leads to increased performance, improved cost/energy efficiency, and more integrated features in modern digital integrated circuits. However, due to the increasing unreliability of devices manufactured in advanced technology nodes, it becomes harder for the designers to guarantee the correct functionality of the chip in the field. Therefore, in order to achieve cost-efficient reliable computing, it is crucial to take the reliability into consideration alongside with the conventional area, power, and performance metrics in the design flow.

Radiation-induced soft error is one of the major reliability concerns in nanoscale technology nodes. The soft error is generated when a radiation-induced particle, such as alpha particle in packaging material or neutron particle from cosmic rays, strike a transistor. The particle strike changes the status of a transistor for a very short period of time which might upset the value stored in a sequential element (i.e., flip-flop or memory cell) or temporally change the output of a combinational gate.

Although soft error has been widely studied over the last forty years, the further technology downscaling significantly changes error generation and propagation behavior. With smaller device geometries in advanced technology nodes, particles with smaller energy could cause soft error. In addition, one particle might affect several adjacent cells due to high transistor density resulting in *Multiple Bit Upset* (MBU) in sequential elements and *Multiple Event Transient* (MET) in combinational gates.

According to recent studies, due to the increasing transistor count per chip (i.e. Moore's law), aggressive transistor downscaling, reduced operating voltage as well as the increasing MBU/MET rate, the soft error sensitivity of integrated circuits has been exponentially grown over the past years. As a result, studying radiation-induced soft errors in modern electronic components and systems and exploiting appropriate modeling and mitigation techniques to address them in cost-effective manner is of decisive importance.

This thesis addresses the challenge of soft error modeling and mitigation in nansoscale technology nodes and pushes the state-of-the-art forward by proposing novel modeling, analyze and mitigation techniques. The proposed soft error sensitivity analysis platform accurately models both error generation and propagation starting from a technology dependent device level simulations all the way to workload dependent application level analysis. The platform employs a combination of empirical models at the device level, analytical error propagation at logic level and fault emulation at the architecture/application level to provide the detailed contribution of each component (flip-flops, combinational gates, and memory arrays) to the overall failure rate. At each stage in the modeling hierarchy, an appropriate level of abstraction is used to propagate the effect of errors to the next higher level.

Using the soft error analysis platform, the soft error sensitivity of several circuits designed in nanoscale technology nodes are investigated. The results of this analysis not only provide useful insights to designers for cost-effective protection of modern integrated circuits, but also invalidates several common assumptions on the relative failure rate of different components.

The proposed platform enables us to quickly evaluate the system-level impact of existing protection techniques. The results show that the conventional MBU/MET mitigation techniques employed to address the soft error issue are quite ineffective in nanoscales. Therefore, several low-cost solutions for mitigating MBU/MET in circuits fabricated using advanced technology nodes are introduced. The proposed mitigation techniques are:

- MET-aware placement strategy for random logic which can mitigate METs by 30% without imposing area and performance overheads.
- Low-cost MBU detection techniques for memory arrays considerably reduces the area and power overheads compared to the stat-of-the-art.
- Robust MBU correction technique for configuration memories of *Field Programmable Gate Array* (FPGA) devices which can detect and correct 100% of errors occurring in configuration memories. Compared to the previous solutions, our scheme provides the highest level of MBU protection at very low costs.

These low-cost solutions can significantly reduce the overall protection cost of systems designed in advanced technology nodes.

ZUSAMMENFASSUNG

Die aggressive Verkleinerung der Transistorabmessungen in den vergangenen Jahrzehnten hat die Halbleiterbranche revolutioniert, und Computersysteme zu einem essentiellen Bestandteil unseres alltäglichen Lebens gemacht. Die immer weiter fortschreitende Skalierung führt zu einer besseren Leistungsfähigkeit, verbesserten Kosten-/Energieeffizienz sowie einem stetig wachsenden Funktionsumfang moderner integrierter Schaltungen. In fortgeschrittenen Herstellungstechnologien wird es für die Designer und Hersteller allerdings zunehmend schwieriger die Zuverlässigkeit ihrer Chips im Betrieb zu gewährleisten. Deshalb muss die Zuverlässigkeit, ebenso wie die Leistungsfähigkeit, der Energiebedarf sowie die Größe einer Schaltung, ein wesentlicher Designaspekt werden, um kosten-effiziente und zugleich zuverlässige Computersysteme zu verwirklichen.

Eine der größten Zuverlässigkeitsherausforderungen stellen dabei sogenannte *Soft Errors* dar. Diese temporären Fehler werden durch das Auftreffen hoch-energetischer Teilchen, z.B Alpha-Partikel im Chip-Package oder Neutronen in der kosmischen Strahlung, auf Transistoren ausgelöst, was eine kurzzeitige Änderung des Transistorzustands auslösen kann. Somit kann der Inhalt einer Speicherzelle oder der Ausgang eines Gatters verfälscht werden.

Obwohl Soft Errors schon seit gut 40 Jahren bekannt sind und erforscht werden, hat sich mit den neusten Herstellungstechnologien das Verhalten der Fehlererzeugung und Fehlerfortpflanzung signifikant verändert. So führen die kleinen Strukturbreiten in modernen Technologien dazu, dass auch Teilchen mit geringer Energie zu Fehlern führen können. Außerdem kann ein Teilchen mehrere benachbarte Transistoren treffen, was Multi-Bit-Fehlern (MBF) in Speichern oder Multi-Event-Transienten (MET) in Gattern nach sich ziehen kann. All dies führt gemäßneusten Studien zusammen mit der immer weiter steigenden Transistorzahl (Mooresches Gesetz) dazu, dass die Soft Error Rate in den letzten Jahren exponentiell gestiegen ist. Deshalb ist das Studium von Soft Errors, sowie das Erforschen und Entwickeln von geeigneten Modellen und Methoden zur Reduzierung der Fehlerrate, in modernen Herstellungstechnologien von entscheidender Bedeutung.

Diese Dissertation adressiert die Herausforderung geeignete Modellierungsmethoden und Verfahren zur Reduzierung von Soft Errors zu entwickeln. Der Stand der Technik wird dabei sowohl in Bezug auf die Modellierungs- und Analysewerkzeuge als auch im Bereich der Methoden zur Fehlerreduzierung deutlich verbessert. Die entwickelte Soft Error Analyse Plattform modelliert die Fehlererzeugung und die Fehlerfortpflanzung sehr genau, durch die Berücksichtigung von Technologieparametern bis hin zu Anwendungseinflüssen. Aufbauend auf dieser Plattform kann die Sensitivität einer Schaltung gegenüber Soft Errors bestimmt werden. Die gewonnenen Erkenntnisse sind dabei nicht nur für die Chip-Designer von großer Bedeutung um kosten-effiziente Methoden zur Fehlervermeidung und Fehlerkorrektur zu entwickeln, sondern zeigen auch, dass gängige Annahmen über die Fehlerverteilung in unterschiedlichen Bauteilen (Flip-Flops, SRAM, Logik, etc.) nicht mehr gültig sind. Ein weiteres Ergebnis unserer Analysen ist, dass aktuelle Verfahren zur Fehlervermeidung und Korrektur sehr ineffizient in modernen Technologien sind. Deshalb werden in dieser Arbeit neue kostengünstige

Lösungen vorgestellt, die effizient das Problem der MBF und MET adressieren. Gegenüber dem aktuellen Stand der Technik können diese Methoden die Kosten für den Schutz von Computersystemen in modernen Herstellungstechnologien signifikant verringern.

Contents

List of Figures

List of Tables

1 Introduction

Computing systems are indispensable part of our daily life with various application domains ranging from consumer electronics such as smartphones and laptops to safety- and mission-critical components in aerospace and automotive systems. The ever increasing demand for higher performance and lower energy consumption of computer systems pushes the semiconductor industry towards further downscaling of integrated circuits. With a higher integration density, designers were able to significantly improve system performance by implementing more sophisticated architectures such as multi/many-core and heterogeneous systems.

Although the continuous technology downscaling is beneficial in terms of performance and energy, it comes with several design challenges. Recent studies reveal that the aggressive transistor downscaling coupled with the increasing transistor count per chip (i.e. Moore's law) and reduced operating voltage dramatically affect the reliability (i.e. ability to provide intended service) of digital circuits, particularly in nanoscale technology nodes [3, 6]. This means that the systems designed in smaller technology nodes are expected to experience more in-field failures. As a result, the reliability has to be considered as a new design metric in the design space exploration of digital circuits fabricated in advanced technology nodes.

The main sources of unreliability in nanoscale integrated circuits are the process and runtime variations as well as the radiation-induced soft errors [6]. Process variation, which is the manufacturing variation of transistors (length, width, and oxide thickness), results in variation on the characteristics of the designed circuit. Runtime variations due to supply voltage fluctuations, temperature variation, and transistor aging change (degrade) the device properties (e.g. delay) during the chip operational lifetime. From circuit-level prospective, the process and runtime variations mostly affect the circuit delay and could be effectively mitigated by exploiting an additional timing margin as guardband. Radiation-induced soft errors are generated when radiation-induced particles, such as alpha particles in packaging material or neutrons from cosmic rays, strike a transistor. The accumulated charge due to the particle strike in the transistor may temporally alter its state, thereby introducing a transient fault into the normal system operation. The radiation-induced soft errors, which manifest in the form of transient computational errors, have a random nature. As a result, the low-cost detection/correction of soft errors is very challenging and require more sophisticated techniques.

The discovery of soft errors dates to 1970s and they been widely studied over the last 40 years. Soft errors were primarily a concern only in space applications. However, as technology scales, transistors become more sensitive to these errors not only due to their smaller drive strength but also due to sensitivity to more radiation sources (e.g. thermal neutrons and protons). The growing sensitivity makes soft errors a key challenge in design of a wide range of application domains at the ground level such as networking, servers, medical, and automotive electronics [7].

1.1 Problem Statement and Objective

Although the source of soft errors are device-level interactions, the generated errors could propagate and cause system-level failures. As a result, it is very important to analyze the impact of soft errors using a device to system-level approach. Therefore, an efficient soft error vulnerability estimation technique has to be able to accurately model the error generation at device-level as well as the masking behavior at higher abstraction levels. Masking happens when the effect of error is completely eliminated from the system. For example, an error in memory entry followed by a write access is overwritten and get masked. Indeed, due to various masking effects at circuit, logic, architecture, and application levels, the majority of the generated errors end up being vanished and only very few lead to visible failures [8, 9]. Non-uniform masking behavior results in vulnerability of different components being significantly unbalanced. Therefore, a fast and accurate soft error sensitivity estimation technique is required to identify the most vulnerable components (e.g. cache or floating point unit) and apply selective protection schemes.

An effective soft error sensitivity evaluation technique has to identify the contribution of each component to the overall *Soft Error Rate* (SER). The SER of each component (e.g. flip-flop or memory array) could be computed by multiplying *Failure in Time* (FIT) and *Architectural Vulnerability Factor* (AVF) [7]. FIT specifies the error generation rate and depends on the device-level parameters and working environment. In contrast, AVF is a conditional probability that a generated error propagates and causes system-level failure (e.g. wrong workload output in a processor) [10]. AVF depends on the functionality of the RTL model as well as the running workload characteristics.

Radiation testing is a fast approach for SER estimation, however, this approach cannot be used as part of design flow and it is not possible to identify the contribution of each component as it cannot differentiate the source of the error in complex circuits. In order to address this issue, variety of analytical SER estimation techniques and fault injection experiments have been proposed. However, such solutions only model a subset of error generation and propagation behavior, failing to capture the full picture. The main objective of this thesis is to exploit the combined knowledge of various layers for presenting a cross-layer framework to investigate the contributions of combinational gates, flip-flops, and memory arrays to the overall SER in typical embedded processors. This study provides useful insights to designers for cost-effective protection of embedded processors.

With smaller device geometries in nanoscale technology nodes, one particle strike might affect several adjacent cells resulting in *Multiple Bit Upset* (MBU) in sequential cell and *Multiple Event Transient* in combinational gates. Multiple errors in older technology nodes could be safely ignored and therefore the detection and correction schemes were mostly concentrating on single errors. The experimental studies show that the share of multiple errors in the overall SER grows with the technology downscaling [3, 11] and it is the dominant contributor to the SER in the advanced technology nodes [1, 12]. In such scenarios, existing single error detection and correction schemes are no longer effective and could lead to catastrophic consequences. In this thesis, we also aim to provide cost-effective solutions for MBU mitigation in circuits fabricated in advanced technology nodes.

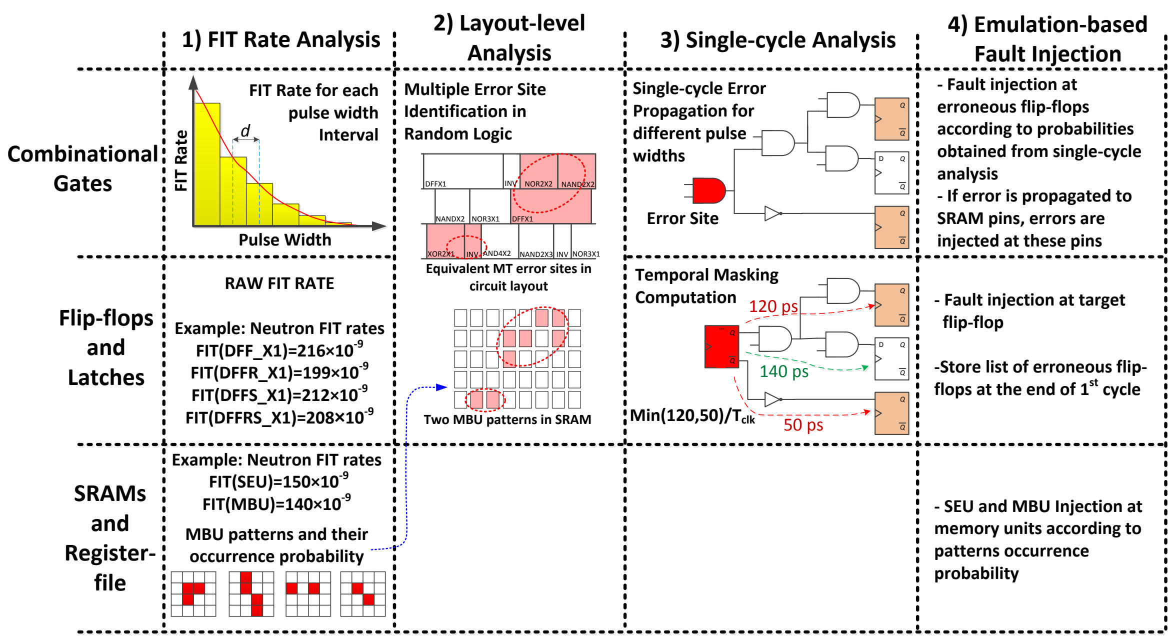

1) FIT Rate Analysis
2) Layout-level Analysis
3) Single-cycle Analysis
4) Emulation-based Fault Injection
Combinational Gates
Flip-flops and Latches
SRAMs and Register-file
FIT Rate for each pulse width Interval
FIT Rate
Pulse Width
d
Multiple Error Site Identification in Random Logic
Equivalent MT error sites in circuit layout
Single-cycle Error Propagation for different pulse widths
Error Site
- Fault injection at erroneous flip-flops according to probabilities obtained from single-cycle analysis
- If error is propagated to SRAM pins, errors are injected at these pins
RAW FIT RATE
Example: Neutron FIT rates
FIT(DFF_X1)=216×10^-9
FIT(DFFR_X1)=199×10^-9
FIT(DFFS_X1)=212×10^-9
FIT(DFFRS_X1)=208×10^-9
Two MBU patterns in SRAM
Temporal Masking Computation
120 ps
140 ps
50 ps
Min(120,50)/Tclk
- Fault injection at target flip-flop
-Store list of erroneous flip-flops at the end of 1st cycle
Example: Neutron FIT rates
FIT(SEU)=150×10^-9
FIT(MBU)=140×10^-9
MBU patterns and their occurrence probability
- SEU and MBU Injection at memory units according to patterns occurrence probability

Figure 1.1: Proposed cross-layer SER analysis flow

1.2 Contributions

During my Ph.D. studies, I developed a cross-layer SER analysis platform which can accurately model all error generation, propagation and masking effects starting from a technology response model derived at the device level all the way to the application-level. Using this platform, I comprehensively studied SER of embedded processors and realized mitigation opportunities. I exploit such opportunities by proposing several low-cost mitigation techniques to reduce the protection cost compared to existing solutions.

1.2.1 Cross-layer SER Analysis Platform

The proposed cross-layer SER analysis platform employs a combination of empirical models at the device level, error site analysis at chip layout, analytical error propagation at logic level and fault simulation/emulation at the architecture/application level to provide the detailed contribution of each component (flip-flops, combinational gates, and memory arrays) to the overall SER. At each stage in the modeling hierarchy, an appropriate level of abstraction is used to propagate the effect of errors to the next higher level. The overall flow of this platform is shown in Figure 1.1.

The most important novel contributions in this platform are:

- **Layout-based error site identification** [13, 14] : With smaller device geometries in nanoscale technologies, it is very likely that a high energy particle strike affects several adjacent cells in a circuit. One major challenge of SER analysis in advanced technology nodes is identfying error sites that could be affected by one strike. Unlike previous techniques in which the adjacent error sites are obtained from logic-level netlist, I perform a comprehensive layout analysis to extract adjacent cells. It is shown that layout-based technique is the only effective solution for identification of adjacent cells as netlist-based techniques significantly underestimate the overall SER.

- **Event-driven error propagation [15]**: For propagating errors generated in flip-flops, I integrated transient error propagation in an event-driven gate-level logic simulator which captures the combined effect of various masking factors. By carefully combining various generated errors at different cycles, in one pass all the error generation and propagation effects across all bits and all cycles are analyzed. This enables us to drastically reduce the runtime while maintaining the accuracy.
- **Emulation-based fault injection platform [16]**: I propose an easy-to-develop and flexible FPGA-based fault injection technique. This technique utilizes debugging facilities of FPGAs for fault injection in both flip-flops and memory units. As this technique uses FPGA built-in facilities, it imposes negligible performance and area overheads on the system. The experimental results show that this technique is on average four orders of magnitude faster than a pure simulation-based fault injection. These features make the technique applicable to industrial-scale circuits.
- **Fault injection acceleration [17]**: Due to the long residency of some errors in system states, the error has to be traced for even millions of cycles. However, only a very small portion of injected errors leads to the failure. This means that many emulation cycles are wasted as they contribute to no failure due to various masking effects. In order to address this issue, I present an importance sampling technique to identify the non-vulnerable time intervals in memory arrays and avoid unnecessary fault injections to speedup the soft error vulnerability evaluation process without sacrificing the accuracy. This approach expedites fault injection process by one order of magnitude.

1.2.2 SER Analysis of an Embedded Processor

By exploiting the cross-layer platform, I comprehensively analyzed the SER of an embedded processor with respect to several standard workloads. The results of this analysis [18, 19] reveal contribution of each memory array, flip-flop, and combinational gate to the overall SER. Those results also provide useful insights to designers for cost-effective protection of embedded processors.

My results invalidate several common assumptions on the SER contribution of different components of the processor. Our detailed analysis shows that despite the claim of [20] and [21] in which flip-flop SER is independent of clock frequency, it has an inverse linear relation with clock frequency. Furthermore, the contribution of the clock-tree network to the overall SER is small in this processor whereas the reset-tree SER is not negligible. Previous studies in [22] and [23] reported that the clock-tree contribution could be as large as 20% of overall flip-flops SER. However, because of the large size of clock driver cells, the probability of having soft errors in the clock-tree network is negligible.

1.2.3 Soft Error Mitigation Techniques

Considering the growing MBU issue alongside with the results of the comprehensive SER analysis, we proposed several solutions to mitigate multiple errors:

- **Reliability-aware placement algorithm [14]**: By performing a detailed analysis of multiple errors in random logic, I identified several multiple error mitigation opportunities which can be applied during the placement of physical design. These opportunities are exploited to locally adjust the position of cells in the layout based on the vulnerability of multiple errors affecting adjacent cells. The experimental results show that this technique is able to considerably reduce the overall SER without any area and performance penalty. This technique is built on top of an existing commercial place & rout tool and has a modest runtime and is scalable for industrial size circuits.

- **Low-cost error detection in embedded memories [24]**: The detailed SER analysis of the embedded processor reveals more potential for low-cost error detection in embedded memories which is required for error correction in write-through caches. I address this issue by proposing a low-cost error detection technique. In this technique, each data bit is associated with several parity bits in a way that the most common multiple error patterns could be detected by at least one of the parity bits. The experimental results reveal that the proposed technique reduces the area and power overheads of the stat-of-the-art solutions by about 30% for a given reliability constraint.
- **Low-cost error correction in FPGA configuration bits [25]**: Occurrence of soft errors in the configuration frames of an FPGA device permanently affects the functionality of the mapped design. Periodic configuration scrubbing combined with a low-cost error correction scheme is an efficient approach to avoid such a permanent effect. Existing techniques employ error correction codes with considerably high overhead to mitigate multiple errors in configuration frames. In this regard, I present a low-cost error detection code to detect multiple bit upsets in configuration frames as well as a generic scrubbing scheme to reconstruct the erroneous configuration frame based on the concept of erasure codes. The proposed scheme does not require any modification to the FPGA architecture. This scheme can detect 100% of multiple errors in the configuration frames with only 3.3% resource occupation while the recovery time is comparable with previous schemes.

1.3 Outline

The reminder of this thesis is organized in five chapters. A short introduction of each chapter is as follows:

- Chapter 2 summarizes the basic information on soft errors including their origins and existing models. It also shows the impact of technology scaling on soft error rate and main concerns in nanoscale technology nodes. Additionally, this chapter contains a survey of existing soft error estimation and mitigation techniques and a summery of remaining issues.
- Chapter 3 introduces our cross-layer soft error modeling approach. This chapter starts by describing the rationality behind our cross-layer approach. Afterwards, the error generation and propagation steps are explained in details and validation results for each step of the flow are presented. At the end, the soft error rate computation methodology based on our cross-layer approach is described.
- In Chapter 4, the results of SER analysis of an embedded processor with the proposed cross-layer approach is demonstrated. This detailed analysis shows the contribution

of each memory array, flip-flop, and combinational gate in overall SER. Moreover, the impact of selective protection techniques on the overall SER is studied. This chapter also presents effect of voltage and frequency scaling in overall SER.

- By identifying the shortcomings of existing techniques in protection of different parts of the processor, several low-cost techniques for protecting different parts of the processor in ASIC and FPGA implementations are proposed in Chapter 5.
- Finally, Chapter 6 concludes the thesis and points out potential direction of future research.

2 Preliminaries and State of the Art

Radiation-induced soft error was discovered four decades ago. From that time, it has been considered as a main reliability threat in integrated circuit. Although there is a rich literature of soft error analysis and mitigation techniques over the last 40 years (see Figure 2.1), due to the physical nature of this error, this threat evolves by rapid technology scaling. In older technology nodes, soft error was only a major concern in memory arrays of avionics systems, however, in advanced technology nodes, it is even important on random logic of consumer electronics. Moreover, in nanoscale technology nodes, it is highly probable that one particle strike affects multiple adjacent cells. These changes in the soft error behavior and the growing number of applications affected by this reliability issue increase the importance of this threat and necessitates continuous research on this topic in the future.

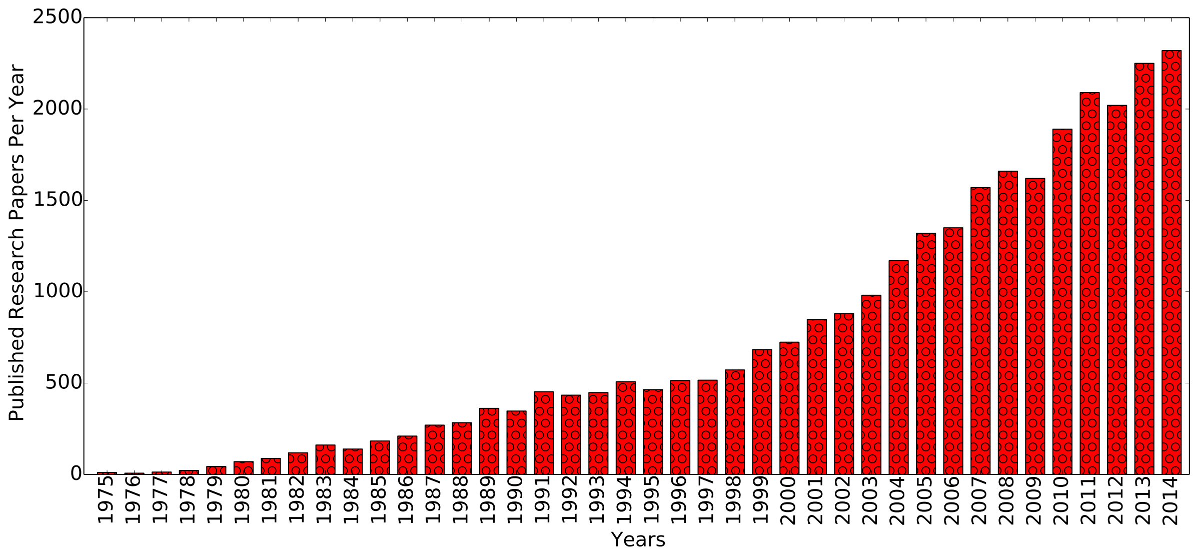


Figure 2.1: Number of research papers on soft error topic published annually

In this chapter, the basic knowledge on soft errors including origins, existing models, and evaluation metrics are introduced. Furthermore, the impact of technology scaling on SER is discussed. In addition, the existing soft error evaluation and mitigation techniques are reviewed.

2.1 Fault, Error, and Failure

The system *dependability* is defined as its capability to deliver services according to their specifications [26]. The acceptable quality of the service is completely dependent on the running application and user demand. For example, in video processing, the loss of one pixel might be acceptable as it is not recognizable by the end user while the loss of a complete frame could disturb the user. On the other hand, each and every bit of data is important in controller circuit of avionic systems.

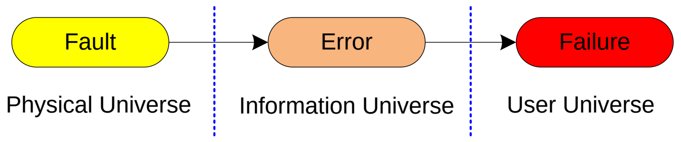


Figure 2.2: Fault, error, and failure chain

The *reliability* is a measure to show the continuous service accomplishment over the time. The reliability of the system is typically estimated by analyzing the fault, error, and failure chain (see Figure 2.2). The fault is a physical phenomenon which affects the system and causes some latent errors in the information stored in the system. The latent error could activate and propagate to other parts of the system and eventually lead to a failure in the delivered service. For instance, an unwanted change in the supply voltage of a memory array is a fault. This could disturb the stored content of that cell and cause a latent error. This particular memory cell might contain a valid data which will be read in future and cause an active error. The error also might degrade the quality of the delivered service and cause failure.

Faults are generally classified as transient and permanent [27]. A transient fault momentarily shows an undesired behavior, though there is no permanent damage. The common approach for dealing with a transient fault is to recover the system from erroneous state and continue using the faulty module. Radiation-induced soft errors, retention failures, and random telegraph noise are just few examples of transistor faults. Permanent faults are due to a permanent physical defect in the fabricated circuit and could be detected by repeating the same operational and environmental condition. Faults due to process variation as well as the manufacturing defects (short and open) are classified as permanent faults. This thesis is focused on transient faults and more specifically on radiation-induced soft error which is introduced in the next section.

2.2 Basics of Soft Errors

Radiation-induced soft errors are transient malfunctions due to interaction between energetic particles originating from packaging materials and cosmic rays. The impact of soft errors on the circuit is completely dependent on the type of affected component (i.e., memory cell, flip-flop, and combinational gate), and hence, different models are exploited to mimic the soft error impact on each type of components.

2.2.1 Origins of Soft Errors

Once a radiation-induced particle strikes a transistor device, it generates a track of electron-hole pairs on the device. These electron-hole pairs can be collected by the source and drain of the transistor (see Figure 2.3) and form a short unwanted pulse in the logical state of the transistor. This may eventually flip the logic value of a memory cell or the output of a combinational gate. It is worth to mention that this type of errors does not permanently damage the transistor and once the system is reset, the system will again function correctly.

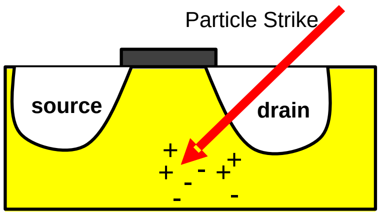


Figure 2.3: Generation of electron-hole pairs on a transistor device due to a particle strike

In terrestrial applications, alpha particles from packaging material and neutron particles from cosmic rays are two main sources of soft errors. Although the impact of these two types of particles on the circuit is similar, their error generation mechanisms are slightly different.

Alpha Particles

In 1978, Intel identified alpha particles due to the contaminated packaging material as main source of soft errors in their chips [28]. The main source of alpha particle-induced soft errors are the radioactive impurities in the chip manufacturing and packaging process such as solder bumps and semiconductor material. In particular, in the 3-dimensional integrated circuits where the solder bumps are very close to the active silicon devices, even the low-energy alpha particles could cause soft errors [29].

The kinetic energy of alpha particles is typically in the range of 4 MeV to 9 MeV. When alpha particles pass through a semiconductor device, they lost some of their kinetic energy due to the interaction with the transistor substrate, and hence, deposit a dense track of charge which eventually generates electron-hole pairs.

There are several approaches to mitigate the alpha particle-induced soft error rate. The first approach is to exploit materials with low radioactive compositions. The second approach is to develop barriers on the chip which can prevent the alpha particles from affecting the internal components. However, such approaches not only significantly increase the manufacturing costs and processing complexity, but also cannot completely eliminate all errors [7]. As a result, effective fault tolerant techniques are required to protect system against alpha-particle induced soft errors.

Cosmic Rays

In 1979, a series of experiment by IBM showed that the cosmic rays could also cause soft errors [30]. Although primary particles of the cosmic rays does not reach the surface of the Earth, they create a shower of secondary particles. The large portion of secondary particles capable of causing soft errors are energetic neutrons (approximately 95%) with the remainder composed of pions and protons [31]. Neutrons are uncharged particles, and cannot disturb a circuit on their own, but undergo neutron capture by the nucleus of an atom in a chip. This process may result in the production of charged secondaries, such as alpha particles and oxygen nuclei, which can then cause soft errors. This mechanism is called *indirect ionization* [7].

Cosmic ray flux is completely dependent on the altitude. For the common reference of New York City in USA, the flux is about 14 neutrons/cm2/hour. In the lower levels of the atmosphere, the flux grows by 2.2× for every 1000 m. As a results, digital circuits working on top of mountains experience an order of magnitude higher SER compared to those at sea level. Furthermore, SER in aircraft may be more than 300× higher than the sea level. This is in contrary to alpha particle induced soft errors which do not change with location.

The neutrons are the main source of radiation-induced soft errors in current technology nodes. However, pions and protons can also cause soft errors. In particular, with smaller device geometries in advance technologies, the direct ionization of protons will increase their contribution and they will become a new concern in soft error analysis.

2.2.2 Soft Error Modeling

Transient errors caused by a single particle strike in sequential elements (i.e., memory cells, latches, and flip-flops) and combinational gates are called *Single Event Upset* (SEU) and *Single Event Transient* (SET), respectively. With miniaturization of device geometries in nanoscale technologies, it is very likely that a high energy particle strike affects several adjacent cells in a circuit resulting in *Multiple Bit Upsets* (MBU) in sequential elements [32, 33, 34, 35] or *Multiple Event Transients* (MET)[1] in combinational gates [2].

MBUs/METs in older technology nodes could be safely ignored and therefore the detection and correction schemes were mostly concentrating on SEUs/SETs [3, 36]. The experimental studies show that the share of MBUs/METs in the overall SER grows with the technology downscaling [3, 11, 37] and it is the dominant contributor to the overall SER in the advanced technology nodes [12, 1]. In such scenarios, existing single error detection, correction and mitigation schemes are no longer effective and could lead to severe consequences such as undetected errors.

2.2.3 Masking Factors

It is known that the majority of the generated errors get masked due to circuit, architecture, and application-level masking factors and only a vary small fraction (typically below 5% [8, 9]) leads to failures. There are different masking factors associated with errors generated in combinational gates, flip-flops, and memory arrays which make the soft error analysis and mitigation more challenging.

Raditation-induced soft error in combinational gates generates a transient pulse at the output of the affected cell. This pulse might not be latched in the sequential elements (flip-flops or memory arrays) due to three circuit-level masking factors, namely logical, electrical or latching-window masking [38]. The logical masking happens when a transient pulse reaches to an input of a gate, but has no impact on the gate output as another input of the gate is in the controlling state (e.g. 0 for an AND gate). The electrical masking occurs when the transient pulse is attenuated due to the electrical properties of logic gates to the point that it does not reach to sequential element inputs. Furthermore, the latching-window masking happens when a transient pulse reaches the input of sequential elements outside their latching windows. Once the error is latched in sequential elements at the end of the error occurrence cycle, only logical masking factor can mask the error in the subsequent cycles.

[1]a.k.a. Single Event Multiple Transient (SEMT)

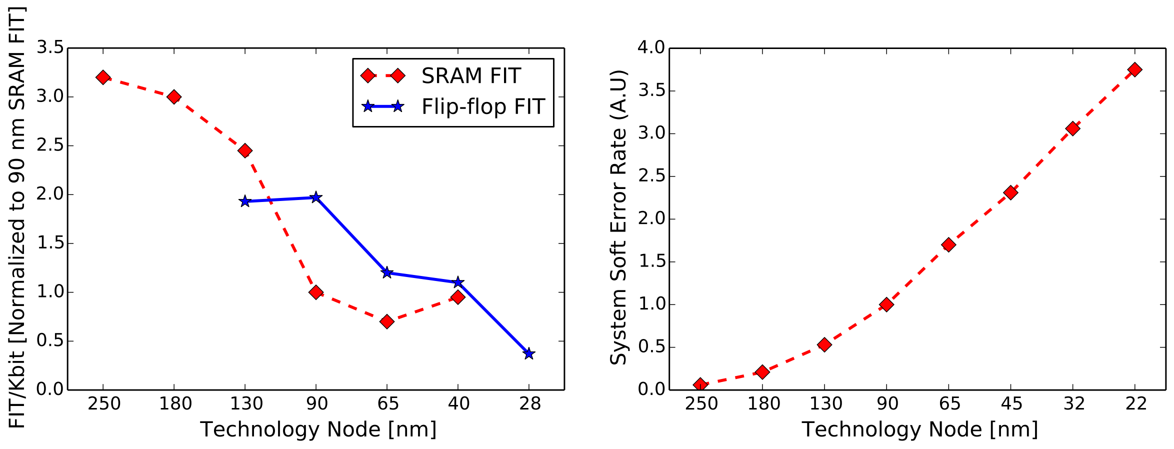
In a flip-flop, there are two factors which might prevent the error from propagating to down-stream flip-flops, namely logical and *temporal* masking. Logical masking is similar to the combinational gates. Temporal masking in flip-flops happens when the error occurs sufficiently late in the clock cycle and is not able to propagate to the downstream flip-flops before the latching window. Among these two masking factors, temporal masking is only important in the first cycle after error occurrence while logical masking has to be considered in the subsequent cycles as well.

In addition to the aforementioned circuit-level masking factors, an error might get masked due to architecture and application-level masking. For example, an error in write-through cache results in a cache miss and the data is reloaded from the lower memory hierarchy. As another example, an error in the least significant bit of an integer number could be masked by a division operation in the application-level.

2.2.4 Scaling Trend

As technology scales down, the size of transistors become smaller which in turn leads to less critical charge of cell (i.e. memory cell or standard cell). The critical charge specifies the minimum amount of energy required to upset the value stored in the cell. As a result, cell with smaller critical charge has higher vulnerability to radiation induced soft errors as larger spectrum of particles could affect that cell. On the other hand, smaller transistor size means smaller cell footprint. The probability that a particle strikes a cell is linearly dependent on its area. Therefore, the struck probability reduces as technology scales. Experimental results reveal that the decreasing struck probability dominates the smaller critical charge and the cell SER reduces by technology scaling [1]. This fact could clearly be seen for SRAM cell and flip-flop FIT rate in Figure 2.4.a.

Although the cell SER slightly reduces by technology scaling, the increasing number of devices integrated in a system (i.e. Moore's law) results in an exponential increase in the system SER [1, 3]. This fact is demonstrated in Figure 2.4.b. This exponential increase motivates SER analysis and mitigation in advanced technology nodes.



(a) Cell FIT Rate [1]

(b) System SER rate [3]

Figure 2.4: Impact of technology scaling on soft error rate

The increasing ratio of MBUs and METs in advanced technology nodes is another major concern which soft error analysis and mitigation techniques have to address in a cost-efficient manner. As shown in Figure 2.5, not only the ratio of MBUs, but also their sizes grow due to the smaller feature size. Furthermore, the experimental results show that the share of MBUs in 45 nm and beyond exceeds 50% [3, 39]. This mandates costly SER mitigation techniques such as multiple error correction coding schemes in memory arrays fabricated in nanoscales.

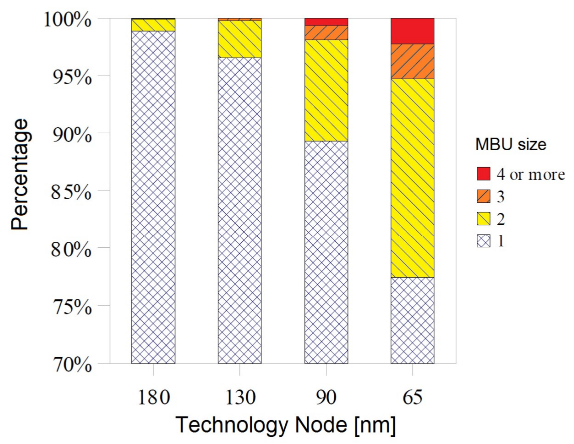


Figure 2.5: Impact of technology scaling on MBU ratio [1]

Experimental study of METs in real circuits is very challenging issue due to low observably of this components. Therefore, most of the research in this context is concentrated on the simulation results. There is only one radiation-testing experiment [2] on a simple inverter chain circuitry fabricated in TSMC 65 nm technology. The results of this experiment show that about 11% of errors affect more than one invertor. The most common MET patterns for that study are shown in Figure 2.6. According to simulation results [14, 40], the MET ratio will considerably grow by technology scaling.

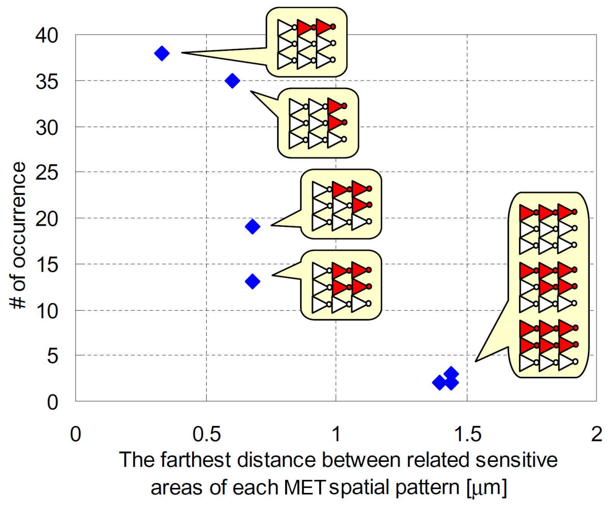


Figure 2.6: Common MET patterns in inverter chains fabricated in 65 nm [2]

2.3 Existing Soft Error Estimation Techniques

Existing soft error vulnerability estimation techniques could be broadly classified to radiation testing experiments, emulation-based fault injection campaigns, and analytical techniques at various abstraction layers. In this section, these approaches are reviewed by emphasizing on advantages and shortcomings for each category.

2.3.1 Radiation Testing

Radiation testing is commonly employed to estimate the SER. In this approach, the device under test is exposed to radiation source which reproduces the desired radiation environment (e.g. terrestrial). However, the intensity of the artificial source is much higher than the real environment that device has to work in. Therefore, radiation testing is able to provide useful data in a considerably short time (i.e. only few hours or days) and the obtained results have to be extrapolated to obtain the SER under real condition.

Although radiation testing is a common industrial approach, it requires a manufactured device prototype as well as expensive testing facilities. Therefore, it is not suitable for SER estimation during design stage. Furthermore, radiation testing techniques cannot accurately determine the contribution of each components. Indeed, there are several studies investigating the relative contributions of sequential and combinational SER based on the results of radiation testing which use simple test structures such as inverter chains, comparators, and shift-registers [20, 21, 22, 23, 41]. Such experiments reveal that the contribution of combinational logic is increasing with every new technology node and also has a linear relation with the circuit frequency. The major issue with these experiments is the simplicity of the test structures. Although such experiments give insight into the relative SER of different components, they are not representative for complex circuits. In fact, during radiation testing experiments on complex circuits, it is difficult to differentiate between the sequential and the combinational contribution.

2.3.2 Emulation-based Fault Injection

The de facto approach to estimate the soft error vulnerability of systems in early design stage is statistical fault injection. In this approach, faults are injected in different bits of the target component at random execution cycles and their effects are tracked for millions of cycles of application execution. This approach could be done in both simulation and emulation platforms, however, the emulation-based fault injection is 3-4 orders of magnitude faster than simulation-based one. Therefore, we only concentrate on emulation-based techniques.

Emulation-based[2] fault injection techniques are divided into reconfiguration-based [42, 43, 44, 45] and instrumentation-based [46, 47, 48] techniques. Reconfiguration is a built-in facility of FPGAs and can be used during compile time or run-time to change the configuration bits of the FPGA device in such a way that the preferred fault model is injected in the desired location. A technique based on the *Compile-Time Reconfiguration* (CTR) for injecting stuck-at fault model is presented in [42]. In this technique, for each stuck-at fault, circuit is re-compiled and during the compile-time, configuration bits of FPGA is changed in a way that

[2]a.k.a. FPGA-based

the desired node is connected to the ground or VDD. In order to decrease the intractable runtime of the CTR technique, several FPGA-based fault injection techniques have employed the *Run-Time Reconfiguration* (RTR) mechanism. The RTR-based technique presented in [43] covers transient and permanent faults in combinational parts and SEU fault model in flip-flops. Another RTR-based technique for injecting a wider set of fault models appearing in deep sub-micron technology is presented in [44].

In instrumentation-based techniques, a fault injector circuit named saboteur is added to each fault site to produce the preferred fault model. These techniques use a non-commercial tool to add saboteurs to *Design Under Evaluation* (DUE) and to create a modified DUE with built-in saboteurs for fault injection purpose. Then, the modified DUE is programmed to an FPGA device and faults are injected in desired fault sites by activating their corresponding saboteurs. Saboteur circuits for stuck-at [46, 49], SET [49], SEU [47], and bridging [48] fault models are proposed in the literature.

Recent experiments shows that the contribution of combinational logic to overall SER is remarkable and in high frequency systems, it might exceed that of flip-flops [20, 21, 19]. As a result, it is essential to evaluate the vulnerability of the system against SETs. Although there are several techniques for injecting SEUs and MBUs, SET injection is much more complicated for two reasons. First, in FPGA platforms creating a transient pulse with a small width is impossible; second, combinational logic structure changes when a design is programmed into an FPGA device and as a result electrical masking and logic masking factors are completely different from the ASIC implementation due to the delay-dependent nature of SET effects. A hybrid simulation-emulation SET injection technique presented in [50]. In this technique, simulation is employed for propagating the SET from the fault site to flip-flops/latches and emulation is then performed to study the resulting single or multiple-bit upset. A voltage-time quantization technique presented in [51] divides each clock cycle into very small time intervals and values of each signal during different time intervals are stored in a shift register. This technique rounds the delay of each gate to the nearest time interval length and uses the shift registers for SET propagation. This technique is efficient for modeling SET with large pulse width (larger than FPGA minimum clocking possibly) in very small combinational circuits. A multi-level technique proposed in [52] uses the quantization technique for propagation of SET from fault site to flip-flops and then an ordinary RTL emulation is performed to propagate the fault in subsequent clock cycles. Since all operations are performed inside FPGA, this technique is much faster than simulation-emulation technique while it provides less accurate results.

2.3.3 Analytical Techniques

Circuit/Logic-level Analytical Error Propagation Techniques

In analytical SER estimation technique at circuit or gate level, the SER of a circuit is estimated by employing analytical models such as binary decision diagrams and error probability propagation. In such technique, instead of using time-consuming fault injection or fault simulation, the masking factors are estimated by statistical analysis of a circuit using probabilistic models.

a) BDD/ADD-based techniques: The technique presented in [53] employs *Binary Decision*

Diagrams (BDDs) to compute the logical masking of combinational circuits. This technique is based on static SER analysis as it relies on enumeration of input vectors. Since the worst case complexity of BDD encoding algorithm is exponentially proportionate with the number of primary inputs [54], a heuristic algorithm has been proposed to decrease the number of inputs by partitioning the circuit [53]. The main source of inaccuracy in this technique is disregarding mutual masking of reconvergent glitches. This problem has been addressed in [55]. Unlike BDD-based techniques that can only handle uniform input distributions, the technique presented in [55] uses *Algebraic Decision Diagrams* (ADDs) to model arbitrary input distributions. A SER computation framework based on *Probabilistic Transfer Matrices* (PTMs) using ADDs is presented in [56]. Despite its accuracy in SER estimation, this technique suffers from large amount of computation and memory consumption as it is based on matrix multiplication. An ADD-based technique, called *Probabilistic Model Checking* (PMC) is proposed in [57]. In this technique, a circuit is specified as a *Discrete-Time Markov Chain* (DTMC) and the reliability of the circuit is evaluated by computing the probability of reaching specific DTMC states. Similar to PTM, this technique has also excessive memory requirements. The excessive memory requirement of PTM and PMC techniques has been addressed in [58]. Although BDD/ADD-based techniques have acceptable accuracy level, the runtime of these techniques is challenging for circuits with the overall number of primary inputs greater than 50 [59].

b) Error Probability Propagation (EPP)-based techniques: Techniques proposed in [60, 61, 62, 63] introduce propagation rules for propagating error from error site toward primary outputs and sequential elements. 4-value logic (0, 1, a, and $\bar{a}$) presented in [60, 61] to propagate the error through gates. This technique has addressed the reconvergent glitches in the forward cone of an error site, while correlations between error-free and erroneous signals are ignored. This technique is very fast and its runtime is from the order of $O(n^2)$, where n is the total number of gates in the circuit. Recently proposed technique in [62] replaces each gate with its equivalent *super gate*. In super gates, there is a virtual error signal for each signal that is used during error propagation calculations. This technique considers the correlation between error-free and erroneous signals, which has been ignored in [60] and, as a result, its inaccuracy is less than 4-value EPP. However, since it considers the correlation between all signals, its runtime is from the order of $O(n^3)$.

Architecture-level Soft Error Vulnerability Analysis

Architecturally Correct Execution (ACE) [64, 10] is a fast approach for investigating the susceptibility of the memory arrays to soft errors. In this approach, the lifetime of each bit in a memory array is divided into *ACE* and *un-ACE* intervals. The time intervals that a transient error does not affect the output of the running workload is un-ACE, otherwise it is ACE. In the ACE analysis, all the intervals not proven to be un-ACE are assumed to be ACE [10], therefore, the susceptibility of memory arrays could be significantly overestimated (by about 10X [65, 66]). Although this approach has a considerably shorter runtime which is almost equal to that of just one fault injection in a simulation environment, the significant overestimation limits its application on the SER estimation.

The generic ACE model, which only requires the information in the memory boundary (i.e. read and write access), is applicable to any memory array. This ACE model is explained with respect to the simple example illustrated in Figure 2.7. This figure shows an example of serial accesses to a particular memory entry during the execution of a workload. In this model, all time intervals which lead to a write access are un-ACE as the write access overwrites the possible error. In contrast, an error in a time interval leading to a read access is always propagated to the memory output and has a chance to impair the final output of the workload. Hence, such intervals are assumed to be ACE.

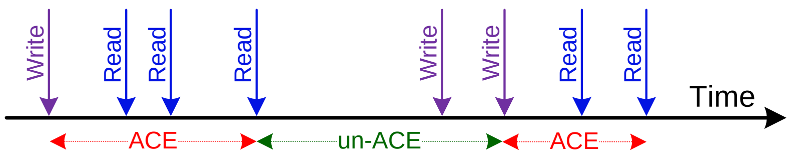


Figure 2.7: ACE and un-ACE intervals in the employed ACE analysis

Beside the generic ACE model, there are also more advanced techniques which exploit the architectural information to identify more non-vulnerable time intervals, and hence provide less pessimistic results. Such techniques leverage the specific behavior of each memory component such as data characteristics or access patterns to further reduce the ratio of ACE intervals to the entire execution time. For instance, there are dedicated techniques for data cache [67, 68], instruction cache [69], L2 cache [70], and register-file [71] which are less conservative than the generic model. However, there is still a considerable gap between the failure probability obtained from fault injection and ACE analysis results (7X [65]).

2.4 Existing Soft Error Mitigation Technique

This section reviews existing soft error mitigation techniques for random logic and memory arrays. Actually these components have inherently different mitigation techniques due to their different structures.

2.4.1 Soft Error Mitigation in Memory Arrays

In order to increase the density and performance, memory arrays (such as SRAM and DRAM) are typically designed with minimum size transistors. Hence, the critical charge required to upset the cells in these components are considerably smaller than that of the logic elements (i.e. flip-flops and combinational gates) [36, 72]. Memory arrays are the major contributors to the overall system SER [1, 37] as they typically constitute a large percentage of the chip area (approximately 50%-80% [73, 74]) and are more vulnerable to the soft errors. Thus, applying cost-efficient SER mitigation techniques on memory arrays is of decisive importance.

Error Correction Techniques

Applying *Error Correction Codes* (ECCs) uniformly on all memory entries is a common approach to mitigate soft errors in memory arrays. In this approach, an ECC is appended to each memory entry and during each write and read operation, the parity computation and the error correction are performed, respectively. The most common ECC technique is a Hamming code which could perform single erroneous bit correction and double erroneous bit detection (SEC-DED). This approach is commonly implemented in nowadays commercial processors [75, 76].

By increasing the MBU rate and size in the advanced technology nodes, an effective technique has to correct more number of bits in one entry. Although in theory the cost of error correction grows linearly with respect to the number of erroneous bits [77], several techniques exploit the locality of erroneous bits in MBUs for low-cost multiple error correction. Recently, error correction techniques for correcting double [78, 79, 80] and triple [81] adjacent errors are introduced.

Built-In Current (BIC) sensors could be also exploited to correct MBUs [82, 83]. Such sensors can identify the erroneous column(s) and then the error could be corrected by detecting the erroneous word(s) according to a word-wise error detection coding. Although such a technique is quite efficient for correcting small MBUs, it cannot correct MBUs when multiple bits in one column are affected [83].

Another approach to correct large MBU incidents is to isolates the error detection and correction. This kind of cost-efficient techniques [84, 85, 86, 87] employ a low-cost error detection technique to identify the location of the errors and then correct the errors by another set of redundant data. Although these techniques provide considerably higher error correction coverage than other techniques, their recovery time are typically high as all/several memory entries have to be accessed to compute the correct contents of the erroneous entries.

Error Detection Techniques

Although the ultimate objective of MBU mitigation techniques is to correct the errors, in some cases achieving the same objective requires a low-cost MBU detection technique. MBU correction in write-through cache by detecting and invalidating the erroneous word [88], and the error detection technique required as part of some correction techniques [82, 83, 84, 85, 86, 87] are just few representative examples of such cases.

The most common technique to mitigate MBUs is to physically interleave several words in a way that each MBU affects one bit of several words rather than several bits in the same word. In case the physical interleaving is costly due to the aspect ratio issue or excessive overheads, the virtually interleaved parity is a widely used technique to detect MBUs [84, 83, 85]. In this technique, data bits are divided into several parity groups in a way that the data bits belonging to different parity groups have constant distance from each other and are associated to one parity bit.

An interesting approach to optimize the parity organization of the virtually interleaved parity based on AVF of different memory columns is presented in [89, 90]. This technique results in different parity organization for each memory array. However, obtaining accurate AVF value for each memory column requires a detailed analysis for each column of every memory arrays which could be very time consuming. Furthermore, AVF values and hence the obtained parity organization, are highly dependent on the running workloads which not only change during runtime, but also are not known during the design phase.

2.4.2 Soft Error Mitigation in Random Logic

Hardened latches [and flip-flops] preserve their value even if a particle strike affects some of their internal nodes. Various hardened latch designs have been proposed in the literature [91, 92]. The main disadvantage of these cells are their excessive area and power overheads as they require particular transistor sizing. Due to the sizing issue, these cells do not scale in the same pace as the device size and overheads grow by technology downscaling. In contrast to these cells, *Dual Interlocked storage CEll* (DICE) [93] and its successors [94, 95] achieve immunity against any error affecting only one of the internal nodes with any specific transistor sizing. These cells could be implemented in any technology using minimum size transistors. As detailed in [96, 97], due to shared diffusion and multiple node disruption in advanced technology nodes, the residual SER in SEU tolerant latches is not zero as an error could simultaneously affect several nodes. In recent years, several latche designs [98, 99] have been proposed to address this issues. These cells could tolerate multiple upsets in their internal nodes at the cost of more area and power overheads.

SET mitigation in combinational logic is a challenging issue mainly due to the analog nature of SETs. One approach is to reduce the error generation chance by modifying the process [100], layout hardening [101, 102], and gate-sizing [103]. Another approach it to reduce error propagation probability by inserting transient filters [104] or by exploiting redundancy in data path [105, 106].

2.5 Summary

Radiation-induced soft error originated from cosmic rays or packaging materials is a major reliability concern. The experimental studies reveal that the system SER exponentially grows by technology scaling. At the same time, due to smaller device geometries, the share of MBUs/METs increases. Therefore, the effective SER analysis and mitigation techniques have to take these kind of errors into account.

There have been various SER estimation techniques in the literature. Table 2.1 summarizes the main features of the different SER estimation approaches. While the objective is to compute the SER of each component, the majority of the techniques are able to either compute FIT or AVF, failing to capture the entire picture. Hence, such techniques cannot quantitatively compare the SER of different components (e.g. floating point unit vs. register-file). Radiation-testing is the only approach which could model both FIT and AVF in a combined manner, however, due to sophisticated error propagation behavior of complex systems and low observability, it cannot identify the origin of the error, and hence, the contribution of each component to the overall SER. Such information could assist designers toward low-cost SER mitigation.

Table 2.1: Comparison of main features of existing and proposed study

	FIT	AVF	Overall SER	Determine Impact of Each Cell
Device-level FIT Rate Analysis	Accurate	No	No	No
Emulation-based Fault Injection	No	Accurate	No	No
Analytical Single-cycle and Multi-cycle Error Propagation	No	Either Inaccurate or Slow	No	No
Analytical ACE Analysis	No	Inaccurate	No	No
Radiation Testing	Accurate	Accurate	Accurate	No

While there have been various techniques for low-cost MBU correction techniques in memory arrays, the MBU detection is mainly limited to either physical or virtual interleaving.

Furthermore, there have been various cell- and circuit-level techniques for protecting random logic against soft errors, however, these are mostly concentrated in single error assumption which is no longer valid in nanoscales.

3 Cross-layer Soft Error Modeling

While soft errors are generated at device-level, they could propagate and impair the output of the running workload. As a result, it is important to evaluate the impact of this kind of errors using a combined knowledge of various layers in a cross-layer approach. Development of a fast and accurate soft error analysis requires a detailed information about underlying technology, functionality of the RTL code, and also the running workload characteristics. Among these, the underlying technology is only important in error generation phase as well as the first cycle of error propagation. In addition, the error generation phase is independent of the functionality of the design. Considering such dependencies, as shown in Figure 3.1, the error generation and propagation phases could be divided into three phases: error generation at device-level, error propagation in first cycle, and error propagation after first cycle.

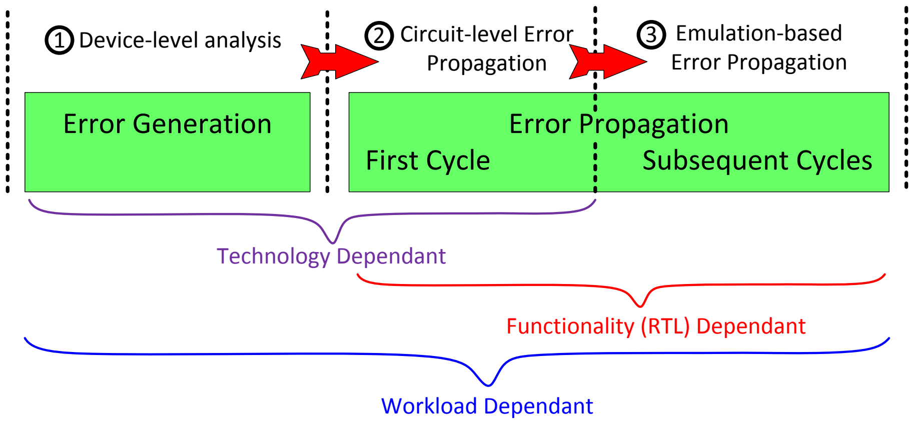


Figure 3.1: Reasoning behind proposed cross-layer soft error analysis platform

The proposed cross-layer platform is shown in Figure 3.2. For error generation, all the cells were analyzed using an accurate SER analysis tool to obtain the raw FIT rates. Also, MET error sites are extracted according to the layout information. Then, by employing an accurate analytical model considering electrical, logical, and latching-window masking, errors are propagated from combinational logic towards flip-flops and memory units. In addition, temporal masking of each flip-flop is determined using a novel event-driven error propagation approach according to the post-layout timing information. Finally, an emulation-based tool is employed to inject errors in flip-flops and SRAM arrays. This tool is able to propagate errors and determine their effect (masked, latent, failure) with respect to an actual workload. Then, the SERs of all components are computed based on the results obtained during these these steps.

The rest of this chapter is organized as follows: Section 3.1 describes the device-level FIT rate analysis tool. Section 3.2 introduces our proposed layout-level multiple transient error site identification method for random logic. The single-cyle circuit-level analysis for flip-flops and combinational gates are presented in Section 3.3 and Section 3.4, respectively. The proposed

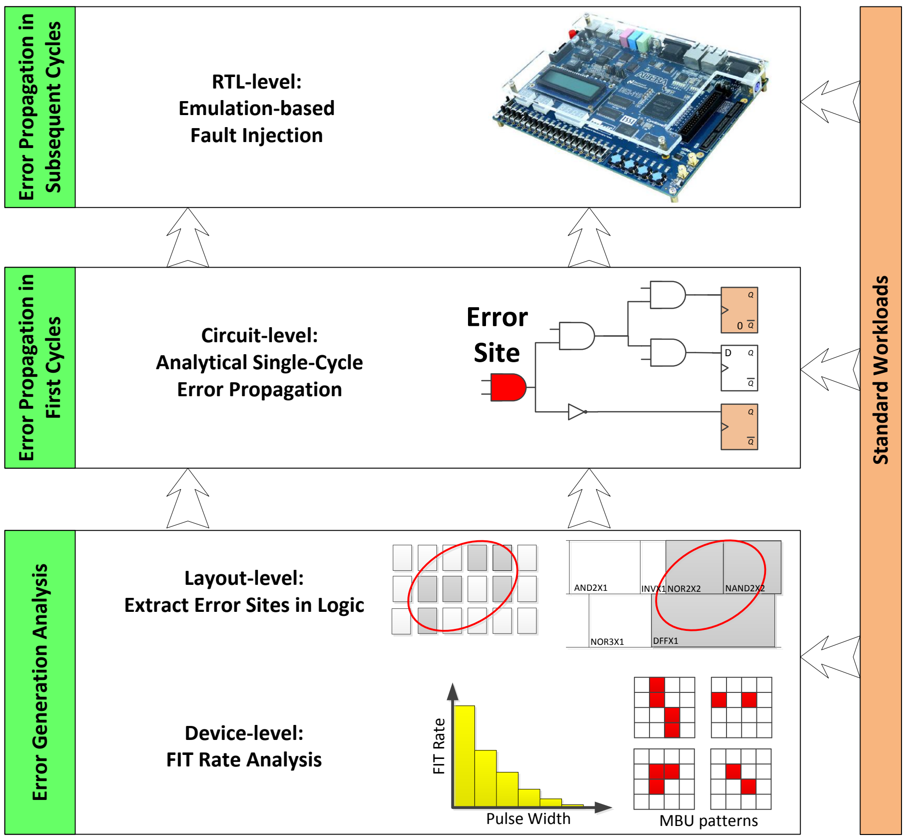

Error Propagation in Subsequent Cycles
RTL-level: Emulation-based Fault Injection
Error Propagation in First Cycles
Circuit-level: Analytical Single-Cycle Error Propagation
Error Site
Error Generation Analysis
Layout-level: Extract Error Sites in Logic
AND2X1
INVX1
NOR2X2
NAND2X2
NOR3X1
DFFX1
Device-level: FIT Rate Analysis
FIT Rate
Pulse Width
MBU patterns
Standard Workloads

Figure 3.2: Rationality behind proposed cross-layer soft error analysis platform

emulation-based platform is explained in Section 3.5 followed by a novel importance sampling technique for accelerating fault injection experiments in Section 3.6.

3.1 FIT Rate Analysis at Device-level

The first step of soft error rate estimation is to characterize the intrinsic FIT rate of each library cell under multiple load conditions. This is done using an accurate FIT rate analysis tool [107]. This tool uses a process response model which is generated based on a set of device-level simulations. Using this response model, the tool is able to characterize the sensitivity of standard cells by simulating a series of current pulses in a SPICE netlist of the cell, taking into account the physical layout. The layout information is used by FIT rate analysis tool to identify which transistors in the standard cell are physically adjacent. The adjacency information is exploited to consider charge sharing effects which can result in multiple current sources to be added to the SPICE netlist to model the fault. The tool also accesses a nuclear database which enables it to compute the interactions between neutrons and the atoms in the device, the secondary particles which are produced, and the distribution of generated current pulses. The accuracy of this tool for raw FIT rate analysis has been validated on several commercial processes.

For combinational gates, this tool provides the distribution of pulses with various widths. This distribution is dependent on the output load seen by the cell, thus each cell is characterized for multiple loads. During the full circuit analysis, for each gate, based on its load, an interpolation is performed between the two closest load conditions that were simulated for the given cell type. In order to accelerate the analysis flow in the next phases, the pulse width distribution curve is discretized into a few pulse widths with distance of d. Smaller values of d result in higher accuracy but larger runtime. Therefore, an appropriate value for this variable must be selected considering the trade-off between runtime and accuracy.

Figure 3.3 shows the pulse width distribution of few gates in Nangate 45 nm standard cell library for a typical output load of 3 fF and at a supply voltage of 1 V. From this figure, it becomes clear that Alpha particles mostly generate very short pulse widths which are very likely to be electrically masked. On the other hand, Neutron-induced transient pulses have considerably longer widths which thus have a higher chance to be latched in the downstream flip-flops.

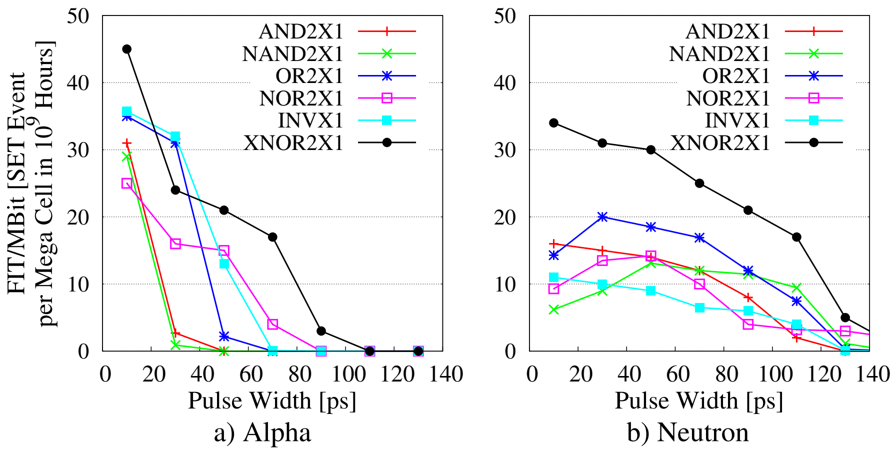


Figure 3.3: Pulse width distribution for few standard cells from Nangate 45 nm standard cell library for output load of 3 fF and supply voltage of 1 V

In the case of flip-flops and latches, this tool reports the SEU rate. For single-port and multi-port SRAMs, the tool provides SEU and MBU rates as well as MBU patterns and their occurrence probability. The Neutron and Alpha-induced FIT rate of different components in a 45 nm technology are reported in Table 3.1. As shown, for a single-port SRAM cell which is employed for caches and their tags, almost half of the Neutron-induced soft errors are in the form of MBU, while the ratio is considerably smaller for alpha particles. A similar trend could be seen for dual-port SRAM cell which is used to implement the register-file. For the case of flip-flops, we reported the range of FIT rate as there are different types of flip-flops in Nangate 45 nm standard cell library. Moreover, both SRAMs and flip-flops are more sensitive to Alpha particles in this technology node.

Table 3.1: Neutron and Alpha-induced FIT rate of different components in a 45 nm technology in terms of FIT/Mbit per billion hours of operation

		Neutron	Alpha	Overall	
Single-port SRAM cell	SEU	150	351	501	693
	MBU	140	52	192	
Dual-port SRAM cell	SEU	211	448	659	867
	MBU	160	48	208	
Flip-flop		119-217	181-216	300-435	

The FIT rates of different MBU sizes for Alpha and Neutron-induced errors are demonstrated in Figure 3.4. In addition, some of the most common MBU patterns and their contribution to the overall FIT rate are reported in Figure 3.5.

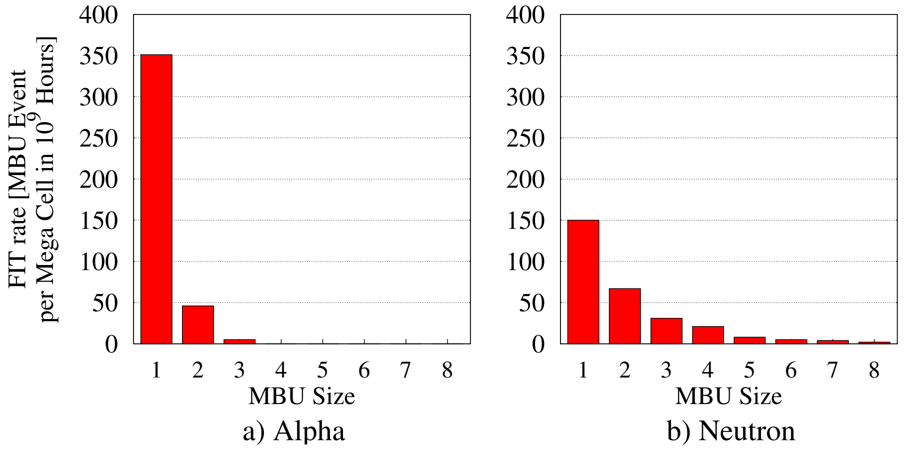


Figure 3.4: MBU size distribution for Alpha and Neutron particles in a typical 45 nm single-port memory

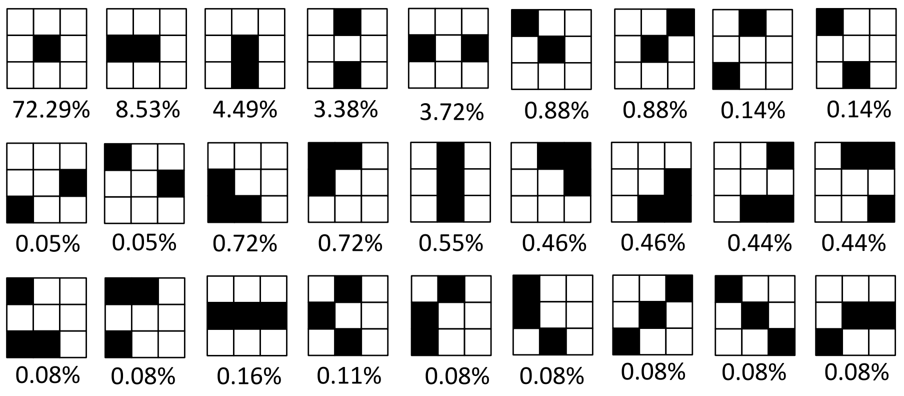


Figure 3.5: Most common MBU patterns for Alpha and Neutron particles

3.2 Multiple Transient Error Site Identification using Layout Information

As mentioned in Chapter 2, with miniaturization of device geometries in nanoscale technologies, it is very likely that a high energy particle strike affects several adjacent cells in a circuit resulting in MBU in sequential elements or MET in combinational gates. Although the FIT rate analysis tool is able to accurately model MBUs, due to the irregular structure of random logic, it cannot analysis METs as it requires a detailed information about the location of cells in the circuit layout. Given the fact that a considerable fraction of particle strikes in random logic results in METs [2, 40], their impacts at the layout and logic levels must be accurately modeled.

The analytical techniques presented in [108] and [109] address the MET fault model in logic circuits. The EPP-based technique presented in [108] propagates the error probabilities from the error sites towards both primary outputs and sequential elements. The technique presented in [109] is based on BDDs and provides more accuracy at the expense of runtime, compared to the earlier method. The major shortcoming of these techniques in estimating SER due to METs is that they use logic-level netlist for identification of MET error sites, neglecting layout-level adjacency of error sites. Such assumption can significantly underestimate the circuit SER as we will experimentally demonstrate later in this section. Additionally, the distribution of affected error sites and the number of affected cells completely depend on the layout-level details and cannot be extracted from the logic-level netlist. For example, our experiments show that a considerable fraction of particles simultaneously affects both combinational gates and flip-flops, which is completely ignored in previous techniques. Ignoring such cases can further increase the SER inaccuracy. In the rest of this thesis, the occurrence of multiple transient errors at sequential elements, combinational gates, or combination of them is called *Multiple Transients* (MT).

In this section, a fast and accurate technique MT error site extraction technique is proposed. In the proposed technique, the surface affected by a particle in combinational and sequential logic is estimated with oval shapes obtained from available MBU patterns in memory arrays. Considering MBU patterns occurrence probability, ovals are randomly placed at different locations inside the circuit and the list of affected cells are extracted.

Analysis of ISCAS'89 and ITC'99 benchmark circuits reveal that less than 10% of the netlist adjacent cells are physically adjacent in the layout. Also, more than 60% of physically adjacent cells are not adjacent in the netlist. Experimental results show that neglecting layout adjacency can cause inaccuracy up to 36.04% in the circuit overall SER.

3.2.1 Motivation for Layout-based MT Analysis and Mitigation

An important step in MT analysis is identifying physically adjacent cells as error sites. Although the layout of the circuit is necessary for the accurate identification of adjacent cells, the previous MT analysis techniques presented in [108] and [109] employ some heuristic approaches to extract the list of adjacent cells from the netlist. In these techniques, four categories including a gate and its fan-in (GFI), a gate and its fan-out (GFO), common fan-ins of a gate (CFI), and common fan-out of a gate (CFO) are considered as adjacent nodes for MT error sites. Fig. 3.6 shows several examples of netlist-based adjacencies. In these techniques, a gate is first selected as primary error site and then its MT pair are randomly selected among its netlist adjacent cells.

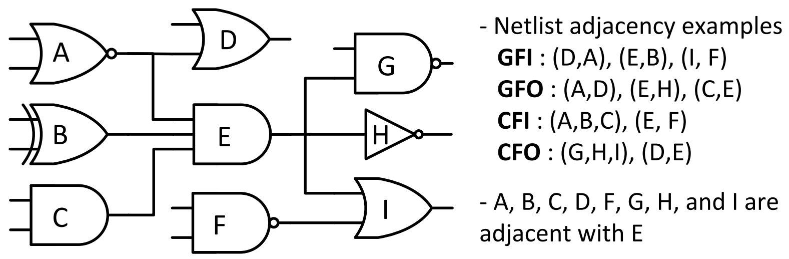

A
B
C
D
E
F
G
H
I
- Netlist adjacency examples
GFI : (D,A), (E,B), (I, F)
GFO : (A,D), (E,H), (C,E)
CFI : (A,B,C), (E, F)
CFO : (G,H,I), (D,E)
- A, B, C, D, F, G, H, and I are adjacent with E

Figure 3.6: Examples of different adjacency scenarios considered in the netlist

In order to check the accuracy and layout-relevance of this model, i.e., extraction of adjacent cells from the logic netlist, the layout of several circuits selected from ISCAS'89 and ITC'99 benchmarks synthesized with respect to Nangate 45 nm have been comprehensively analyzed (the details of this analysis framework is provided in Section 3.2.3). In this regard, all possible adjacent pairs for different netlist adjacency categories are first extracted from the netlist and then the physical adjacency of each pair in the circuit layout is investigated. Since the order of adjacent pairs is not important in this investigation, both GFI and GFO categories are equal and are assumed as a single category called GFI/GFO. In the example given in Figure 3.6, E is the fanout of gate B and gate B is a fan-in of gate E. As a result, pair (B,E) belongs to both GFI and GFO categories. Figure 3.7 shows the results of this experiment. As it can be seen, on average, only less than 10% of netlist adjacent pairs are also adjacent in the circuit layout. Also, the probability of physical adjacency of GFI/GFO category is much higher than that of CFI and CFO categories.

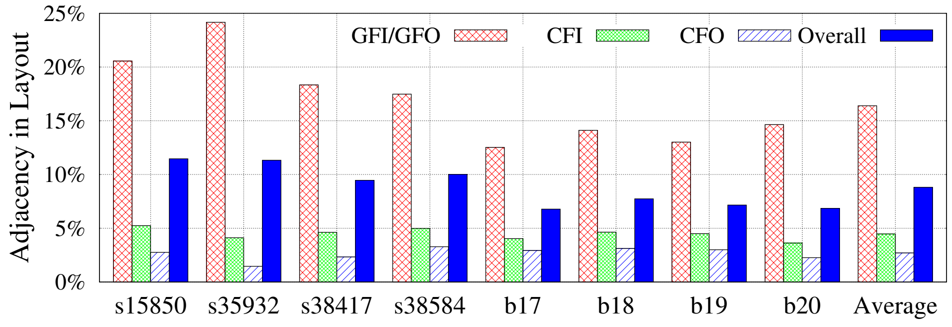


Figure 3.7: Relevance of different adjacency scenarios considered in the netlist to layout adjacency

There is another experiment conducted in which all layout adjacencies on the circuit layout are first extracted and then for each physical adjacency, its netlist adjacency category is investigated (Figure 3.8). It can be inferred from Figure 3.8 that more than 60% of the physical adjacencies do not belong to the previously defined netlist adjacency categories.

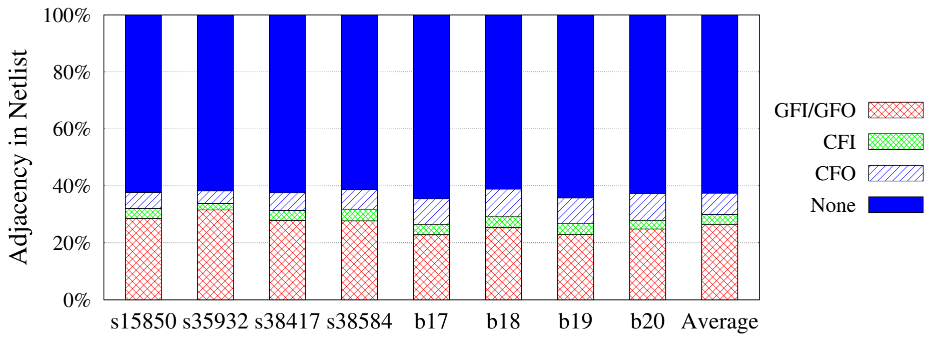


Figure 3.8: Relevance of extracted adjacencies from layout to netlist-level MT error models

From these two experiments, it is clear that there is no statistically meaningful correlation between netlist and layout adjacencies. This means that for an accurate MT analysis, netlist-level analysis for MT error site abstraction is not sufficient and layout-level analysis must be performed. To address this issue, we propose a fast and accurate layout-based MT modeling technique.

3.2.2 Proposed Layout-based MT Error Site Extraction Technique

MT Error Site Extraction Using MBU Analysis

The first step for accurate MT modeling is extracting physically adjacent error sites from the circuit layout. This requires to have MT patterns projected in the circuit layout and their occurrence probabilities. To the best of our knowledge, no field results about MT patterns on combinational and sequential logic have been reported in the literature. This could be mainly due to the fact that the logic gates and sequential elements (such as flip-flops) have significantly less observability as compared to regular structures such as SRAM-based memory arrays. Since memory arrays are more regular and dense than logic structures and also have a full observability, the affected area can be accurately estimated.

In the proposed technique, we try to use available MBU patterns in memory arrays for identification of MT error sites in logic circuits. In this method, it is assumed that the surface affected by a particle strike is almost the same for the memory arrays and logic cells in different technology nodes. In order to support this assumption, the average affected area of SRAM for Neutron-induced particle strikes across different technology nodes are reported in Figure 3.9. These results are extracted according to the cell area and MBU size information reported in [3]. Basically SRAM cells act as a guideline for determining the affected area and smaller cell size could provide more accurate results due to high resolution in estimating the area. As it can be seen from this figure, the estimated affected area reduces from 250 nm to 90 nm which is mainly due to larger SRAM cell size, and hence, less resolution in the area estimation. However, the estimated affected area is relatively constant from 90 nm to 22 nm, independent of the cell size. This clearly shows that the affected area is independent of the underlaying SRAM technology. Hence, it could be conclude that the affected area would be also the same for the irregular logic circuits.

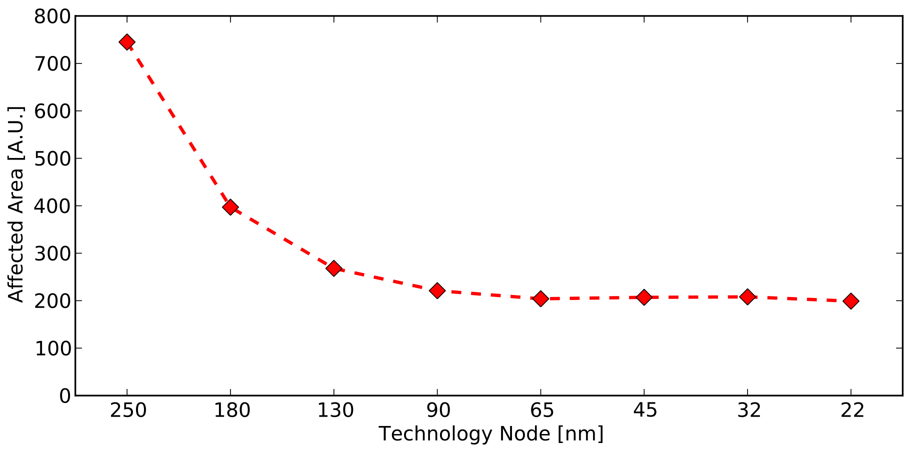


Figure 3.9: Average affected area for Neutron particles across different technology nodes based on the cell and MBU size information reported in [3]

The main idea behind our work is to extract the affected area in a memory array and then consider all logic cells covered by a similar surface to be MT error sites. In this regard, the affected area for each MBU pattern is first extracted. Predominant MBU patterns in memory arrays have been comprehensively studied using neutron beam-based accelerated SER estimation [32, 33, 110]. The number of affected bits by a single strike depends on several

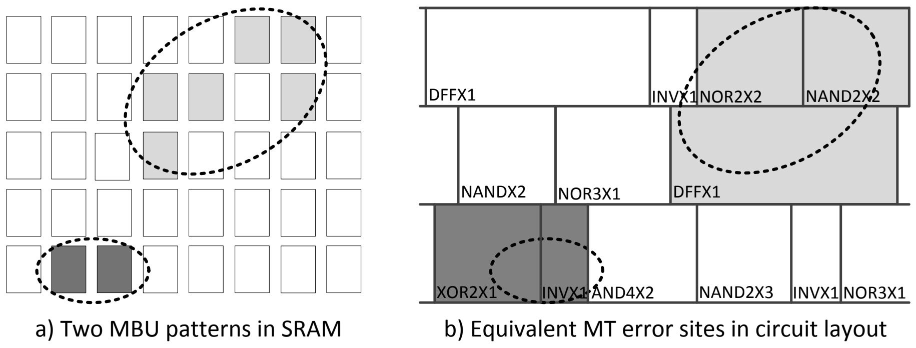


Figure 3.10: Extraction of MT error sites from existing MBU patterns

parameters including particle type and its energy, strike angle, cell type, cell size, and output load [34]. Given memory cell dimension as well as vertical and horizontal distance between adjacent cells, for each MBU pattern the surface affected by the strike can be calculated. Most of the MBU patterns, especially MBUs with more than eight bits, can be effectively covered by an oval surface. As a result, in our approach as shown in Figure 3.10, all cells affected by an MBU are first surrounded by an oval. Then, during SER estimation the same oval is transferred to random locations inside the logic layout and all cells affected by the MT are listed as error sites.

Library Characterization

In the circuit layout, only a subset of cells which have an overlap with the oval surface are considered as error sites. A cell has overlap with an oval surface if at least one of the *sensitive zones* to soft errors (N-diffusion and P-diffusion) falls within the surface. In fact, when a particle strikes a cell, it causes an additional charge to be collected in the diffusion parts of the transistor which in turn disturbs the normal operation of the transistor. The charge collected to the diffusion parts which are connected to VDD and GND pins is evacuated and does not affect the circuit behaviour. The other parts of the diffusion can be disturbed by collection of additional charge. Figure 3.11 shows how sensitive zones for a NAND gate layout are extracted based on this explanation. Using this approach, all cells inside the technology library are characterized and sensitive zones coordinations are extracted.

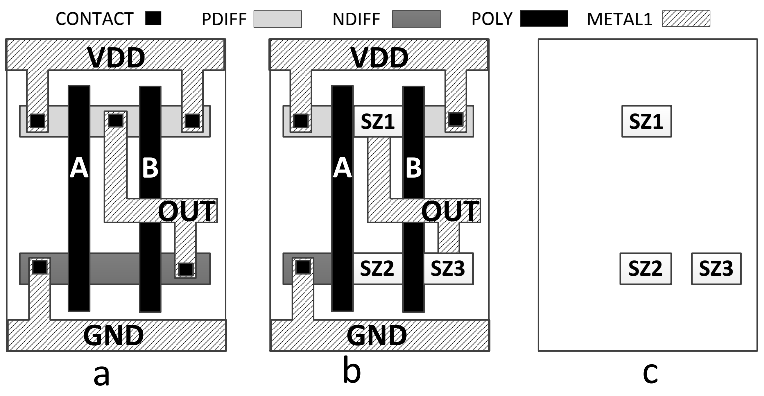


Figure 3.11: Sensitive Zone (SZ) extraction for a NAND gate: a) cell layout b) identification of diffusion parts which are not connected to supply voltage c) sensitive zones (adapted from [4])

Algorithmus 1 : Layout-based SER estimation

```
1  Extract a surrounding oval for each MBU pattern;
2  Extract sensitive zones of each cell by library characterization;
3  Divide layout into smaller grids and extract the list of cells in each grid;
4  while sampling error < predefined value AND number of cells covered by at least one MT < 99.9% do
5      Randomly select an oval based on their occurrence probability;
6      Place the oval in a random location on the layout;
7      Find the list of grids which have overlap with the oval;
8      Search overlapped grids cell lists and construct overlapped cells list;
9      Remove cells without overlapped sensitive zones from cell list;
10     Mark all cells in overlapped cell list as covered by an MT;
11     Propagate MT at logic-level and calculate failure probability;
12 end
13 Report average failure probability
```

Overall Flow

The main steps of the proposed layout-based SER estimation are summarized in Algorithm 1. Initially, the list of ovals and their occurrence probability are extracted from existing MBU patterns (line 1) and then technology library is characterized for identification of sensitive zones (line 2). These two steps are performed once in advance and their results are used for all circuits to be analyzed in the same technology and library settings.

Due to the large number of cells and sensitive zones inside industrial-size circuits, a hierarchical approach is employed to minimize the time needed for identification of error sites affected by an MT (line 3). In this approach, the entire layout area is divided into smaller grids and the list of cells inside each grid is extracted. During SER estimation, instead of searching among large number of cells, in the first step, for each grid it is checked whether it has an overlap with the oval surface and the list of layout grids overlapped by the oval surface is extracted. Then, the list of overlapped cells is extracted by investigation of cells inside overlapped layout grids. At the end, those cells which have no overlapping sensitive zone with the oval surface are eliminated from the target cell list. The remaining cells will be used as candidate MT fault sites (line 7-9).

3.2.3 Experimental Results of MT Modeling

Using the proposed layout-based MT error site extraction, we have performed an extensive analysis on the impact of particle energy on the MT error sites. By performing experiments on both Nangate 45 nm (Planner) and 15 nm (FinFET) technology nodes, we were able to show the impact of technology scaling on MT size. Additionally, the impact of netlist adjacency assumption on the overall SER of the circuit is investigated.

Experimental Setup

In order to show the scalability of the proposed approach, we have evaluated the largest available benchmark circuits in ISCAS'89 and ITC'99 benchmarks suites. For each benchmark, the HDL description of the circuit is first synthesized using a Synopsys Design Compiler for the maximum possible operational clock frequency[1]. Then, the layout of the netlist is extracted using Cadence SoC Encounter by considering utilization ratio of 75% (typically about 65%-85% [111]). The minimum clock period as well as the chip area for each circuit are reported in Table 3.2. According to these results, the area and frequency are decreased by around 4X for the circuit synthesized for 15 nm compared to 45 nm.

Table 3.2: Minimum clock period and area of benchmark circuits synthesized into 45 nm and 15 nm Nangate standard cell libraries

Benchmark	Nangate 45 nm		Nangate 15 nm	
	Clock Period [ps]	Area [μm^2]	Clock Period [ps]	Area [μm^2]
s15850	825	9,134	184	2,723
s35932	487	32,735	105	9,669
s38417	924	27,760	139	9,320
s38584	610	28,709	121	8,754
b17	1,357	32,175	331	8,451
b18	2,133	101,200	549	27,200
b19	2,046	194,360	530	52,232
b20	1,734	29,433	452	8,259

The MBU patterns for particles with 22, 37, 95, and 144 Mev provided by [32] are used during the layout-based MT extraction. This information is given to our layout-based SER estimation to calculate the overall SER of the circuit according to Algorithm 1. In our framework, SER estimation analysis terminates when the maximum inaccuracy of the Monte-Carlo is less than 0.5%. In this experiment, the layout is divided into 30×30 μm^2 grids, where each grid includes around 800 cells.

MBU Patterns and MT Error Sites

As mentioned earlier, in order to extract MT error sites, the area affected by MBU patterns are first extracted and then a surrendering oval for each MBU pattern is constructed. These ovals are used for identification of MT error sites. For this purpose, detailed information about different MBU patterns in a memory array is necessary for identification of MT error sites.

Radaelli et. al. [32] have reported a detailed information about predominant MBU patterns in a 150 nm technology SRAM device and their occurrence probability for particles with 22, 47, 95, and 144 Mev energies. Considering the SRAM cell dimensions, the area affected by each MBU surrounding oval can be accurately estimated. For these cases, the oval shapes and their occurrence probability (same as the occurrence probability of the corresponding MBU pattern) are computed. Table 3.3 shows the average area affected by each particle energy obtained by wighted averaging of oval surfaces based on their occurrence probability.

The area affected by a particle strike is mostly a function of particle energy, while the strength of the transient pulse mostly depends on other parameters such as diffusion volume (width, length, depth) and load capacitance [3]. As a result, the affected area information

[1]This requires two rounds of synthesis: First round is for obtaining the maximum operational frequency. In this round, the clock period is set to zero to force the synthesis tool to implement all paths as short as possible. The maximum negative slack obtained shows the maximum operational frequency. Then, another synthesis operation is performed in the second round to minimize the area and power for the maximum operation frequency.

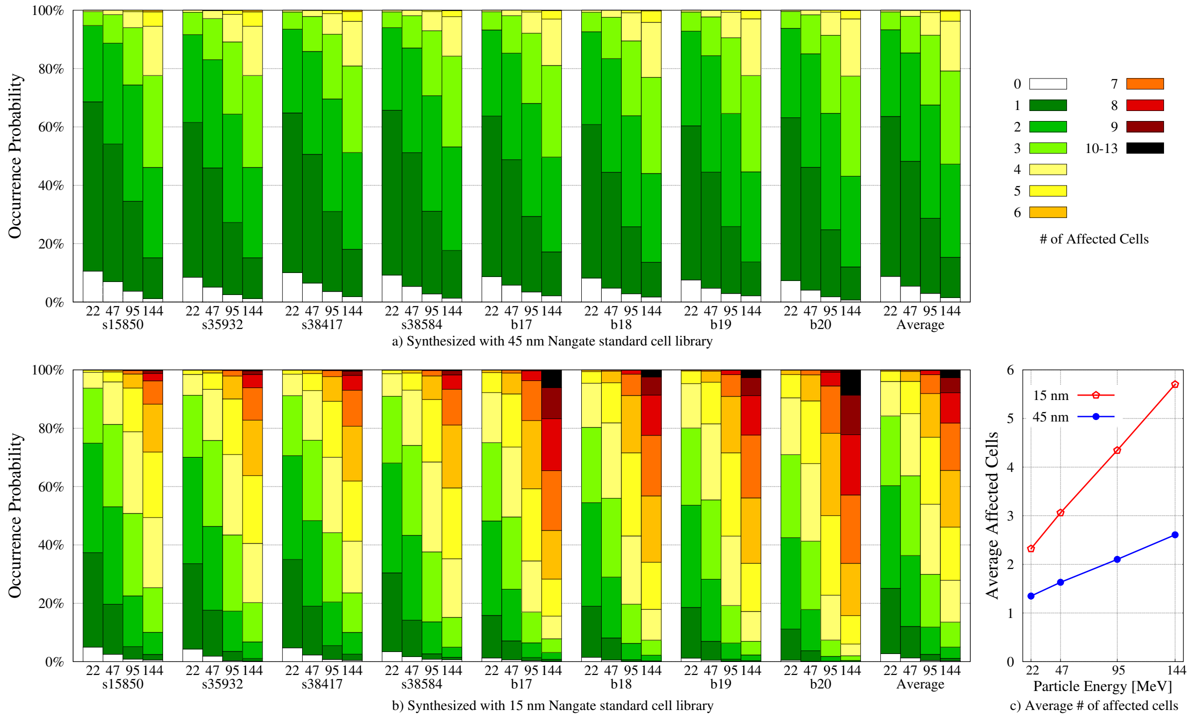


Figure 3.12: Distribution of # of cells (flip-flops or combinational gates) affected by particle strikes with 22, 47, 95, and 144 Mev energies for benchmark circuits synthesized with respect to a) 45 nm and b) 15 nm Nangate standard cell library as well as c) average affected cells per technology node

Table 3.3: Average area affected with different particle energies

Particle Energy (Mev)	Average Affected Area (μm^2)
22	1.178
47	1.902
95	2.903
144	4.613

acquired for a 150 nm SRAM technology can also be used for the logic area affected by a particle strike with the same energy in the 45 nm and 15 nm technology nodes. Please note that although the affected area remains constant, however, due to the technology downscaling, the number of affected cells increases in smaller technologies.

By randomly locating these ovals on the circuit layout according to their occurrence probability, different combinations of affected combinational gates and flip-flops are extracted and identified as MT error sites. Figure 3.12 shows the occurrence probability of different number of affected cells for particle strikes with 22, 47, 95, and 144 Mev energies. As it is expected, by increasing the particle strike energy, the occurrence probabilities of SET and SEU decreases significantly and MT becomes predominant. In addition, the MT ratio in 15 nm is significantly larger than that in 45 nm.

In this experiment, the maximum number of affected cells at 45 nm and 15 nm are 8 and 13 cells, respectively. In order to clearly show the impact of technology scaling on MT rate, the average number of affected cells for the aforementioned particle energies are reported in Figure 3.12.c. This figure reveals that the average number of affected cells is almost doubled for all particle energies from 45 nm to 15 nm. Hence, accurate modeling and mitigation of MTs are more challenging in smaller technology nodes with larger number of affected cells.

In previous netlist-based techniques [108, 109], it is assumed that a particle strike leads to either MET on combinational logic or MBU on sequential cells. Also, the number and type of affected cells was a function of the particle energy rather than a function of the layout. However, our results show that 1) adjacent combinational and sequential cells can be simultaneously affected by a single particle strike, and 2) the number, type, and the combination of affected cells depend on the layout structure as well.

SER Estimation

Although the affected area information could be acquired from MBU patterns, the pulse width of generated SETs is completely technology dependent [112, 113]. In order to obtain the distribution of the generated pulse widths in each technology node for a certain particle energy, we exploit the commercial tool introduced in Section 3.1. Using this tool, the distribution of pulse widths for each standard cell was generated for a given particle energy. For error generation, we exploited such distributions to generate random pulse widths in the affected combinational gates.

For propagating the generated pulse width from MT error sites, we exploit an analytical technique presented in Section FIXME. This technique model logical, electrical, and latching-window masking in a unified framework and propagates the errors for several clock cycles (here, 10). In this section, the failure probability is defined according to [114, 61] as the probability of propagation from error sites to primary outputs during first few cycles after error occurrence.

Impact of SET/SEU Versus MT Model on Overall SER

In order to explore the importance of MTs and technology scaling impact on the failure probabilities, overall SERs extracted for particles with 22, 47, 95, and 144 Mev energies are compared with the case that the simple SET/SEU model is considered for both 45 nm and 15 nm technology nodes (Figure 3.13). In case of SET/SEU, a single error is injected in each gate/flip-flop and the average of the failure probabilities of all cells are reported as the circuit failure probability.

The results shown in Figure 3.13 reveal that the average failure probability significantly increases from 45 nm to 15 nm (by more than 2X in some cases). One reason for such an increase is that the number of affected cells is almost twice in the smaller technology node (as shown in Figure 3.12.c). Another interesting reason is higher operational frequency and smaller latching window of the circuits designed with Nangate 15 nm (see Table 3.2) which significantly affects the error latching probability. The average latching windows of flip-flops in 45 nm and 15 nm are 42.4 and 12.7 ps, respectively. As shown in [21], the combinational logic SER linearly grows by increasing the frequency due to increased ratio of latching window to the entire clock period.

The results also reveal that the circuit failure probability does not linearly increase with the particle energy. As an example, on average, the SER in the presence of 47 Mev particles is only 1.15X greater than when considering 22 Mev particles. This can be explained by the number of affected cells for different particle energies as reported in Figure 3.12.

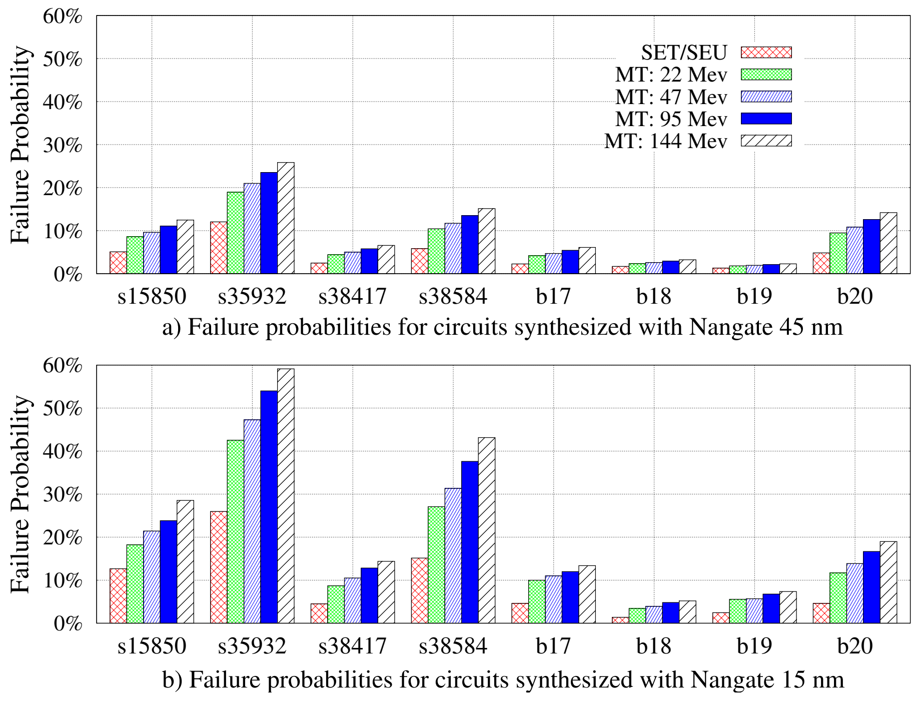


Figure 3.13: Average failure probabilities of different particle energies for circuits synthesized with respect to Nangate a) 45 nm b) 15 nm standard cell

One reasonable solution to show the SER trend from 45 nm to 15 nm is to investigate the trade-off between failure probability increase and the decrease in the particle strike rate across different technology nodes. As shown in Table 3.2, the area and clock period of circuits in 15 nm are reduced by 3.5X and 4.6X compared to 45 nm. This means 15.8X less particles strikes for a certain task running on a circuit fabricated in 15 nm due to the smaller area and shorter runtime. On the other hand, the failure probability grows on average by 2.2X from 45 nm to 15 nm (see Figure 3.13). Hence, the overall trend would be 7.2X reduction in the overall SER. These results are inline with that of the existing studies [113, 115, 116].

Impact of netlist adjacency assumption on SER

In order to investigate the effect of netlist adjacency on the overall SER, we have implemented a netlist-based approach. The error propagation method of the netlist-based approach is similar to the layout-based approach. Although different combinations of affected cells and their occurrence probability are unknown at the netlist-level, in order to have a fair comparison, the same occurrence probabilities is also used in the netlist-based approach. Figure 3.14 reports the failure probability obtained by both netlist- and layout-based approaches. As it can be seen, the netlist-based approach always underestimates the overall failure probability. Our analysis reveals that there are two main reasons for this underestimation. First, when there are simultaneous errors at the outputs of the CFI pairs, these transient pulses reach at the same time to the inputs of the fanout gate. In this case, the propagated transients are either completely masked, attenuated, or at least converged to one transient pulse. However, as shown in Figure 3.6, most of CFI pairs are not physically adjacent in the layout. Second, the forward cones of netlist adjacent pairs are highly overlapped and share similar paths from error sites to the circuit outputs. This can increase the chance that several errors are masked due to one kind of masking (e.g., logical masking in a common gate in the forward cone of both error sites). When MT occurs in error sites which have non-overlapping forward cones, they are independent and the probability of masking is much lower.

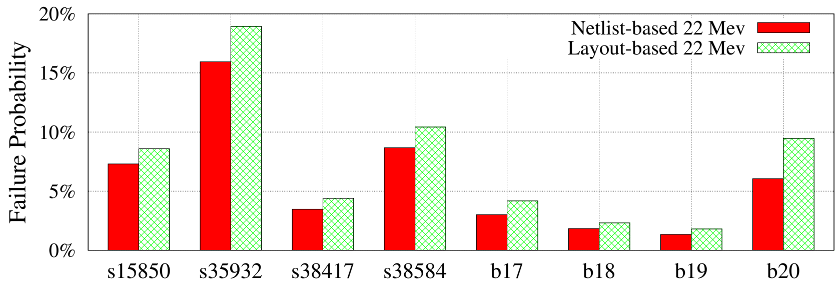


Figure 3.14: Comparison of overall SER obtained by netlist-based and layout-based approaches for circuits synthesized for Nangate 45 nm standard cell library and particle with 22 Mev energy

On average, netlist-based MT analysis has an inaccuracy of 22.34% which is as high as 36.04% for b20 benchmark. Please note that since the information regarding the occurrence probability of different affected gate/flip-flops does not exist at the netlist level, the inaccuracy of those techniques could be even higher.

3.3 Propagating Flip-flop Errors at Circuit-level

Since the soft error vulnerability of flip-flops is quite non-uniform [18], a fast and accurate *Soft Error Rate* (SER) estimation technique is required to identify the most vulnerable flip-flops and replace them with robust alternatives. The combined effect of interdependent and correlated logical and temporal masking factors determines the vulnerability of flip-flops to soft errors and has to be considered in a unified manner to reach a reasonable accuracy. These factors are interdependent and highly correlated as they are related to the logic state and delay of the components (logic gates and flip-flops) on the error propagation paths. As a result, an accurate soft error estimation technique should be able to capture combined effect of these factors.

While both logical and temporal masking factors can prevent error propagation in the error generation cycle, once the error is latched in the downstream flip-flops at the end of that cycle, the errors can only be logically masked in the subsequent cycles. The logical and temporal masking factors are dependent on the logical value and the delay of the error propagation paths. When there is no activity on a certain part of the circuit, error propagation behavior on that part is the same as the previous cycle. Therefore, computing error propagation possibility for all paths in every clock cycle is unnecessary and adjusting error propagation paths according to the logic events serve the same purpose. In this regard, in order to compute the vulnerabilities of flip-flops to soft errors, we propose to embed transient error propagation in an event-driven logic simulator.

Our objective is not only to consider the logic masking in the propagation of the errors, but also to accurately compute the vulnerable time interval within each cycle according to the temporal masking factor. For this purpose, in the cycle of error occurrence, the delay required for an error to reach each downstream flip-flop has to be computed. We perform such a timing analysis in the proposed event-driven error propagation approach. Thus, the proposed approach not only identifies the vulnerable cycles, but also determines the vulnerable time interval within the vulnerable cycles.

In the proposed approach, errors originating from different flip-flops are concurrently propagated across multiple clock cycles, so there would be no need to repeat error propagation for different flip-flops and for different cycles of the error occurrence. This results in multi-fold speedup over statistical fault injection. While conventional concurrent fault simulation techniques reduce the simulation time by only evaluating multiple flip-flops simultaneously, the proposed approach reduces the number of error propagations by decreasing number of error injection cycles as well. Even when there is a logic event, the error propagation behavior changes for the subset of error propagation paths. These result in significant speedup (two orders of magnitude as shown in experimental results section) over conventional concurrent fault simulation techniques.

3.3.1 Event-driven Logic Simulation

We exploit a very accurate logic simulation method which implements the three-valued logic system ('0','1', and 'X') and considers different rise and fall delay (i.e. (t_r, t_f)) for each input-output pair of a component (flip-flop or a combinational gate). The detailed timing information are obtained from a Standard Delay Format (SDF) file. Similar to other event-driven logic simulators, we exploit the *time wheel* data structure to store the list of events to be simulated and their corresponding simulation time. When an event is simulated which changes the logical value of a component output, for each fan-out node of that output, an event is inserted in the time wheel with respect to the delay of that component. This event-driven simulator accepts a synthesized gate-level netlist and an SDF file as inputs and performs the timing-aware post-synthesis logic simulation.

3.3.2 Error Propagation from Single Flip-flop

We first explain the proposed error propagation approach for a single flip-flop as the error site and in the next subsection, we show how multiple error sites (flip-flops) are evaluated concurrently in one pass without sacrificing the accuracy.

In the proposed approach, we do not explicitly inject errors during workload simulation. Instead, we compute the list of time-intervals in which an SEU error on the target flip-flop is propagated to each node in the circuit. Each node in the circuit has an associated error list, and each element of this list is represented as a tuple of $[FF_i, c, (t_s, t_e)]$ which means that if an SEU occurs in flip-flop FF_i during time interval of (t_s, t_e) at c cycles ago, it will propagate to the node which is assigned. (t_s, t_e) is a subinterval of one clock period, i.e. $(0, clock\_period)$, and determines the temporal masking factor.

Figure 3.15.a demonstrates an example of error lists associated with internal nodes. Each list has two elements, both originating from the same flip-flop FF_S. In all examples used in this section, the clock period is assumed to be 1000 ps. The tuple $[FF_S, 0, (0, 500)]$ at node a shows that an SEU occurring in the first half of the current cycle propagates to this node. an error occurred in the last 40 ps of $(0, 500)$ interval is temporally masked due to the delay of

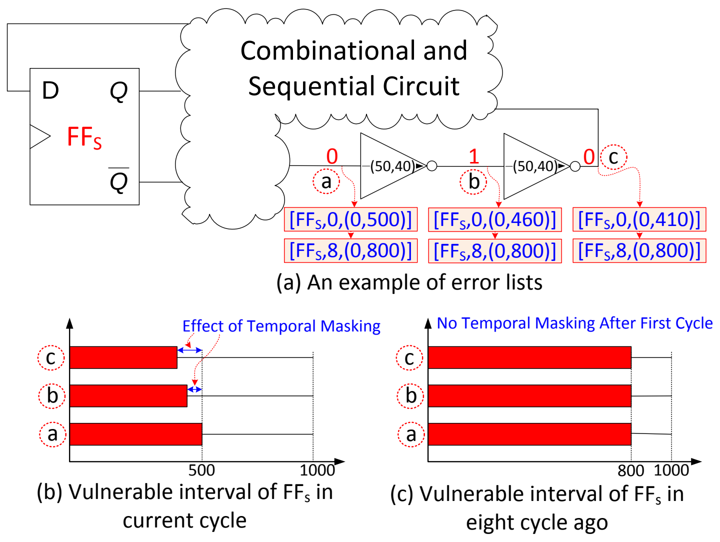

Combinational and Sequential Circuit
D
Q
FFS
Q
0
a
(50,40)
1
b
(50,40)
0
c
[FFS,0,(0,500)]
[FFS,0,(0,460)]
[FFS,0,(0,410)]
[FFS,8,(0,800)]
[FFS,8,(0,800)]
[FFS,8,(0,800)]
(a) An example of error lists
Effect of Temporal Masking
No Temporal Masking After First Cycle
500
1000
800
1000
(b) Vulnerable interval of FFs in current cycle
(c) Vulnerable interval of FFs in eight cycle ago

Figure 3.15: A conceptual example of error propagation in proposed approach

the inverter and cannot reach to node b. Thus, errors occurred in $(0, 460)$ can only reach to this node. The effect of temporal masking factor during error propagation is shown in Figure 3.15.b. While SEU errors in 1-7 cycles ago are masked and cannot reach these nodes, error in the time interval of $(0, 800)$ of 8 cycle ago is also propagated to node a. As mentioned earlier, the temporal masking factor is only important in the first cycle, hence, it has no effect on the propagation of this error and it is propagated to nodes b and c without any change (see Figure 3.15.c).

The error lists of the nodes are updated according to data events during the event-driven logic simulation. As we will show later, the data events and the update of error lists are highly correlated. This means that error lists remain unchanged when there is no data event to be simulated. On the other hand, only the error lists associated with the nodes with data event will change.

During the workload simulation, the vulnerability of the target flip-flop is computed according to the errors propagated to POs. For example, if $[FF_S, 5, (0, 500)]$ is added to a PO node at cycle k and removed at cycle $k+100$, flip-flop FF_S is vulnerable at time intervals of $(0, 500)$ of cycles $k-5$ to $k+95$ as error requires 5 cycles to propagate to this node. By combining of the errors propagated to the POs, the overall vulnerability of the target flip-flop is computed.

Error Generation at Target Flip-flop

In case the value stored in a flip-flop has the state of 'X', its value is *don't care* for the correct execution of the workload. Therefore, errors are only generated when the flip-flop stores either '0' or '1'. An error occurring inside a flip-flop requires some time to propagate to its output(s). We assume this time is equal to $Clk \rightarrow Q$ and $Clk \rightarrow \overline{Q}$ delays for Q and $\overline{Q}$ nodes, respectively.

As an example, consider the flip-flop shown in Figure 4.3. The rise and fall pair, i.e. (tr, tf), for $Clk \rightarrow Q$ and $Clk \rightarrow \overline{Q}$ are (117,97) and (96,89) ps, respectively. When the flip-flop output is 'X', there is no error at the output nodes. When the flip-flop stores '1', an SEU would change its Q ($\overline{Q}$) output from '1'('0') to '0' ('1') which requires $t_f^{Clk \rightarrow Q}$ =97 ($t_r^{Clk \rightarrow \overline{Q}}$= 96) ps to reach

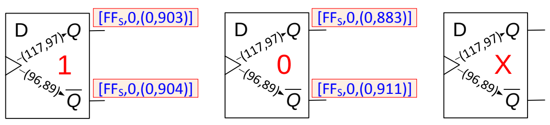


Figure 3.16: Error propagation to target flip-flop output

to this node. This means that if an SEU occurs within 97 ps (96 ps) before the clock edge, it does not have enough time to propagate to Q ($\overline{Q}$) node. Therefore, for the clock period of 1000 ps, an SEU in the time interval of [0,903] ([0,904]) could propagate to Q ($\overline{Q}$) node. The case in which the flip-flop stores '0' can be analyzed similarly.

When the value stored in the target flip-flop changes at positive clock edge, an event is inserted at Q and $\overline{Q}$ outputs in the time wheel with respect to $Clk{\rightarrow}Q$ and $Clk{\rightarrow}\overline{Q}$ delays. When such an event is simulated, the error list of the corresponding target flip-flop output is updated. The update could be the elimination of an existing element (state: '1'→'X' or '0'→'X'), update of an existing element (state: '1'→'0' or '0'→'1'), or adding a new element (state: 'X'→'0' or 'X'→'1'). Please note that these error lists are only updated when there is a data event at flip-flop output, otherwise, the error propagation behavior is the same as that of the previous cycle and re-computation is unnecessary.

Error Propagation Through Combinational Gates in First Cycle

In case an error reaches an input of a combinational gate, depending on the logic value of other inputs, it might propagate to its output. However, when there is no event, except a special case which is detailed later, the error propagation behavior remains the same. As a result, when simulating an event on a gate, we should also update the error list of the output node with respect to those of the all inputs.

Let us assume that $[FF_i, 0, (t_s, t_e)]$ is an element in the input error list of a gate and the other inputs of the gate are not in the controlling state. This error propagates to the output node of the gate with respect to rise or fall delay (d) of the gate dependent on the error polarity and gate functionality. Although errors occurred at time interval of (t_s, t_e) propagate to the gate input, due to the gate delay, errors in $(t_e - d, t_e)$ are temporally masked and cannot reach to the gate output. Hence, $[FF_i, 0, (t_s, t_e - d)]$ is added to the output error list.

An example of propagation behavior in combinational gates, when error occurs in flip-flop FF_s, is shown in Figure 3.17. In step (a), the target flip-flop has a value of 'X' and, as discussed earlier, there is no error at its output error list and hence there is nothing to be propagated in combinational gates. In step (b), the state of this flip-flop is changed to '1' and the flip-flop output error list contains $[FF_S, 0, (0, 903)]$. The other inputs of the NAND and NOR gates are not in the controlling state and the error is propagated through these gates when the corresponding logic events are simulated from the time-wheel. In step (b), the output of the NAND gate is '0'. In case of an SEU error in FF_S, the output of this gate would change to '1' which requires a delay equal to $t_r = 38ps$. As a result, the output event for this gate would be $[FF_S, 0, (0, 903 - 38 = 865)]$. The Similar concept applies to the NOR gate output while for the AND gate, since the other input is in controlling state, the error is masked and does not propagate to its output. In step (c), the value of flip-flop changes to '0'. In this case, all the error lists are updated by replacing the new timing intervals with the old ones.

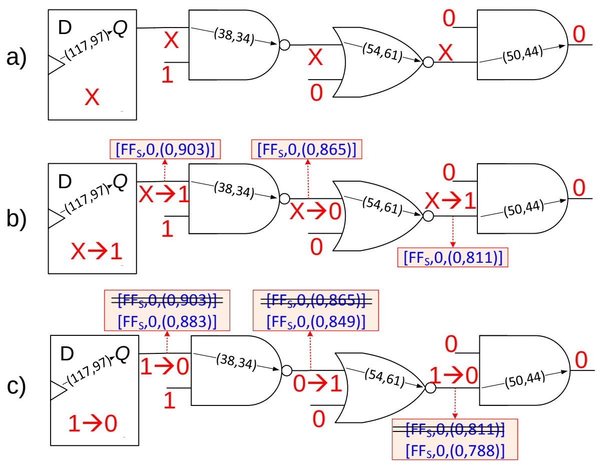


Figure 3.17: Examples of error propagation through combinational gates a) no error b) adding new elements to lists c) updating existing elements of lists

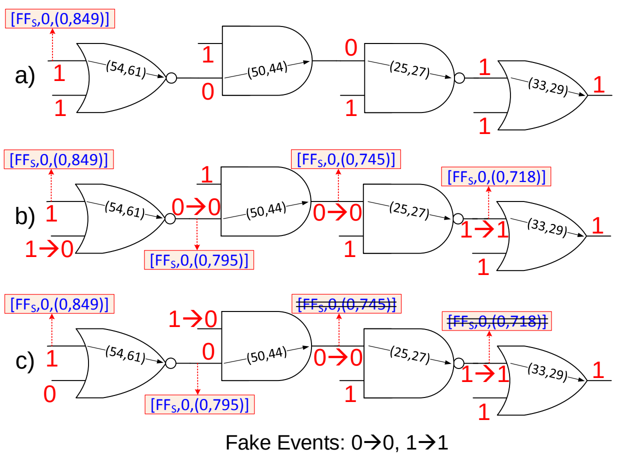


Figure 3.18: Error propagation with fake events a) Error is logically masked b) Error is propagated with fake events c) Error is eliminated with fake events

Although in the example shown in Figure 3.17, the update of error lists is always done during the simulation of an event in the corresponding node, there are some exceptional cases where an error list has to be updated while there is no event at that node. Figure 3.18 demonstrates an example of such cases. In step (a), there is an error in the first input of the NOR gate which is logically masked as the second input is in the controlling state. In step (b), the second input of the NOR gate changes to '0'. This input is not in the controlling state any more and the error could propagate through the gate. However, this change does not affect the gate output and there is no event on this node. In order to propagate this kind of errors, a *fake event* is inserted in the logic-simulator time wheel. The fake events assign the previous value of a node to it and, hence, has no effect on the functionality of the event-driven simulator. After simulating the fake event, the error list of the node is updated. In step (b), there are three fake events demonstrated by either '0'→'0' or '1'→'1' to add an error to the error list. In step (c), an event changes the first input of AND gate to '0'. Consequently, the error in the second input cannot propagate to the output of this gate. In this case, similar to step (b), fake events are employed to update the error list when there is no event, however, it results in the elimination of some errors due to logical masking factor.

Error Propagation After First Cycle

Temporal masking only occurs at the cycle of error occurrence and once the error is latched in the flip-flops, only logical masking can prevent the error propagation. All errors in a flip-flop input error list are propagated to its output. While propagating the errors to flip-flop output, the number of cycles (i.e. c) for that error is incremented to show that it belongs to previous cycles while the time interval remains the same as that of the corresponding error in the input. Indeed, in the cycle error occurs, the timing intervals are adjusted with respect to the gate delays during error propagation to estimate the temporal masking factor. Once error reached to an input of a downstream flip-flop, its time interval shows the amount of temporal masking factor. For instance, the error $[FF_i, 0, (0, t_e)]$ in the input of a flip-flop shows that an SEU occurring in the time interval of $(t_e, clock\_period)$ is going to be temporally masked. Once an error is latched in the flip-flops, the propagation in the combinational logic does not change its vulnerable time interval and it could be only masked by means of the logical masking factor.

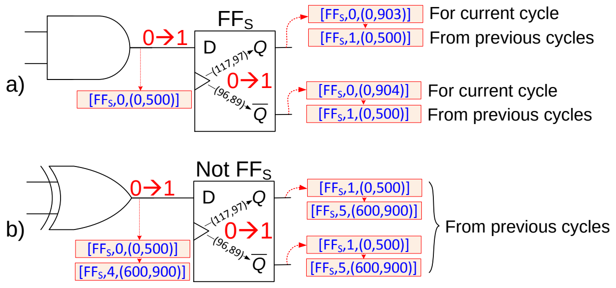


Figure 3.19: Error propagation through a) target flip-flop b) non-target flip-flop

Figure 3.19 shows error propagation through flip-flops. As shown, all errors in the input error list propagate to the output list by incrementing the number of cycles. Beside the errors propagated from the input, the target flip-flop has also new elements in its error list for the errors generated in the current cycle. Figure 3.20 shows the error list of different nodes of a combinational logic for errors occurred in the current cycle and those propagated from previous cycles. As it can be seen, for the error occurred in this cycle (i.e. $[FF_i, 0, (0, 849)]$), the timing interval is adjusted during the propagation step to consider the effect of temporal masking factor while for those propagated from one and seven cycles ago, the timing interval is unchanged.

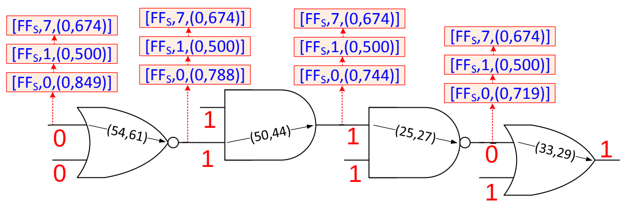


Figure 3.20: Error propagation through combinational gates after first cycle

Vulnerability Time Computation

The vulnerability of target flip-flop is estimated with respect to the error list of POs. Let us assume that $[FF_S, c, (t_1, t_2)]$ is an error appeared at a PO at cycle k_1 and removed from this node at cycle k_2. Considering the fact that the propagation of the error takes c cycle, it would be clear that errors occurred at (t_1, t_2) time interval of $(k_1 - c, k_2 - c)$ cycles of flip-flop FF_S are propagated to this PO. For each error in the PO, the vulnerable time intervals and their corresponding clock cycles are computed. The union of all these time intervals is the overall vulnerable intervals of flip-flop FF_S. The vulnerability of this flip-flop is the sum of all vulnerable intervals over the overall workload execution time.

3.3.3 Concurrent Transient Error Propagation from Multiple Flip-flops

In Section 3.3.2, only vulnerability of one flip-flop is computed during the workload execution. Therefore, the event driven error propagation has to be repeated for each flip-flop in the design as the error site. Considering the fact that an error only affects a small subset of the nodes, error lists of most of the nodes are empty and major proportion of the runtime is spent for logic simulation rather than error propagation. By concurrent transient error propagation from multiple flip-flops, the time spent for re-computation of logic events for each flip-flop could be significantly reduced. In this case, errors from different flip-flops are propagated simultaneously, however, they do not affect each other. This means that in case one gate has errors from multiple flip-flops in its inputs, their effects at the output are independently computed.

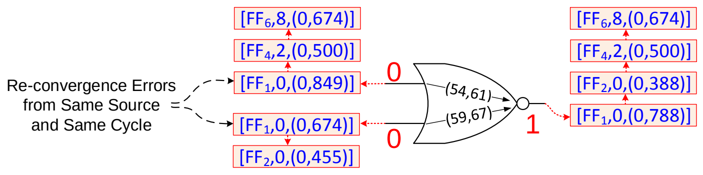


Figure 3.21: An example of concurrent error propagation

In order to expedite the operation of finding events that could affect each other (i.e. from the same flip-flop and the same cycle), the error lists of the nodes are stored in a sorted order with respect to the flip-flop index and cycles. During the error propagation, these lists are processed in the sorted order and for each pair of flip-flop that have the same flip-flop and cycle, the combined effect of these errors is computed. Figure 3.21 demonstrates an example of concurrent error propagation on a combinational gate. As it can be seen, two errors are propagated from FF_1 in the current cycle and these create one output. All the remaining errors have unique flip-flop index and propagation cycle, and hence, are processed independently.

3.3.4 Experimental Results

In order to show the effectiveness of the proposed approach, we conducted an experiment to evaluate the vulnerability of all flip-flops of the OR1200 processor with respect to four MiBench workloads.

Experimental Setup

The OR1200 core is synthesized with Nangate 45 nm standard cell library using Synopsys Design Compiler. The resulted netlist has 2693 flip-flops and 30,986 combinational gates. The minimum clock period for the synthesized netlist is 1.39 ns. Four workloads from the MiBench [117] benchmark suite namely *Bitcount, CRC32, QSort, and Stringsearch* are simulated on the OR1200 processor during our experiments. All experiments are performed on a workstation with Intel Xeon E5540 2.53 GHz processor and 16 GB RAM.

Validation of Error Propagation Results

The error propagation capability of the proposed approach is compared with fault injection results using Modelsim to validate both masking factors for 2,000 randomly injected errors. Each error is traced for 100 cycles and propagation behavior is compared with our simulator, which showed 100% accuracy.

Runtime of Transient Error Simulator

Figure 3.22 shows runtime of event-driven error propagation for concurrently evaluating various number of flip-flops and propagation cycles during the execution of Stringsearch workload. As shown, the runtime of the our approach grows linearly with the number of evaluated flip-flop until 2,000 and then becomes exponential. Also, it has linear relation with the limit for the number of propagation cycles.

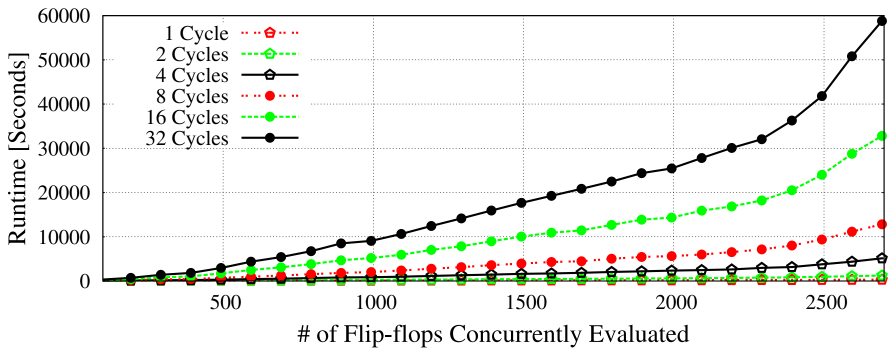


Figure 3.22: Runtime vs. # of flip-flops concurrently evaluated

Since the runtime of the proposed approach is highly dependent on the number of evaluated flip-flops, evaluating all flip-flops in one pass might not be the best option from the runtime point of view. Indeed, simultaneous evaluation of large number of flip-flops increases the computation complexity as error events in the input of a gate have to be sorted to identify the re-convergent events. By having more events in the inputs of a gate, the sorting algorithm requires more time as it has a complexity of $O(n^2)$ where n is the number of error events to be sorted. In order to show this effect, we estimated the overall runtime required for evaluating all OR1200 flip-flops with respect to Stringsearch workload by concurrently evaluating different number of flip-flops. The results of this assessment are presented in Figure 3.23. As shown, evaluation of flip-flops in 10 sets with 270 flip-flops gives the minimum runtime.

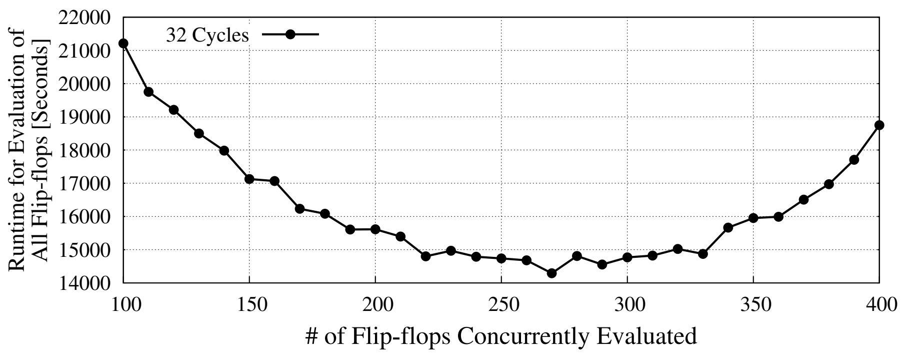


Figure 3.23: Runtime for evaluation of all the design for different number of concurrently evaluated flip-flops

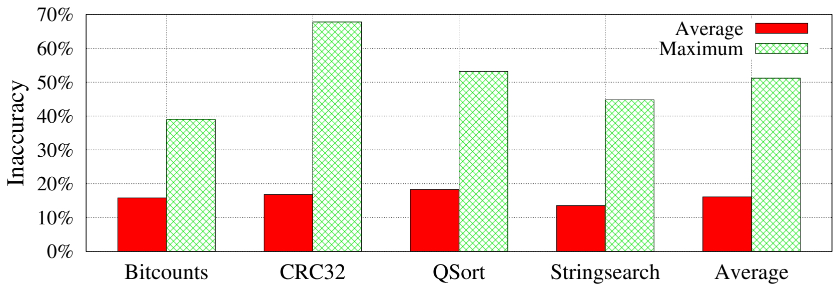


Figure 3.24: Average and maximum inaccuracy imposed due to independent analysis of logic and temporal masking factors

Independent Analysis of Masking Factors

An important feature of the proposed approach is the combined analysis of the logical and temporal masking factors. In order to show the benefits of this approach, we compared the soft error vulnerability obtained by this approach by that of the independent analysis of these two factors. For the independent analysis, the logical masking factor is computed by our tool as the ratio of the number of vulnerable cycles over all cycles (i.e by ignoring computed vulnerable time intervals inside vulnerable cycles which represents the temporal masking factor). The temporal masking factor is extracted with respect to shortest path starting from the target flip-flop according to the method detailed in [118]. As shown in Figure 3.24, the independent analysis imposes inaccuracy by up to 67.8% in vulnerability estimation as error does not necessarily propagate through short paths and and typically there is a considerable difference in the delay of paths in a circuit.

Comparison with Other Techniques

In order to show the effectiveness of the proposed approach in terms of accuracy and runtime, we experimentally compared two versions of our technique with two other techniques:

- Proposed-16 and Proposed-32 propagate the error for 16 and 32 cycles, respectively, and in each iteration of the error propagation 270 flip-flops are evaluated concurrently.
- SFI is a statistical fault injection built on top of our logic simulator.

- Concurrent Fault Simulation (CFS) [119] is employed to expedite the statistical fault injection by concurrently propagating errors from different flip-flops. However, this technique needs to repeat the vulnerability assessment for each cycle and even for random times within each cycle in order to compute the temporal masking factor.

Table 3.4: Average and maximum inaccuracy of different techniques

Workload	Bitcounts		CRC32		QSort		Stringsearch		Average	
	Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.	**Avg.**	**Max.**
Proposed-16	0.2%	0.7%	0.6%	1.7%	1.7%	3.6%	0.5%	1.1%	**0.7%**	**1.8%**
Proposed-32	0.1%	0.5%	0.1%	0.4%	0.8%	2.1%	0.3%	0.9%	**0.4%**	**0.9%**
CFS-5% SFI-5%	Maximum inaccuracy of 5.00% with confidence-level of 95% which requires 1,034,112 fault injection according to [120]									

Table 3.5: Comparison of runtime of different techniques [Minutes]. speedups are w.r.t SFI

Workload		Bitcounts	CRC32	QSort	Stringsearch	Average
Proposed-16	absolute	178	233	669	76	
	speedup	91,356	88,370	101,308	90,767	**92,950**
Proposed-32	absolute	560	698	2,164	239	
	speedup	29,037	29,599	31,346	29,091	**29,768**
CFS-5%	absolute	141,628	144,389	517,976	48,114	
	speedup	115	143	131	144	**133**
SFI-5%	absolute	16,287,264	20,647,769	67,854,982	6,928,550	

The average and the maximum inaccuracies of different techniques for evaluation of OR1200 flip-flops are reported in Table 3.4. Additionally, the runtime of each technique for evaluation of different workloads is reported in Table 3.5. As it can be seen, CFS provides 133X speedup over SFI for simulating same number of errors. Among these, our proposed approach provides four orders of magnitude speedup with respect to SFI while it is highly accurate. The only source of inaccuracy in our proposed approach is the errors which require large number of simulated cycles to cause a failure. This approach is much faster than CSF as it skips unnecessary computations when there is no logic event. These features make our technique appropriate solution for soft error vulnerability of flip-flops.

3.4 Propagating Combinational Gates Errors at Circuit-level

A transient pulse in combinational gates is only latched in flip-flops if it is not masked by any of the logical (where another input of a gate is in controlling state), electrical (pulse electrically attenuated during propagation), or latching-window masking (pulse is not captured in the flip-flops as it is not arrived in the latching-window of receiving elements) factors [121].

Since the electrical and latching-window masking behavior of combinational logic is completely different in a conventional emulation environment [52], SPICE-based fault injection is the most accurate approach for analyzing SER of these elements. However, as the number of combinational gates is much more than flip-flops and also the error propagation should be repeated for different pulse widths, reaching a reasonable level of accuracy with such fault injections is intractable even for a small circuit with a few thousand gates.

In this regards, we employed a combination of previously proposed analytical techniques to propagate the errors in the combinational logic in the first cycle. The techniques presented

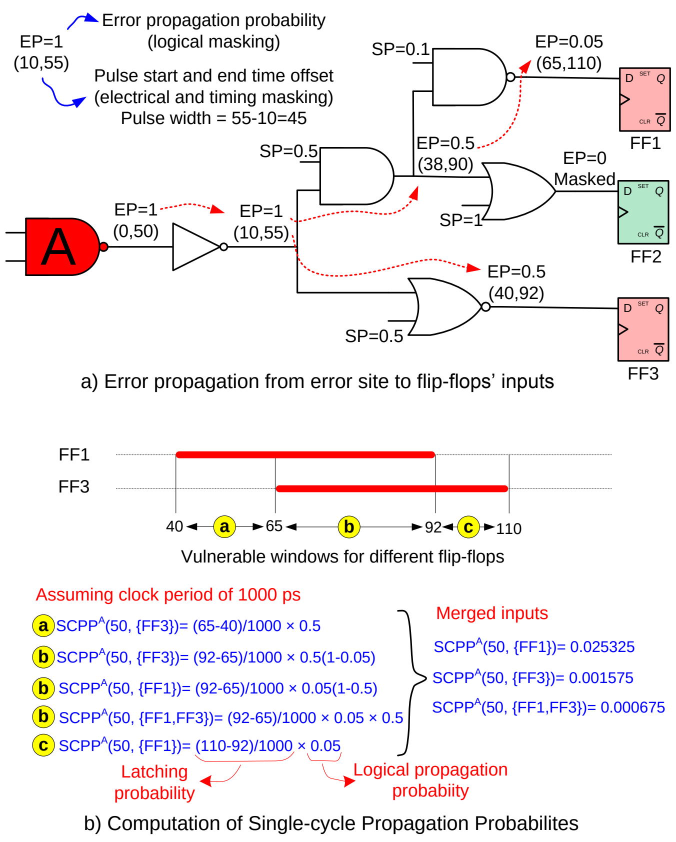

Error propagation probability
(logical masking)
EP=1
(10,55)
Pulse start and end time offset
(electrical and timing masking)
Pulse width = 55-10=45
SP=0.1
EP=0.05
(65,110)
FF1
SP=0.5
EP=0.5
(38,90)
EP=0
Masked
A
EP=1
(0,50)
EP=1
(10,55)
SP=1
FF2
EP=0.5
(40,92)
SP=0.5
FF3
a) Error propagation from error site to flip-flops' inputs
FF1
FF3
40 a 65 b 92 c 110
Vulnerable windows for different flip-flops
Assuming clock period of 1000 ps
a SCPP^A(50, {FF3})= (65-40)/1000 × 0.5
b SCPP^A(50, {FF3})= (92-65)/1000 × 0.5(1-0.05)
b SCPP^A(50, {FF1})= (92-65)/1000 × 0.05(1-0.5)
b SCPP^A(50, {FF1,FF3})= (92-65)/1000 × 0.05 × 0.5
c SCPP^A(50, {FF1})= (110-92)/1000 × 0.05
Latching probability
Logical propagation probabiity
Merged inputs
SCPP^A(50, {FF1})= 0.025325
SCPP^A(50, {FF3})= 0.001575
SCPP^A(50, {FF1,FF3})= 0.000675
b) Computation of Single-cycle Propagation Probabilites

Figure 3.25: Error propagation using single-cycle analysis

in [62], [122], and [123] have been used for modelling logical, electrical, and latching-window masking factors, respectively.

The single-cycle analysis was repeated for different pulse widths. At the end of single-cycle analysis for pulse width w, we obtained $SCPP^g(w, s)$ which is the probability of propagation of the transient pulse from gate g to all flip-flops in set s.

A running example of error propagation using the employed single-cycle analytical technique is shown in Figure 3.25. In this example, an SET with pulse width of 50 ps is propagated from gate A towards flip-flop inputs. For each gate in the forward cone of the error site, logical and electrical masking as well as the vulnerability window are computed.

Logical masking: The probability of *Error Propagation* (EP) to the gate output is computed with respect to the signal and error probabilities of its input nodes according to the rules provided in [62]. For instance, in the running example, one input of the AND gate has EP=1 while the other input has signal probability of 0.5. Hence, only 50% of errors are propagated to the output of this gate and EP of the output node is 0.5.

Electrical masking: The pulse width of the transient error at the output of a gate is computed according to the equation-based model described in [122]. This technique approximates a transient pulse with a trapezoidal waveform by computing width, amplitude, rise and fall time of the transient pulse. Basically, for large pulse widths, as described in [122], the transient pulse at the output of a combinational gate could be accurately modeled by $pw + t_f - t_r$ where pw is the input pulse width, and t_r and t_f are the rise and fall delays of the combinational gate, respectively. The resulting transient pulse can be either attenuated or broadened depending on the ratio of rise and fall delays of the gate.

Vulnerable window: An error might reach the inputs of several flip-flops at the same time or at different time intervals. These cases result in significantly different failure probabilities and hence, have to be accurately identified. Therefore, the vulnerable window for each node has to be computed and based on the intersection of vulnerable windows of different flip-flop inputs, various latching possibilities could be investigated. Since the error occurrence time is random, the relative start and end of vulnerable window of nodes are described with respect to the error occurrence time. The relative start time of the transient pulse is set to zero for the error origin (gate A in Figure 3.25), and it is computed for other gates in the forward cone with respect to the time required for error to propagate to those nodes. Considering the pulse width (pw) computed by electrical masking model as well as the relative start time (t_s), a pair of relative start and end time of vulnerable window denoted by ($t_s, t_e = t_s + pw$) could be specified. In the running example, for the error origin, this is $(0, 50)$, as we want to evaluate the error propagation for pulse width of 50 ps. The average propagation delay of inverter gate is 10 ps and the width of transient pulse at the output node is 45 ps according to the electrical masking model, hence, the vulnerable window for the output node would be $(10, 10 + 45) = (10, 55)$.

Once the error is propagated to the inputs of flip-flops, the latching probabilities for different sets of flip-flops are computed. The latching probability could be computed as $\frac{pulse\ width}{clock\ period} \times EP$ as detailed in [123]. In the running example, the error propagates to FF3 with probability of 0.5 during $(40, 65)$ interval, hence, by assuming clock period of 1000 ps, $SCPP^A(50, FF1) = \frac{65-40}{1000} \times 0.5$. In the interval $(65, 92)$, the error is propagated to both FF1 and FF3 flip-flops. In such cases, we have to compute the probability of error logically propagating to each subset of flip-flops. As shown in Figure 3.25, error could be propagated to either FF1 or FF3 or both of them. The error computation for all three cases are shown in this figure. As shown, compared to the case with only one vulnerable flip-flop, the only difference is in the computation of logical masking (EP). For example, the probability that error is logically propagated to both flip-flops is $EP = 0.5 \times 0.05$ which is computed according to input EP of two flip-flops. Once all single-cycle propagation probabilities (SCPP) are computed, all similar cases are merged by summation of similar cases to obtain one unique number for each subset of flip-flops (see Figure 3.25).

In order to show the efficiency of the employed single-cycle analysis, it is compared with an SPICE-based fault injection technique [124]. In this technique, the SPICE netlist of the forward-cone of the target error site is constructed by combining the existing SPICE description of the standard cells. Then the fault injection is performed on the resulting SPICE netlist of the forward-cone. Comparison of the employed analytical techniques with SPICE-based fault injection for 100 randomly selected combinational gates from the OR1200 processor shows average inaccuracy of 2.1% while offering five orders of magnitude speedup compared to SPICE simulations.

3.5 Emulation-based Fault Injection Platform

Once we have analyzed how the errors propagate to flip-flops and SRAM cells, we perform an emulation-based fault injection using our novel platform on those cells to extract the effect of architecture and application masking. Emulation-based fault injection techniques are either reconfiguration-based [42, 43, 44, 45] or instrumentation-based [46, 47, 48]. Reconfiguration-based techniques rely on an internal mechanism of Xilinx FPGAs. These techniques change the configuration bits of the FPGA device using full or partial reconfiguration in such a way that a target fault model is applied on the desired fault site. In these techniques, the reconfiguration process is the speed bottleneck [125].

In instrumentation-based techniques, fault injector circuits called saboteurs are added to each fault site. By activation of each saboteur, a fault is injected in its corresponding fault site. One disadvantage of instrumentation-based techniques is that, unlike reconfiguration-based techniques which are completely dependent on commercial tools, a special tool should be developed for the placement of saboteurs in the design, because none of saboteurs presented so far are supported by commercial tools. The development of a non-commercial tool is a very time consuming task. Additionally, since non-commercial tools do not pass the exhaustive software test process, the possibility of the existence of bugs in such tools is much more than commercial tools. To address this issue, we present an instrumentation-based technique which completely relies on commercial tools for placement of saboteurs.

In this section, we briefly introduce our instrumentation-based fault injection platform which completely relies on commercial tools for the placement of saboteurs. This technique uses two Altera debugging facilities including *In-system Memory Content Editor* (MCE) and *In-system Sources and Probes* (SAP) for fault injection purpose. Since these debugging facilities have no impact on the behaviour of the design, we call the proposed technique, *Shadow Components-based Fault Injection Technique* (SCFIT). Unlike existing instrumentation-based techniques, SCFIT completely relies on commercial tools to implement the saboteurs. Reliance on commercial tools makes the development of the proposed technique much easier and faster than that of other instrumentation-based techniques. SCFIT has the capability of fault injection in both flip-flops and memory units. For fault injection in flip-flops, a scan chain is inserted in all DUE flip-flops and fault is injected in the target flip-flop value during scan chain circulation. For fault injection in memory units, single-port memories are converted to dual-port memories using MCE facility. The additional port is used to write the faulty data at the fault injection time and to read memory contents whenever is needed. In SCFIT, the SAP facility has the responsibility of establishing communication between Quartus II software in host computer and fault injection board inside FPGA board.

Experimental results of OpenRISC 1200 processor evaluation show that the SCIT technique is four orders of magnitude faster than a simulation-based technique. This enables the designers to inject more faults and get more accurate results in a limited time period.

3.5.1 Shadow Components

Altera Corporation provides a complete set of design debugging facilities that can be easily adapted to any design. All these debugging facilities are integrated in the Quartus software to ease the design verification process. These facilities utilize combination of available memory, logic, and routing resources to help the designers in the verification process by increasing the controllability and observability of the design. We have exploited two of these debugging facilities namely In-System Sources and Probes (SAP) and In-System Memory Content Editor (MCE) to propose a fast, flexible and easy-to-develop FPGA-based fault injection technique. These facilities are used to control and observe the value of the control registers and single-port memories in the proposed technique.

A simplified block diagram of a design connected to these facilities is shown in Figure 3.26. As shown, both SAP and MCE are instrumented by a JTAG controller. The JTAG controller acts as an interface between debugging facilities inside FPGA board and the Quartus II software on the host computer.

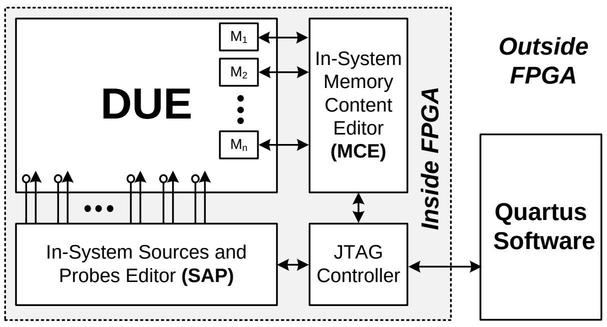

DUE
M1
M2
Mn
In-System Memory Content Editor (MCE)
Inside FPGA
Outside FPGA
Quartus Software
In-System Sources and Probes Editor (SAP)
JTAG Controller

Figure 3.26: MCE and SAP Facilities Connected to a Design in FPGA (Adapted from [5])

The MCE facility allows the designer to easily observe and modify memory contents. This tool converts a single-port memory to a dual-port memory. The Block diagram of an ordinary single-port memory and an MCE dual-port memory are depicted in Figure 3.27. The additional port is connected to the JTAG controller in order to observe and modify the memory contents.

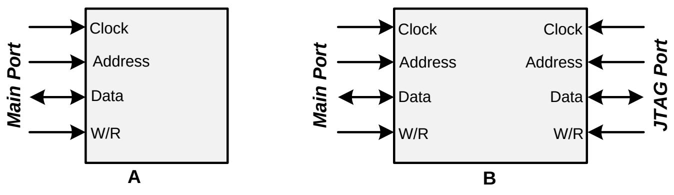


Figure 3.27: A) A Single-Port Memory B) An MCE Dual-Port Memory

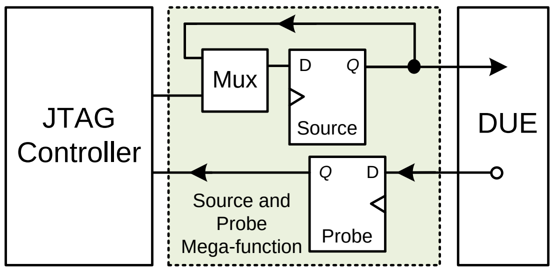


Figure 3.28: An SAP Unit Connected to DUE

The SAP facility is a perfect choice for toggling control signals. Each SAP unit contains several source ports for driving values into and several probe ports for sampling values from the signals that are connected to the ports. Figure 3.28 shows a simplified SAP unit connected to an internal node in the design. As it can be seen, SAP provides single-cycle sample and single-cycle write to selected logic nodes. The SAP allows designers to easily implement control signals in the design as virtual stimuli. By means of this facility, we can observe and change the value of control registers employed in our fault injection board.

3.5.2 Shadow Components-based Fault Injection Technique (SCFIT)

SCFIT uses different approaches for fault injection in memory units and flip-flops. Fault injection in memory units can be easily done using the MCE feature of the Altera FPGAs while a scan chain is employed for fault injection in flip-flops. A simple and general purpose fault injection board is designed to ease the communication between the host computer and the FPGA board.

Fault Injection in Memory Units

As mentioned in the previous section, using MCE the contents of the memory units can be observed or modified at any time. We have used this feature for fault injection in memory units. In this regard, in the fault injection time, the value of target address(es) is read using the MCE feature and after fault injection in the target bit(s), the faulty value is written back to its prior place by the same facility.

Fault Injection in Flip-flops

For fault injection in DUE flip-flops, a scan chain is inserted in DUE. Scan chain is one of the well-known and widely used techniques in design for testability. In this technique, in test mode, all flip-flops in the circuit form a long shift-register, while in the normal mode functionality of the circuit is preserved. By scan chain insertion in DUE, faults can be injected during complete circulation of the scan chain by inversion of the target flip-flop value.

Figure 3.29 shows a fault injection controller designed for SEU injection in DUE flip-flops. A register called Target_FF is used to store the target flip-flip number for injecting SEU fault. While shifting the flip-flops values in the scan chain, a counter is used to enumerate the number of clock cycles. This counter is first loaded by total number of flip-flops. Afterwards, during the shift process, the counter is decremented by one at each clock cycle. Whenever the value of Counter and Target_FF are matched, the fault-free value of the fault site can be accessed at the scan chain output port. By inversion of this value for one clock cycle the value of target flip-flop is inverted. After fault injection, circulation continues until the circulation process is completed. Once the Counter reaches zero, the fault injection process is completed and all flip-flops except the target flip-flop have their previous values.

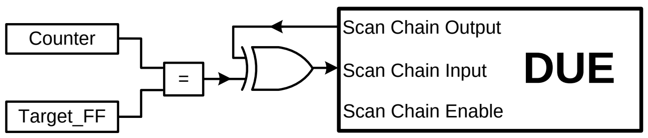


Figure 3.29: Fault Injection Controller for Fault Injection in Flip-flops

For injecting an MBU fault in several flip-flops, one approach is to repeat the operation of SEU injection for each target flip-flop. Another approach is to use multiple Target_FF registers and whenever one of the these matches with counter value, a bit-flip is injected at scan chain input. We prefer the first approach as time needed for multiple circulation is negligible in emulation environment and it has simpler circuitry than the second one.

3.5.3 Experimental Results

As a case study, we have evaluated the soft error vulnerability of the OR1200 processor using SCFIT. Five workloads from MiBench embedded benchmark suite [117] including Bitcount, CRC32, FFT, QSort, and Stringsearch run on OR1200 during the fault injection campaigns.

An Altera Cyclone IV FPGA board is employed as our platform in SCFIT. Table 5.3 shows FPGA resources used by OR1200 in two cases: first, it is normally programmed on the FPGA without any debugging facility; second, it is modified according to SCFIT and then programmed on the FPGA. As it can be seen in this table, applying SCFIT to OR1200 does not significantly affect the allocated resources.

Table 3.6: FPGA resources used by OR1200 with and without SCFIT

Resource	Without SCFIT	With SCFIT	Overhead
Memory bits	712,384	716,274	0.5%
Logic elements	26357	28,002	6.2%

The FPGA board is connected to a host computer with Intel Xeon E5540 2.53 GHz processor and 16 GB RAM. In order to verify the functionality of SCFIT, it is compared with a simulation-based fault injection technique. The simulation-based fault injections are done using the ModelSim simulator on the same computer. Comparison of 1,000 randomly injected faults using both simulation-based technique and SCFIT reveals that the effects of faults in two cases are exactly the same while fault injection in simulation-based platform is much slower than the proposed technique. Table 3.7 shows the average time required for each fault injection for different benchmarks. As it can be seen from this table, each FPGA-based fault injection takes about 1.79-3.73 seconds to be done which is on average four orders of magnitude faster than the simulation-based fault injection. Our experiments in [126, 127] on the non-protected and protected implementations of Leon2 processor showed the same amount of speed-up.

Table 3.7: Runtime of SCFIT vs. simulation-based fault injection [Seconds]

Workload	Cycles	Simulation-based	SCFIT	Speed-up
Stringsearch	209,935	1,476.4	1.79	825X
Bitcount	532,327	5,481.5	1.80	3,045X
QSort	2,065,603	16,836.8	1.99	8,461X
CRC32	11,391,957	66,627.0	2.64	25,237X
FFT	26,137,266	176,883.3	3.73	47,421X
Average				15,998X

3.6 Fault Injection Acceleration

Fault injection experiments is conventionally based on Monte Carlo simulation. Typically fault injection results show a very small vulnerability to soft errors [8, 9, 17]. This means that most of the fault injection experiments have no system-level impact. For a desired level of accuracy, the Monte Carlo simulations could be accelerated to a large extent by exploiting an importance sampling technique which skips most of unnecessary fault injections.

The ACE analysis is able to classify the time intervals to those that might result in failure (i.e. ACE) and those that definitely have no chance to cause a failure (i.e. un-ACE). In our proposed approach, the ACE analysis is leveraged as an importance sampling technique to identify the majority of non-vulnerable intervals. Afterwards, fault injection experiment is only conducted on the ACE intervals to obtain the failure probability. The fault injection sampling space for both conventional fault injection and proposed approach are presented in Figure 3.30. As shown, the proposed approach skips unnecessary fault injection iterations in un-ACE intervals. This technique significantly reduces the number of fault injection iterations for providing the same level of accuracy.

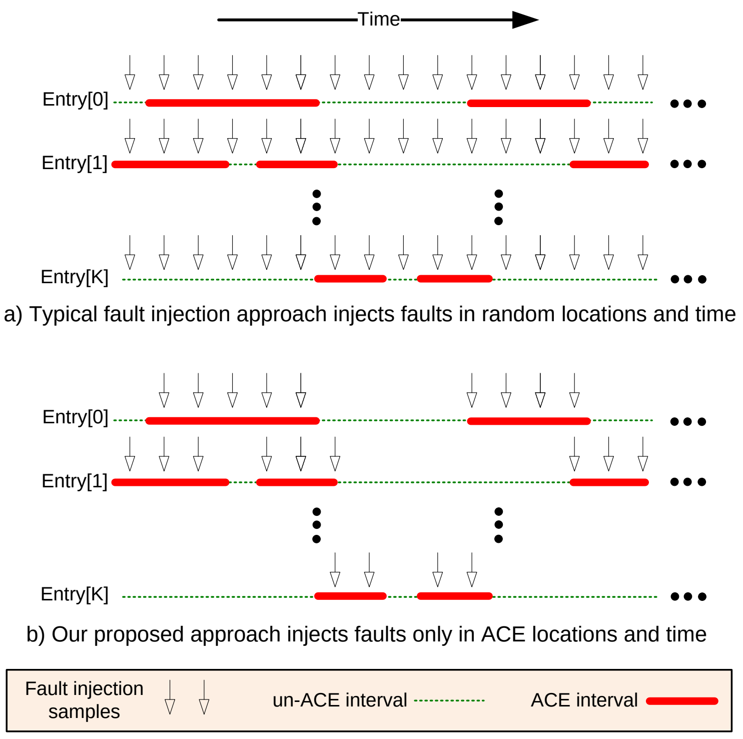

Time
Entry[0]
Entry[1]
Entry[K]
a) Typical fault injection approach injects faults in random locations and time
Entry[0]
Entry[1]
Entry[K]
b) Our proposed approach injects faults only in ACE locations and time
Fault injection samples
un-ACE interval
ACE interval

Figure 3.30: Conventional fault injection vs. proposed approach

3.6.1 Work Flow

The work flow of the proposed approach is illustrated in Figure 3.31. In this approach, a list of fault injection samples is randomly selected first. Afterwards, by performing an ACE analysis, the ratio of ACE intervals to the entire workload execution time (denoted as α) is determined. In addition, the list of fault injection samples is pruned by eliminating samples belonging to un-ACE intervals. Then, fault injection is performed on the pruned fault injection list.

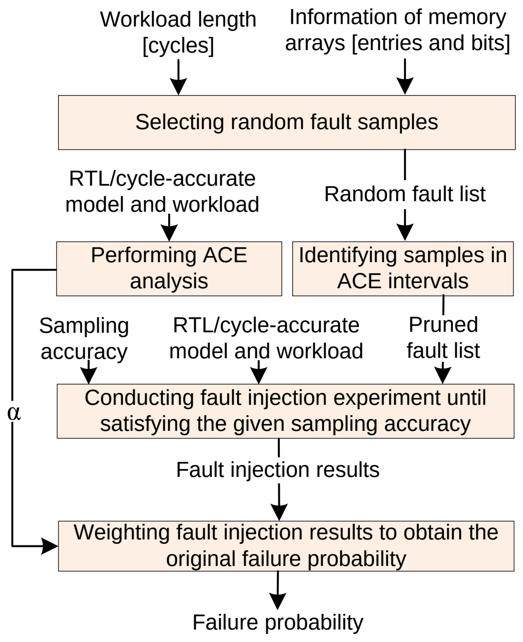

Workload length [cycles]
Information of memory arrays [entries and bits]
Selecting random fault samples
RTL/cycle-accurate model and workload
Random fault list
Performing ACE analysis
Identifying samples in ACE intervals
Sampling accuracy
RTL/cycle-accurate model and workload
Pruned fault list
α
Conducting fault injection experiment until satisfying the given sampling accuracy
Fault injection results
Weighting fault injection results to obtain the original failure probability
Failure probability

Figure 3.31: Proposed flow for fast soft error sensitivity evaluation

3.6.2 Analytical Modeling

In order to obtain the speedup of our proposed technique, we present an analytical model which accepts the ratio of ACE intervals (i.e. α) as input and provides the number of required error injection to reach a desired level of accuracy. The number of samples to reach the maximum sampling inaccuracy of e for failure probability of p and cut-off point of t (depends on the confidence level) in the conventional fault injection approach (i.e. without importance sampling) could be computed as [120]:

$$n = \frac{t^2 \times p \times (1-p)}{e^2}. \tag{3.1}$$

Lets assume that the ACE analysis results show that the memory array is in ACE state with probability of α. In this case, the fault injection has to be performed to evaluate what ratio of the ACE time intervals leads to a failure. However, the failure probability (p') and desired sample inaccuracy (e') are different from that of a typical fault injection. In this case, p' is the failure probability when faults are only injected in ACE time intervals (i.e. $p' = P(fail|ACE)$). According to *Bayes'law* [128]:

$$p' = P(fail|ACE) = \frac{P(ACE|fail) \times P(fail)}{P(ACE)} \tag{3.2}$$

where $P(fail) = p$ and $P(ACE) = \alpha$. In addition, $P(ACE|fail)$ is theoretically equal to one as all failures are related to ACE intervals. Therefore, $p' = p/\alpha$.

In addition, the desired sampling inaccuracy for the fault injection after ACE analysis is not the same. Fault injection on ACE intervals provides the sampling inaccuracy for the results in the range of [0,1], while the share of this results in the final results in the range of [0,α]. Due to the fact that the results of non-ACE intervals have theoretically no inaccuracy, that of ACE intervals could have inaccuracy of e in the range of [0,α], hence $e' = e/\alpha$ in the range of

[0,1]. Since $\alpha \leq 1$, $e \leq e'$. This means that the fault injection only on ACE intervals (proposed approach) requires less accurate results to provide the same level of accuracy on the overall results.

Considering the aforementioned points, the number of fault injection iterations required by our proposed approach is:

$$n_p = \frac{t^2 \times p' \times (1-p')}{e'^2} = \frac{t^2 \times \frac{p}{\alpha} \times (1-\frac{p}{\alpha})}{(\frac{e}{\alpha})^2} = \frac{t^2 \times p \times (\alpha - p)}{e^2} \tag{3.3}$$

The speedup of our approach over conventional fault injection approach could be estimated with respect to the reduction in the number of fault injection iterations according to Equation 3.1 and 3.3. Beside the iterations required for the fault injection, the proposed approach also needs one additional iteration to compute the ratio of ACE intervals. However, fault injection campaigns typically have to provide the results with small inaccuracy which mandates larger number of fault injection iterations. Hence, this additional iteration could be safely ignored in computing speedup (i.e. $1 \ll n_{proposed}$). The speedup of our proposed approach over conventional fault injection approach is computed according to:

$$speedup = \frac{n}{n_p + 1} \approx \frac{n}{n_p} = \frac{1-p}{\alpha - p} \tag{3.4}$$

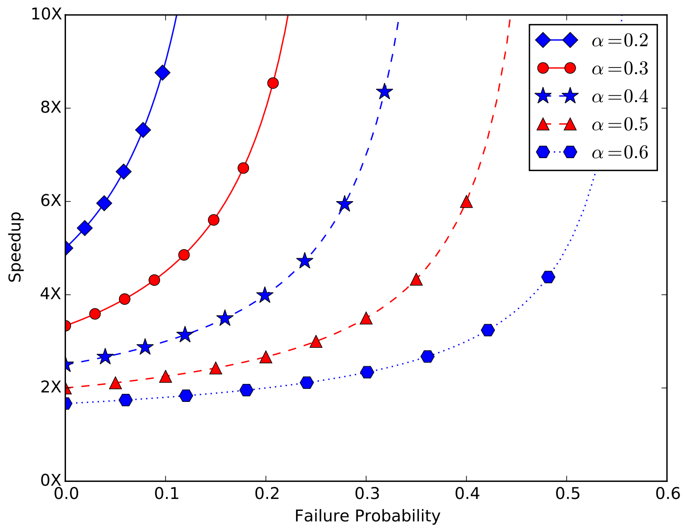


Figure 3.32: Speedup of proposed approach for different value of p and α

Figure 3.32 depicts the speedup obtained for the typical value of α (i.e. 0.2−0.6). As shown, the speedup for small values of p is very close to $1/\alpha$ and as $p \rightarrow \alpha$, $speedup \rightarrow \infty$. However, As shown in the previous studies [65, 66] and also confirmed with our experimental results (demonstrated later in Section 3.6.3), the ACE analysis significantly overestimates the failure probability. This means that the value of p is typically much smaller than α and $speedup \approx 1/\alpha$.

3.6.3 Case Study: Fault Injection on Memory Arrays of Leon3

In order to show the effectiveness of the proposed approach, we evaluated the speedup of this technique for assessing the soft error sensitivity of cache arrays and register-file of the Leon3

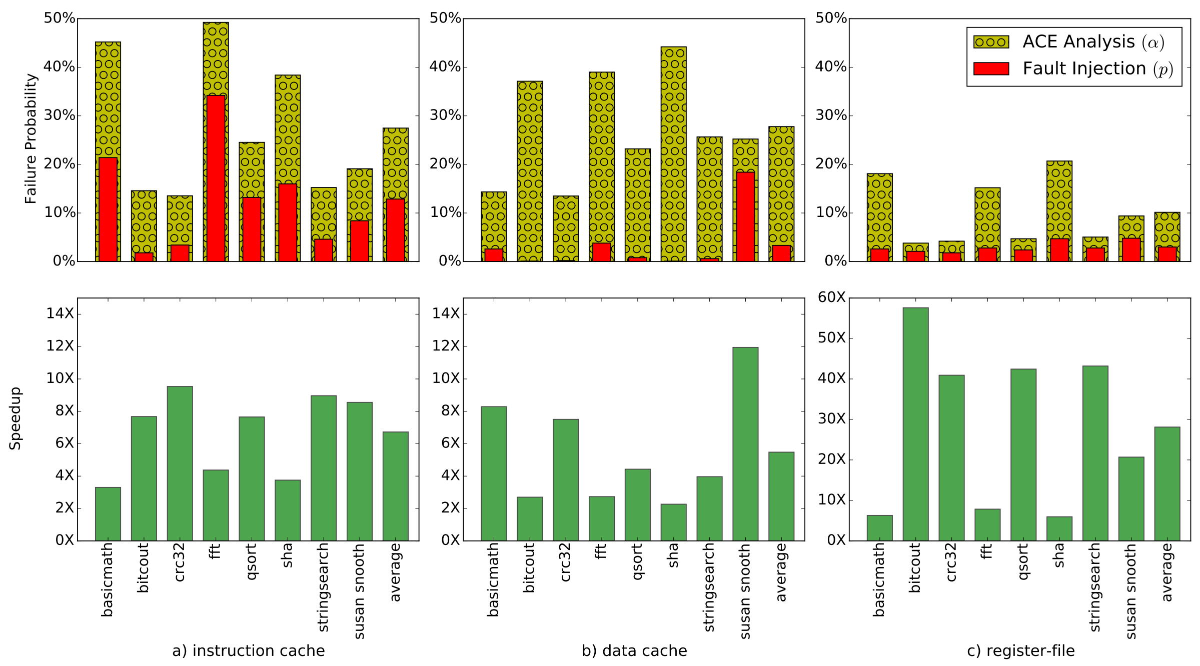


Figure 3.33: Failure probability obtained by ACE and fault injection as well as the speedup of ACE before FI technique for a) instruction cache b) data cache and c) register-file of Leon3 processor

processor. Leon3 is an open-source embedded processor with an in-order 7-stage pipeline unit. This 32-bit processor implements Harward architecture, i.e. it has a separate instruction and data caches. Both data and instruction caches are write-through with direct-mapped policy and have size of 1024×32 bits and their tags are stored in a 128×28 bits memory array. The register-file of this processor has 128 words.

In order to compute ACE intervals, the RTL implementation of the processor is modified to include an ACE model for each on-chip memory array. For register-file, we implemented the generic ACE model while for instruction and data cache, the cache hit/miss status is also taken into account according to the description in [64]. After executing each workload, the value of total ACE intervals for each memory entry is read and the value of α is computed.

Speedup analysis

In order to obtain the speedup of our proposed approach for assessing soft error vulnerability of on-chip memory arrays of Leon3 processor, the runtime of a pure fault injection (i.e. conventional) is compared with that of our approach for the maximum tolerable sampling inaccuracy of 1% ($e = 0.01$) and confidence interval of 95%.

The speedups obtained for evaluating different workloads are illustrated in Figure 3.33. The results reveal that there is a significant variation among different workloads and also between different components. By employing the proposed approach, the soft error sensitivity assessment is expedited for instruction cache, data cache and register-file by 6.7X, 5.5X, 28.2X, respectively (13.5X, on average).

The speedup obtained for register-file is significantly higher than the two other components as generic ACE model is very accurate (i.e. less pessimistic) for this component. This is mainly because all data read from register-file are operands required by the execution unit and have a very high chance to affect the workload output. In contrast, an error in data/instruction cache has to be propagated through several units to reach the execution unit of the processor. Each of these intermediate units masks a subset of the errors which cannot be captured by our employed ACE model and result in a larger gap between α and p in these two components.

Our experimental results are in line with the analytical study provided in Section 3.6.2. For the cases which there is considerable gap between α and p, the speedup is mostly dependent on α, i.e. a smaller value of α results in a higher speedup and vice versa. For instance, in data cache, all workloads excluding *susan_smooth* have speedup close to $1/\alpha$. In contrast, the speedup for the cases with small gap between α and p is strongly dependent on both parameters and is considerably higher than the other cases. This significant difference in speedup could be clearly seen for *susan_smooth* and *stringsearch* workloads in data caches which have almost same value of α, but considerable difference in the value of p. As shown, the speedup for *susan_smooth* workload is considerably higher than *stringsearch*. These results could be interpreted similar to Figure 3.32.

3.7 Conclusions

In this chapter, I presented a comprehensive SER analysis platform which models both error generation and propagation behaviors. This platform consists of a combination of empirical models at the device level, error site analysis at chip layout, analytical error propagation at logic level and fault simulation/emulation at the architecture/application level. By having an appropriate model of abstraction at each step, we are able to accurately propagate the errors and observe their application-level effects.

The proposed platform could evaluate a relatively complex design using its hierarchical modeling methodology. This platform enabled us to accurately model the intrinsic SER as well as all the masking factors for all memory arrays, flip-flops, and combinational gates. The results of SER analysis of an embedded processor using this approach will be presented in the next chapter.

4 Soft Error Analysis of an Embedded Processor

In this section, the in-depth SER analysis of an embedded processor synthesized to a 45 nm technology is presented. This analysis is based on the proposed cross-layer soft error modeling platform introduced in Chapter 3. The results presented in this chapter are for a single processor unit implemented using Nangate 45 nm library, running a set of selected workloads. Under different conditions, the results may vary, however, we believe that most of the major trends and the conclusions hold across different designs and processes, and thus provide useful insights to designers for cost-effective protection of embedded processors.

4.1 Experimental Setup

In this section, the experimental setup for analyzing SER of the OpenRISC 1200 (OR1200) processor using the framework explained in Chapter 3 with respect to a 45 nm technology node is presented. All the experiments are done with respect to the synthesized netlist of the processor, i.e. not the actual chip.

4.1.1 FIT Rate Analysis

The FIT rate analysis tool considers the effect of neutrons and alpha particles. The neutron energy distribution is defined by JEDEC89a standard [129]. The tool uses a nuclear database to determine the secondary reactions that occur when neutrons interact with the atoms in the CMOS structure. Separate simulations for alpha particles were performed, based on the assumption of *Ultra Low Alpha* (ULA) (0.001 alpha/cm^2/hour) packaging materials.

All the cells in the Nangate 45 nm standard cell library were characterized using the FIT rate analysis tool. A generic process response model for a 45 nm CMOS process was used to model the underlying process sensitivity. For the combinational cells, separate simulations were performed with different output loads (0, 1, 3, 6, 10, 20, 40, and 100 fF) and pulse widths in steps of d=20ps. The Nangate library does not contain SRAM cells, therefore, representative FIT rates and MBU patterns for 45 nm SRAMs presented in [39] were used in this study.

4.1.2 OR1200 Processor

The OR1200 processor is an open source 32-bit processor which implements a 5-stage pipeline with Harvard architecture, hardware divider and multiplier, and a floating point unit. Both data and instruction caches are write-through with direct-mapped policy and have size of 1024×32 bit and their tags are stored in 256×21 bit array. It also has a 32×32 bit register-file with one write and two read ports.

The processor was synthesized into the Nangate 45nm library using Synospys Design Compiler. Placement, routing, clock-tree synthesis and post-route optimization were performed using Cadence SoC Encounter. All the experiments in this thesis are done with respect to the typical process corner of the employed library. The final netlist consists of 2,694 flip-flops and 30,986 combinational gates. The maximum operating frequency, after all optimizations was 894 MHz. The clock tree that was built by Cadence SoC Encounter had three levels of inverter/buffer gates. There was one INVX32[1], one INVX32, and 50 BUFX8 in the first, second, and third level, respectively. The average fanout of the leaf-level of the tree is 52. The clock skew and sink transition time[2] for this tree are 13 ps and 181 ps, respectively.

Eight workloads from MiBench benchmark suite [117] were executed on the OR1200 during SER analysis experiments including *stringsearch*, *bitcounts*, *sha*, *susan_smooth*, *qsort*, *crc32*, *basicmath*, and *fft*. The runtime of these workloads varies from 3 to 611 million cycles. The size of the code and input data required for these workloads varies between 48−425 KByte which is reasonable for the employed 8 KByte L1 cache.

4.2 Experimental Results

The results of an in-depth analysis of contributions of different components to the overall SER of the OR1200 processor using our hierarchical SER analysis platform are presented in this section. In addition, we analyze the impact of clock frequency and voltage on the SER of different components. Furthermore, we show the effect of selective protection to mitigate the SER of flip-flops and combinational gates.

4.2.1 Overall SER

Figure 4.1 shows the alpha and neutron SER contribution from all three components for eight workloads. As it can be seen, the alpha and neutron SER are comparable in magnitude. At this point, where no ECC is applied to memory units, the SER is dominated by the memory (>95%) while the combinational contribution is well below 1%. Also, the figure clearly shows the dependence of the SER on the workload with Susan being nearly 3X as sensitive as some other workloads (e.g. CRC32). The reason for higher SER of this workload is the significantly higher vulnerability of the cache units. This workload has a very high miss rate in data and instruction caches. The average time data remains in the cache after its last read access represents the non-vulnerable time. By having higher miss rate data which are not accessed would be replaced with new data very soon and hence the vulnerable time of the cache would be higher.

4.2.2 Memory Protection

It is well known that SRAMs are the largest contributor to the overall SER and in most designs with reliability targets, they are ECC protected. In order to protect the write-through

[1]Xn suffix on the gate name is the drive strength
[2]maximum delay from clock pin to flip-flops

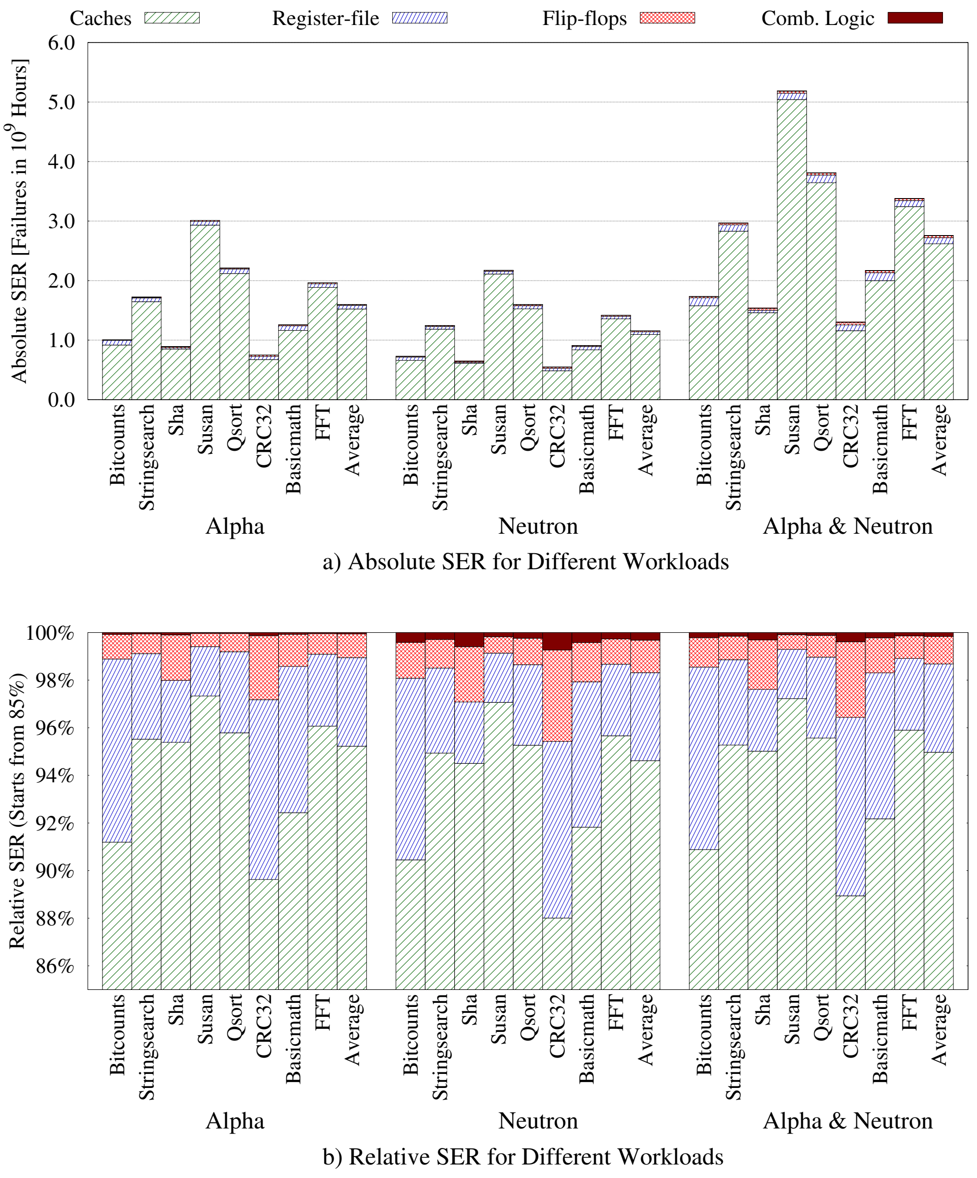


Figure 4.1: Contribution of different types of components on the overall SER

instruction and data cache memory units, a parity bit scheme is implemented. Whenever an error is detected in these memory units, a cache miss is issued and error-free data is recovered from the main memory. For protecting the register-file, a Hamming code with distance three is employed.

Memory interleaving is primarily used to manage the aspect ratio and to improve the timing of memories. However, it can also significantly mitigate the effect of MBUs [130]. Figure 4.2 shows the SER estimation results for different interleaving schemes combined with *Single Error Correction* (SEC). This experiment shows that for an interleaving distance of four (ID=4) the overall memory units SER is virtually zero. This reveals that the effectiveness of SEC codes can be maximized for mitigating MBU errors when used together with an appropriate memory interleaving scheme.

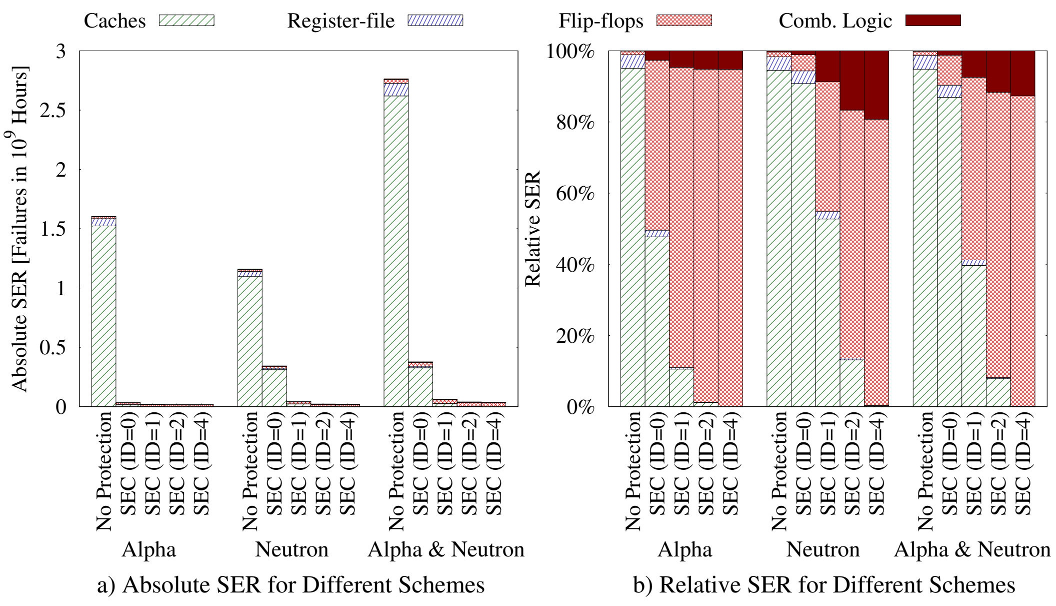


Figure 4.2: Contribution of different types of components on the overall SER where caches and register-file SRAMs have different Interleaving Distances (IDs)

4.2.3 Flip-flop SER

After the memory is efficiently protected, flip-flops become the dominant contributor to the overall SER. Our analysis shows that the SER of flip-flops is non-uniform and over 40% of the flip-flops have a negligible contribution to the overall SER. Although these flip-flops have a considerable raw FIT rate, logical masking prevents errors in these flip-flops from propagating to the workload output. Figure 4.3 shows the effective flip-flop SER assuming the most critical flip-flops are selectively replaced with hardened flip-flops (e.g. DICE, TMR). As it can be seen, protecting 200 flip-flops (less than 10%) reduces the overall flip-flop SER by 72%. This clearly shows the importance of selective protection for flip-flops. Our analysis shows that these top 200 flip-flops are mostly located in pipeline stages.

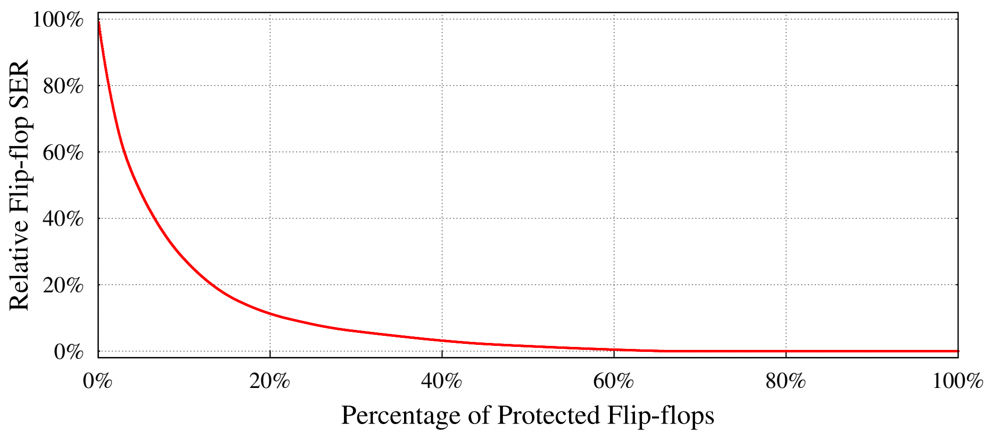


Figure 4.3: Effect of selective protection of the flip-flops on the overall flip-flops SER

In another experiment, we investigated the top 200 vulnerable flip-flops for eight MiBench workloads to see whether the vulnerability of the flip-flops is uniform across different workloads.

Table 4.1: Investigation of top 200 vulnerable flip-flops of eight MiBench workloads

Common in # of workloads	8	7	6	5	4	3	2	1
# of flip-flops	85	42	33	16	14	26	54	106

The results of this analysis are shown in Table 4.1. We observed that 85 of these flip-flops are common to all workloads whereas 42 are common to seven workloads. There are 106 flip-flops which appear in the top 200 list of only one workload. This indicates that on average only $\frac{106}{8} = 13.25$ flip-flops (less than 7% of 200) in each list differ from the others. This data suggests that the fault-grading performed on a few test workloads will generate a list of critical flip-flops that provides reasonably protection for a broad set of workloads, assuming the test workloads exercise all functional units.

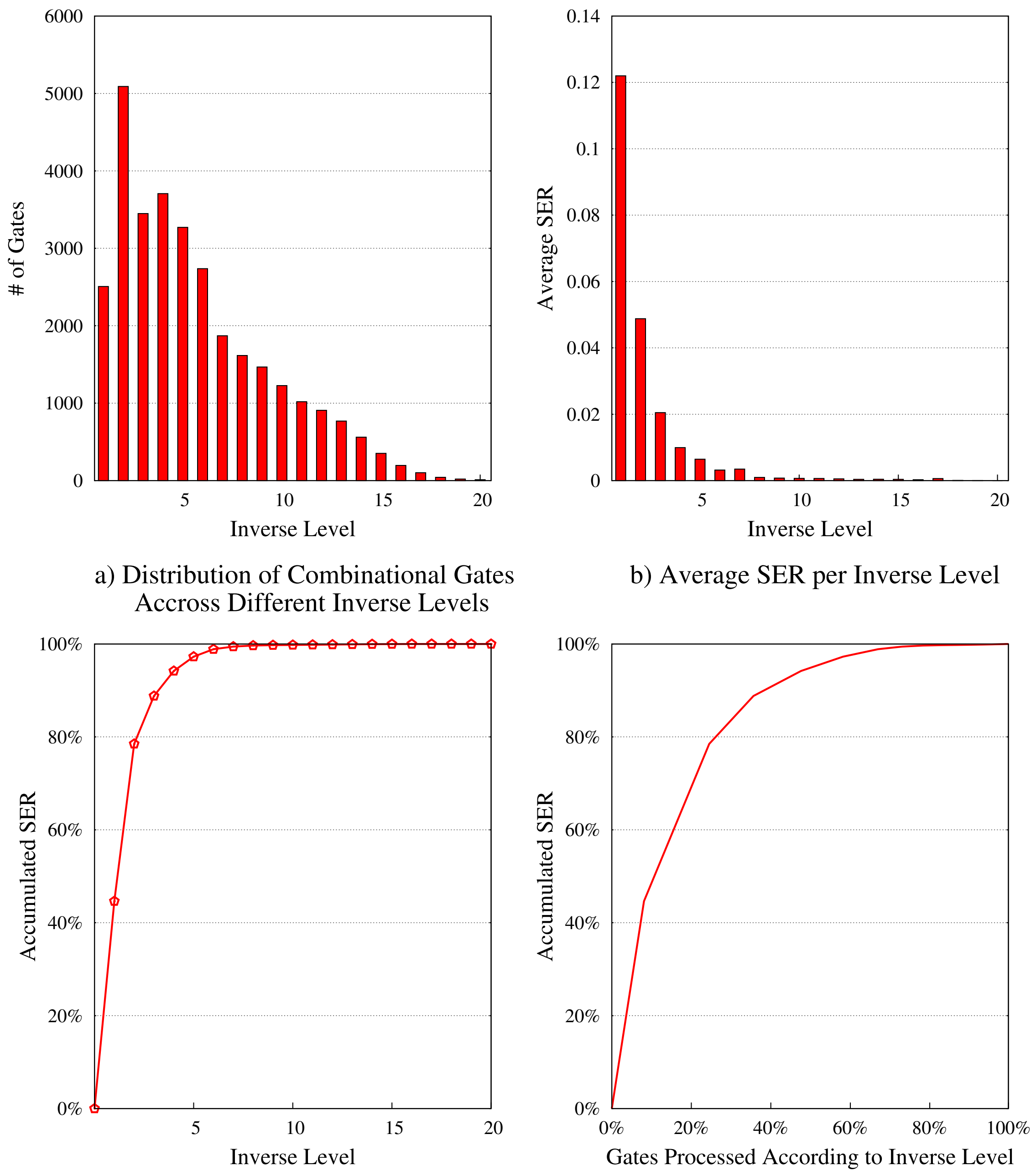
The analysis of contribution of individual flip-flops reveals that protecting only the 85 common vulnerable flip-flops mitigates the overall flip-flop SER by 38.3%. This significant reduction is achieved at a very low cost by protecting only 3.2% of all flip-flops.

4.2.4 Combinational Logic SER

Since the SER contribution of combinational logic in high performance systems with multi-GHz frequencies is considerable, it is important to identify the most vulnerable parts of the logic circuit. Typically, errors generated at logic gates which are closer to the flip-flop inputs are more likely to propagate as they are subject to less logical and electrical masking. In order to show this fact, we analyzed the SER of combinational gates with respect to *inverse level*. The inverse level of a gate is defined as the length of the shortest path (in term of gates) from that gate to flip-flop inputs.

The distribution of gates with respect to inverse level is depicted in Figure 4.4.a. The number of gates with inverse level of one is roughly equal to the number of flip-flops as most flip-flops are driven by a logic gate. The number of gates becomes more for inverse level of two as each gate connected to flip-flop inputs may have more than one input. Afterwards, for higher inverse levels, the number of gates gradually reduces due to re-convergence fanout and short paths.

The average SER of gates belonging to different inverse levels are shown in Figure 4.4.b. This clearly shows that the first few inverse levels have much higher SER and are responsible for the majority of combinational logic SER. On the other hand, the gates with higher inverse levels have negligible SER and could reasonably be ignored in order to reduce the computational load. For example, evaluating only first five inverse levels covers more than 97% of the combinational logic SER (see Figure 4.4.c) while less than 60% of the combinational gate belong to these levels (see Figure 4.4.d). Furthermore, our detailed analysis shows that the most vulnerable gates belong to the clock and reset tree with 9.5% contribution to the overall SER. In addition, similar to the flip-flop analysis, most of the remaining vulnerable gates are in the pipeline stages and belong to the paths which lead to at least one vulnerable flip-flop.



a) Distribution of Combinational Gates Accross Different Inverse Levels

b) Average SER per Inverse Level

c) Accumulative SER of Inverse Levels

d) Accumulative SER versus Percentage of Gates Sorted by Inverse Layer

Figure 4.4: Relation between SER and inverse level of combinational gates

4.2.5 Selective Protection of Combinational Logic

Transient pulses in combinational logic can be mitigated by inserting *transient filters* on the most common error propagation paths. Such filters are designed to mask transient pulses shorter than a given pulse width. We have designed a transient filter according to the technique presented in [104]. The maximum SET pulse width which can be suppressed using the employed transient filter depends on the number of transmission gates. For this work, filters implemented with different numbers of transmission gates are considered and the SER reduction rate is estimated. The results show that a filter with three transmission gate provides the best trade-off between SER mitigation and area overhead. The schematic of this filter is shown in Figure 4.5. It can mask transient pulses shorter than 96 ps and has the propagation delay of 114 ps. It has 10 minimum size transistors and its area is estimated to be twice that of the smallest inverter in the Nangate 45 nm library. Since the filter itself is susceptible to transients, its sensitivity was simulated using the FIT rate analysis tool.

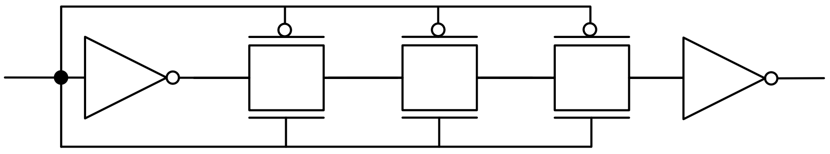


Figure 4.5: Schematic of the employed transient filter

We investigated the SER mitigation due to the insertion of transient filters in flip-flop inputs, memory array inputs and processor output pins. For transient filter insertion, we implemented an iterative procedure. In each step, this procedure finds the candidate site which result in more SER reduction and insert filter on that site and it is removed from candidate list.

Figure 4.6 shows the overall combinational SER for different numbers of inserted transient filters. As it can be seen, by inserting about 600 filters, combinational SER can be reduced by about 80%. This reduces the operating frequency from 894 MHz to 811 MHz (9.3% performance overhead) and imposes less than 4% area overhead for the logic part of the processor.

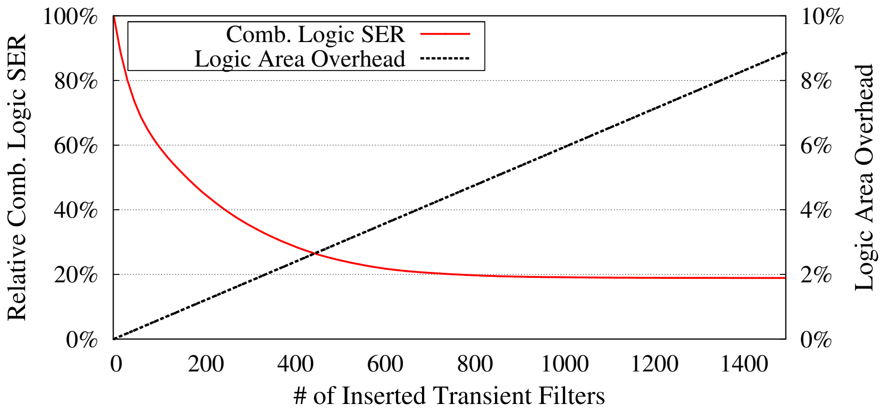
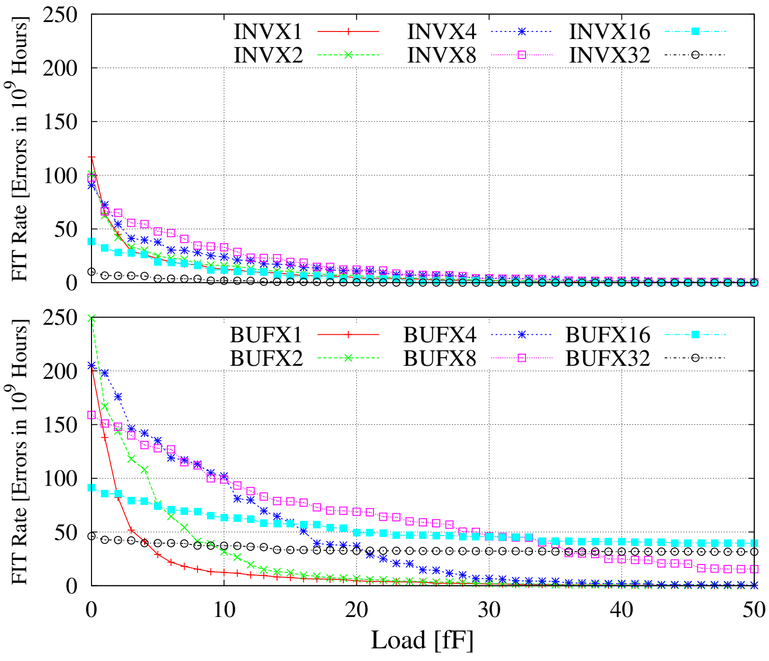


Figure 4.6: Impact of transient filters on SER and area

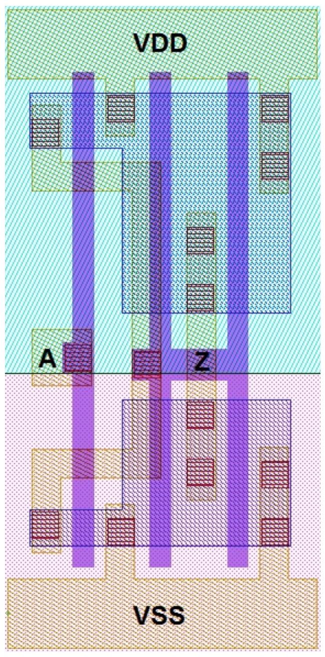
4.2.6 Clock and Reset Tree Contribution

A combinational gate in the data-path is only sensitive to soft-errors during a short time interval when the transient error could propagate to a flip-flop, whereas the gates on a reset or clock-tree are potentially always sensitive. The radiation testing results [22, 23] reveal that the clock tree is a significant contributor to the overall SER.

In order to investigate the validity of such observations for a complex design, we performed an analysis to compute the contribution of the clock and asynchronous reset trees to the combinational logic SER. Before discussing the results of this analysis, it is important to understand the effects of gate sizing and output load on the gate SER. Figure 4.7.a shows



(a) Dependency of FIT Rate on Cell Load



(b) BUFX8 Layout

Figure 4.7: FIT rate vs. load capacitance (Extracted by FIT rate analysis tool)

the total FIT-rate of different inverter/buffer cells for typical load capacitances. As shown, larger gate size and load capacitance result in lower FIT rates for both types of cells. However, unlike inverters, buffers with higher sizing ratio have considerable FIT rate in the presence of load capacitance larger than 40 fF. This can be interpreted with respect to the layout of such buffers. Figure 4.7.b shows the layout of the BUFX8 cell from the Nangate library. Buffer cells consist of two inverters and as it can be seen, sizing is only applied to the second inverter. Consequently, the first inverter has a small load capacitance and the output load capacitance has no effect on the FIT rate of the first cell. However, by increasing the load capacitance of the cell, delay of the second inverter increases and it would be able to electrically mask more pulses from the first inverter. This explains why the FIT rate diagrams for buffers do not become constant after a while and have a slow decreasing rate.

This analysis showed a 2.3% and 7.4% contribution to the overall combinational SER for the clock and asynchronous reset tree, respectively. The small contribution of the clock tree in this design, is due to the fact that the buffers and inverters in the clock-tree were large and had relatively large output loads, thus making them relatively immune to SETs. Furthermore, SETs on the clock tree, only cause an error if they provoke a timing violation or an incorrect sampling of the data. In this design, Cadence SOC encounter chose large inverter/buffers to achieve the required slew rate targets and minimize the clock tree depth. Obtaining a small slew rate is critical to reduce the power consumption due to short-circuit current. Also, clock tree depth determines the sink transition time which is an important factor in the design of clock trees.

Unlike the clock tree, the asynchronous reset tree does not have very tight constraints. It only needs to obey the input to register timing constraints and has to have an acceptable slew rate as specified in the library. The asynchronous reset pin of OR1200 is connected to

1322 flip-flops and we specified an input delay of 400 ps. The synthesis tools generated a reset tree that met the requirements, however, it selected many small gates thus explaining the significant SER contribution.

4.2.7 SER Heat-map

The most sensitive parts of the processor core (combinational and flip-flops) could be studied with respect to the layout of the processor. The SER heat-map of the processor is shown in Figure 4.8. As shown, the cells with relatively high SER clustered in special parts of the layout as they belong to vulnerable units. As we discussed in the previous subsections, for the case of OR1200, these are mostly the cells of the pipeline stages.

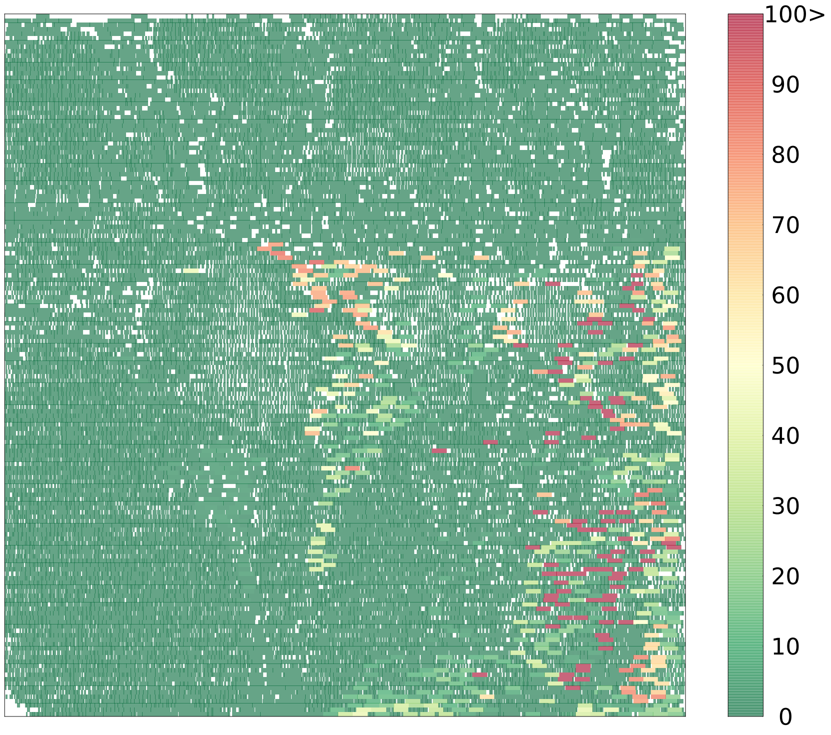


Figure 4.8: Illustration of SER [Failures in 10^{-15} hours] of processor core cells with respect to the layout

4.2.8 SER Dependency on Clock Frequency

Figure 4.9 shows the SER of combinational logic and flip-flops in OR1200 for neutron and alpha particles. As expected, both neutron- and alpha-induced combinational logic SER show a linear relation with clock frequency which is mainly due to decreasing of latching-window masking. In contrast, there is an inverse linear relationship between flip-flop SER and clock frequency because of the increase in the temporal masking factor of flip-flops. This observation is in contradiction to the common assumption that flip-flop SER is largely independent of clock frequency. The major shortcoming of the radiation-based experiments in [20, 21] is that the test circuits are very simple (e.g. flip-flop chain, inverter chain, comparator) and have just one level of flip-flops. Such circuits cannot capture the effect of temporal masking of flip-flops. Our experiments show that such dependency is not only significant, but also dominates the relationship between chip-level SER and frequency.

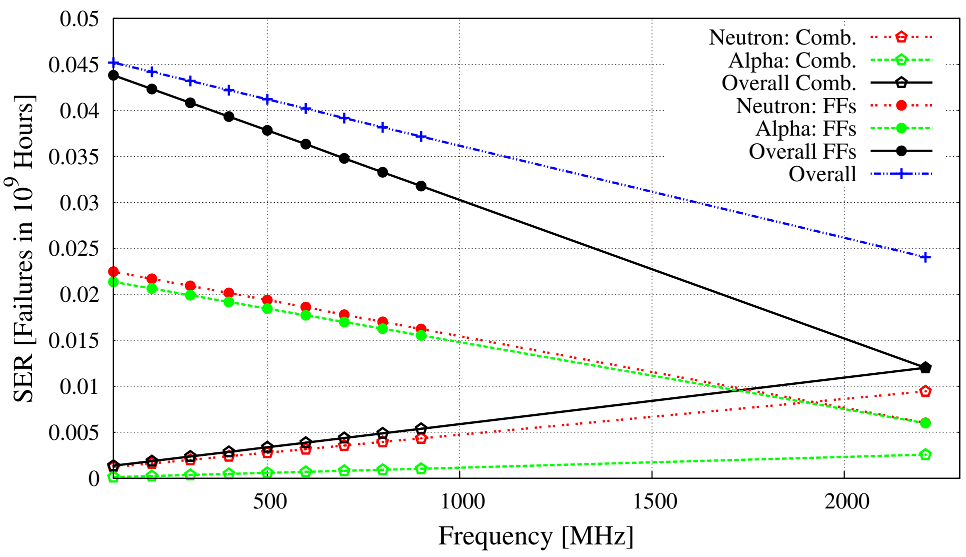


Figure 4.9: Frequency vs. SER

For estimating the threshold frequency at which the logic SER would exceeds that of flip-flops, data extracted from the experiments must be extrapolated. Although this is not an accurate method as the circuit structure would be different for a high-speed design, we believe this gives an indication of the frequency range where such an issue occurs. Using this method, it is seen that the threshold frequency is about 2.2 GHz.

For the results presented in Figure 4.9, the processor is synthesized for the maximum clock frequency of 894 MHz and then the soft error vulnerabilities evaluated when the processor is working at lower frequencies. Another case is to synthesize the processor for the desired operational frequency. In this case, the circuit structure would be different for each frequency. In order to evaluate the SER for this case, we synthesized the OR1200 processor for different clock frequencies and for each of them, the SER was evaluated with respect to that clock frequency. The results of the experiments for these two cases are summarized in Figure 4.10.

The changes in combinational logic SER of the designed for specific frequency case (Figure 4.10.a) could be explained with respect to the distribution of drive strength of gates (Figure 4.10.b) for each synthesized core. Since small gates are more sensitive to SETs, it is expected that if the target design has more X1 gates, it has an increased combinational SER contribution. This explains the high soft error rate of combinational logic for frequency range of 100-400 MHz. On the other hand, another factor which has a direct impact on the combinational logic SER is the number of combinational logic gates. According to our analysis, in the frequency range between 400 and 800 MHz, the increase in the number of gates is primarily in the pipeline stages which we have observed to be the most sensitive region. While in the range between 800 to 900 MHz the additional gates are in less critical sections of the design. This explains the higher slope in the range between 400 and 800 MHz compared to 800-900 MHz. Below the frequency of 400 MHz, the changes are quite linear as the synthesis tool cannot find a lower area implementation.

When the circuit is synthesized for looser timing constraints, the length of paths would be

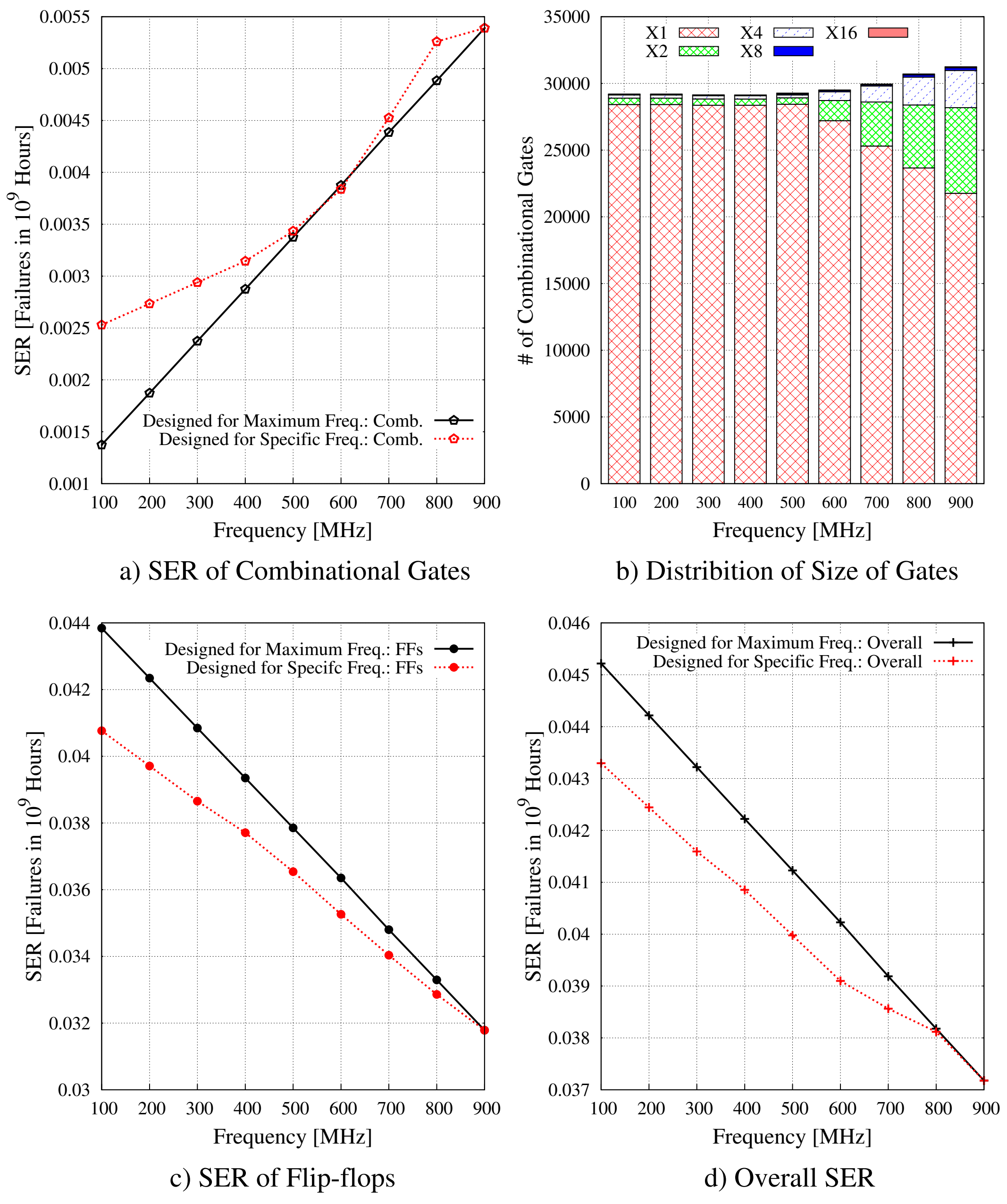


Figure 4.10: Comparison of Frequency vs. SER for two cases: Designed for maximum frequency: the processor is synthesized for maximum frequency, but operates at lower clock frequency; Designed for specific frequency: the processor is synthesized for the operational frequency

longer. Longer paths result in more temporal masking and hence, the soft error vulnerability of the flip-flops becomes smaller compared to the case where the frequency is scaled without changing the circuit structure. This is clearly demonstrated in Figure 4.10.c.

The combined effect of combinational and sequential SER is shown in Figure 4.10.d. As it can be seen, typically the overall SER is smaller when the circuit is optimized for the target frequency.

4.2.9 SER Dependency on Dynamic Voltage and Frequency Scaling

The Dependency of soft error rate on the supply voltage has been widely studied at the circuit-level [1, 131]. However, there has been limited studies done on the effect of voltage scaling on the SER of complex circuits. To address this, we conducted an experiment to compute SER of the processor core for ±10% and ±20% of the nominal voltage. The original design has the nominal voltage and the frequency of 1.0 V and 894 MHz, respectively. For each voltage, the operational frequency and delay of the gates are adjusted accordingly.

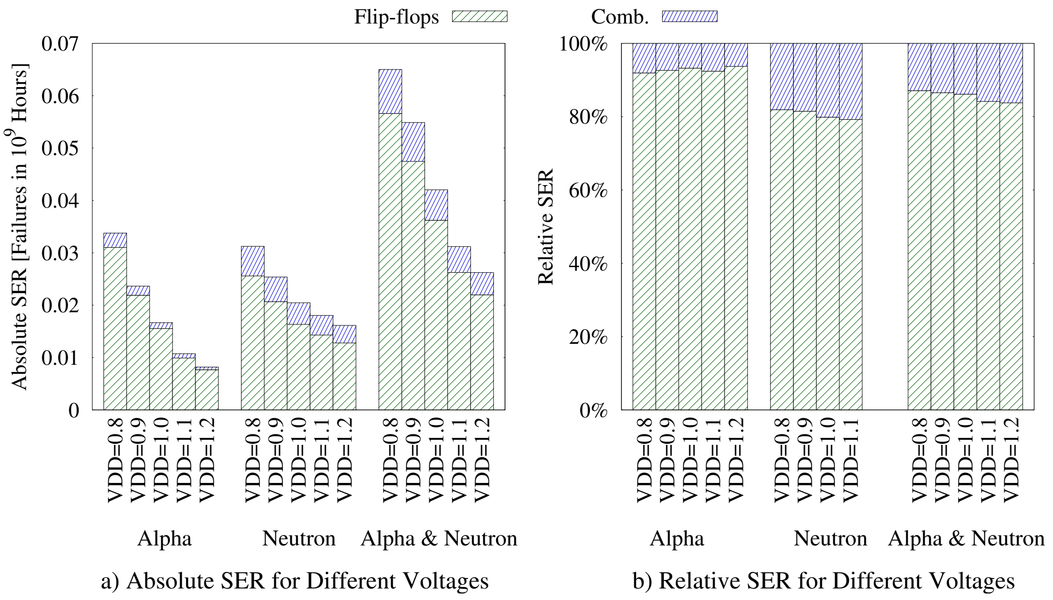


Figure 4.11: Impact of DVFS on SER

The results of SER analysis for different voltage and frequency points during *Dynamic Voltage and Frequency Scaling* (DVFS) are summarized in Figure 4.11. For each point, the intrinsic cell FIT rate was taken from the library characterization for that supply voltage and the propagation is performed according to the propagation delays of the gates at that voltage. As shown, increasing the supply voltage significantly reduces the soft error vulnerability of both flip-flops and combinational gates, however, the ratio of combinational logic SER to that of flip-flops remains nearly constant. Our detailed analysis shows that reducing voltage increases the propagation delay, but since the clock period is also increased with the same pace to meet the timing constraints, the overall ratio of combinational and sequential SER remains the same. Also, it is observed that the Alpha-particle induced SER is affected more during DVFS.

4.3 Conclusions

In this chapter, I presented a comprehensive SER analysis for an embedded processor running realistic workloads using an efficient hierarchical modeling technique that enabled us to accurately model the intrinsic SER as well as all the masking factors up to the application level. The key results obtained by analyzing this processor are:

- The effectiveness of selective protection: After ECC is employed, the flip-flops dominated the SER and we showed that selective mitigation of less than 10% of the flip-flops reduced their SER by 70%. It is also observed by employing an appropriate selective transient filter insertion technique, the combinational SER can be reduced by 80% with only 4% increase in the logic area.

- SER dependency on voltage and frequency: Interestingly, we found that the overall SER decreases with the clock frequency due to the reduced SEU temporal masking. Also, it is observed that the DVFS has considerable impact on SER, however, the ratio of sequential and combinational SER remains essentially constant.
- Clock and reset tree SER: Our results showed that the clock tree was not a significant contributor to the overall SER due to the large size of the clock buffers.

These results can assist circuit designers to employ effective hardening techniques to reduce the SER with minimum impact on other design constraints.

5 Soft Error Mitigation Techniques

As mentioned in Chapter 2, the contribution of MET and MBU to the overall SER is growing in advanced technology nodes. In order to address this issue, in this chapter, we provide three techniques to mitigate these kind of errors in VLSI circuits. The first technique is a layout-based placement technique to mitigate MTs in logic circuits. The second technique is a low-cost error detection technique for detecting MBUs in embedded memories. The third technique aims to correct MBUs in configuration bits of FPGA devices where a transient error has a permanent impact.

5.1 Layout-based Multiple Transient Mitigation

As shown in Fig. 3.12 and confirmed by previous studies [2, 40, 35], it is highly probable that one strike affects multiple adjacent cells in the layout. The experimental results in [132] show that the placement strategy could significantly affect the SER of a circuit. Thus, one can employ various optimizations of cell placement to significantly alleviate the effect of MTs. To this end, we propose a generic approach to identify the most vulnerable pairs of cells and subsequently increase their physical distance in the layout.

5.1.1 Opportunities for MT-aware Placement Optimization

The proposed MT-aware placement is based on the following optimization opportunities:

1) Non-uniform cell density: Since the routing is the limiting factor in irregular structures (i.e., random logic) fabricated with nanoscale technology nodes [133], there are always some *whitespace* (i.e. unused space) among cells in the layout. Hence, the layout *utilization ratio*[1] is typically between 65%-85% [111]. Existing commercial place & route tools (e.g. Cadence SoC Encounter) uniformly place the cells across the die, however, the sensitivities of the cells to soft errors are quite non-uniform. The non-uniformity of MT failure probability on the physical layout of b18 and b19 benchmark circuits are depicted in Figure 5.1. This is obtained by injecting one million errors at random locations of the layout (similar to our experiments in Section 3.2.3). Since nodes with very small failure probabilities have much less contribution to the overall SER, it makes sense to reduce the spacing between less vulnerable cells and exploit this space to increase the spacing among highly vulnerable cells. In other words, the cell spacing should be non-uniform and proportional to the MT soft error vulnerability.

2) Error masking potential of MTs: Our experimental results in Section 3.2.3 showed that failure probability obtained based on the netlist-based adjacency assumption is on average 22.34% less than that estimated by the layout-based approach. The main reason for the smaller failure probability of the netlist-based approach is that the errors propagated from

[1]The ratio of overall area occupied by cells to the entire chip area

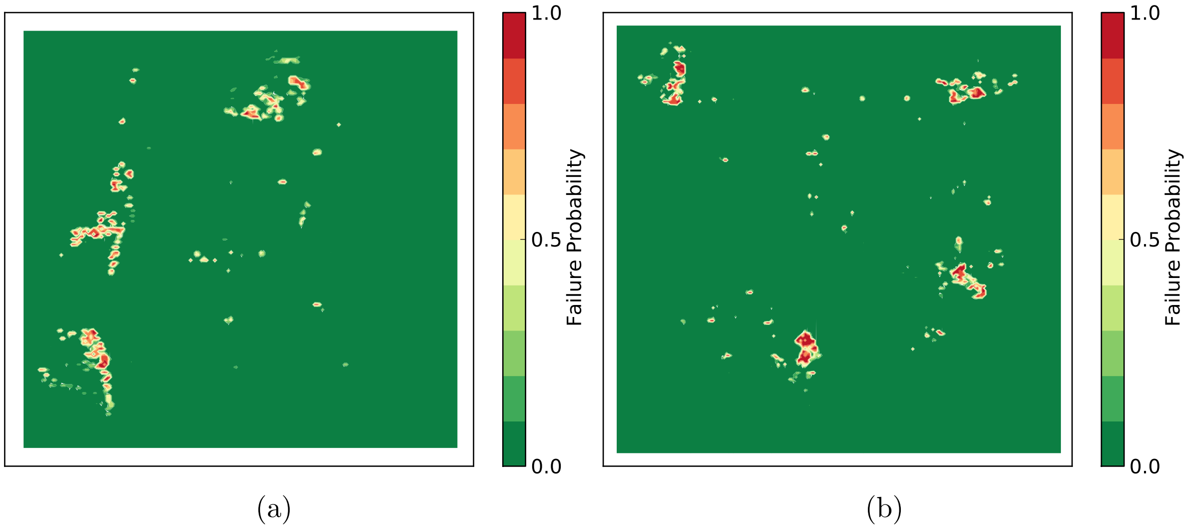


Figure 5.1: Heatmap of MT failure probability for the layout of two benchmark circuits for (a) b18 benchmark circuit and 22 MeV (b) b19 benchmark circuit and 144 MeV

adjacent cells have chance to interact and cancel out each other (i.e. getting masked). For example, in the gate-level netlist shown in Fig. 5.2.a, cells A and D have no common gates in their forward cones. Hence, the failure probability of MT affecting these two cells is $f_{AD} = 1 - ((1 - f_A)(1 - f_D))$. However, when there is an overlapped region in the forward cones of MT error sites, errors might interact with each other. In this case, it is possible that the effect of errors become less or more depending on the circuit structure and the running workload. For example, errors propagating from cells A and B may interact in cell C and the propagated errors to the output of this cell has significantly shorter pulse width than the errors generated at error sites. This phenomenon is known as *pulse quenching* and previous experiments showed that it is more probable specially for cells with GFI or GFO adjacency [134, 135]. Therefore, it is beneficial to identify netlist adjacent cells that could result in such error propagation behaviors and make them physically closer.

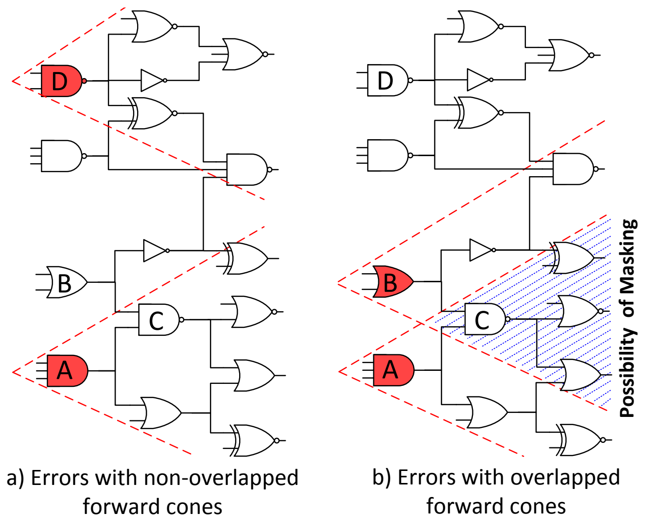

D
B
C
A
a) Errors with non-overlapped forward cones
D
B
C
A
Possibility of Masking
b) Errors with overlapped forward cones

Figure 5.2: Examples of errors with and without common gates in their forward cones

Based on the aforementioned optimization opportunities, we identify the pair of cells which could result in the MT masking and reduce their distance to mitigate MTs.

5.1.2 MT-aware Placement Flow

In a typical place and route scenario, first an initial placement is obtained based on the netlist structure by exploiting a *trial routing* [133]. Although it is possible that the locations of some cells slightly change during the next phases (e.g., clock tree synthesis, timing optimization, and scan chain insertion) to resolve the congestion or timing violations, most of the cells remain in the same location. Thus, if MTs are mitigated in the initial placement, a similar effect is expected on the final layout as well.

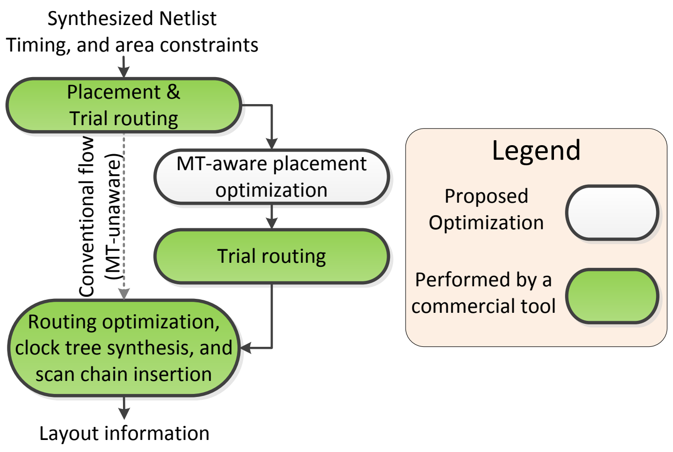

Synthesized Netlist
Timing, and area constraints
Placement & Trial routing
MT-aware placement optimization
Trial routing
Conventional flow (MT-unaware)
Routing optimization, clock tree synthesis, and scan chain insertion
Layout information
Legend
Proposed Optimization
Performed by a commercial tool

Figure 5.3: Flow of the proposed MT-aware placement approach

The flow of our proposed MT-aware placement technique is depicted in Figure 5.3. In this technique, the initial placement is first generated using a commercial place & route tool (e.g., Cadence SoC Encounter) by performing the trial routing. Then, local adjustments in the locations of the cells are performed to mitigate MTs, using the proposed algorithm. Afterwards, the modified placement is given back to the commercial tool to repeat the trail routing according to the modified cell locations. Then, the remaining layout generation steps are performed in the conventional way using a commercial tool. Using this approach, the final layout satisfies all the constraints and have smaller SER.

MT-aware Placement Algorithm

In the conventional design flow, the standard cells have a constant size in at least one dimension that allows them to be lined up in rows on the layout. The layout consists of a large number of rows, each of which has a power and a ground line next to it and contains various cells as well as some whitespace. The proposed MT-aware placement technique accepts an existing placed design as an input and optimizes the location of cells within the row to reduce the average MT failure probability.

The proposed MT-aware placement technique neither moves the cells among different rows nor changes the order of cells in a row, rather it redistributes the existing whitespace in each row according to the vulnerability of the cells. The steps taken in the proposed MT-aware placement optimization technique are In this regard, we compute the disjoint failure

Algorithmus 2 : MT-aware placement optimization

```
1  Read a layout from input;
2  for each row in layout do
3      for each adjacent cell pair i and j in row do
4          $f_i, f_j$, and $f_{ij}$ ← Compute joint and disjoint failure probabilities;
5          $e_{ij}$ ← extract whitespace between cell $i$ and $j$;
6          if $f_{ij} < 1-(1-f_i)(1-f_j)$ then
7              $white += e_{ij}$;
8              $e_{ij} \leftarrow 0$;
9          else
10             $e_{ij}$ marked as ↑;
11         end
12         Redistribute $white$ among all $e_{ij}$s marked as ↑;
13         Change locations of cells in $row$
14     end
15 end
16 Write a modified layout to a file;
```

probability f_i, i.e. when only one cell is erroneous, for each cell i and joint failure probability f_{ij} considering when both adjacent cells i and j are erroneous. For computing both joint and disjoint failure probabilities, the errors are propagated in the netlist according to the methodology described in Section 3.4 and Section 3.3. In other words, the selection of the error sites for joint failure probability is based on the layout adjacency while error propagation is performed on the netlist-level.

According to Section 5.1.1, if $f_{ij} < 1-(1-f_i)(1-f_j)$, there is a possibility of error masking when both cells are affected, hence, it is beneficial to remove the whitespace between the cells which satisfy this condition to increase the probability of such MT (line 6−9). The eliminated whitespace have to be redistributed among the cell pairs which does not satisfy this condition as they introduce no masking. An example of whitespace redistribution is shown in Figure 5.4. This whitespace redistribution reduces the probability of such MTs in the optimized layout. After analyzing all the rows in the layout and redistributing the whitespaces in the favor of MT mitigation, the modified layout structure is generated.

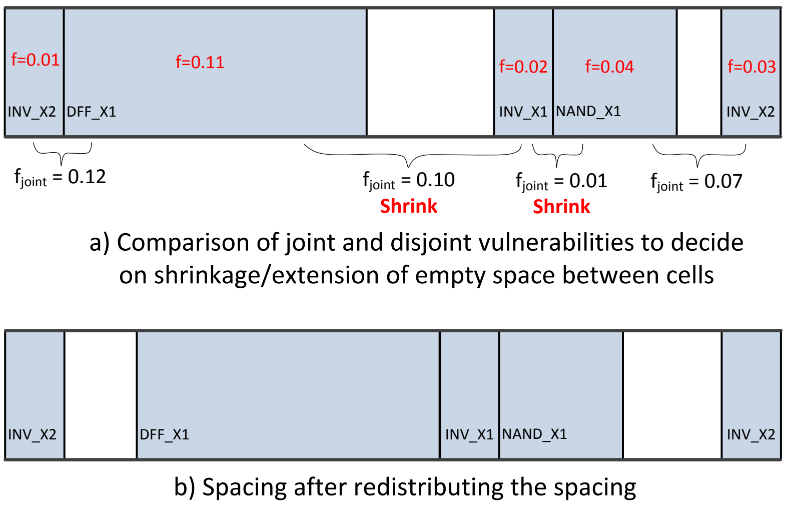


Figure 5.4: Example of spacing before and after applying our MT mitigation technique

In this work, we only adjust the locations of the cells within each row and we do not perform cell swapping across multiple rows. The main reason is to be compatible with the commercial tools. In this regard, the changes with respect to the original layout generated by the traditional placement flow must be both small and local, otherwise, the routing phase will have a considerable runtime and might not be able to find a solution for given area and timing constraints in a reasonable runtime.

5.1.3 Experimental Results of MT-Aware Placement

In order to show the effectiveness of the proposed MT-aware placement technique, the layouts of several large circuits are optimized using this technique. In this section, the SER reduction, overheads, and runtime of this technique are presented. The process of preparations of the benchmark circuits and employed tools are similar to Section 3.2.3.

MT Mitigation

It is very important to notice that the ultimate objective of the MT-aware placement is to reduce the failure probability to be equal to the susceptibility to single transient errors (i.e., $f_{SET/SEU}$). In other words, the target is to reduce the gap between failure probability of MT with that of SET/SEU which is broadening with technology scaling. Hence, we introduce a metric called *MT ratio reduction* as follows:

$$MT\ ratio\ reduction = \frac{f_{MT\text{-}unaware} - f_{MT\text{-}aware}}{f_{MT\text{-}unaware} - f_{SET/SEU}} \tag{5.1}$$

where $f_{MT\text{-}unaware} - f_{SET/SEU}$ is the failure probability reduction potential (maximum achievable) and $f_{MT\text{-}unaware} - f_{MT\text{-}aware}$ is the actual reduction achieved by our technique. Based on this metric, 100% MT ratio reduction means that the SER of the system is equal to SET/SEU rate.

The failure probability of the optimized layout is computed using MC simulation similar to the original one, i.e., by placing the ovals in random locations, extracting the list of affected cells, and propagating errors. It worth to mention that the list of error sites is generated again for the optimized layout as the physical locations of the cells change during the optimization.

Figure 5.5 shows the MT ratio reduction for the employed benchmark circuits synthesized for both Nangate 45 nm and 15 nm standard cell libraries. In this figure, the results for four particle energies (i.e., 22, 37, 95, and 144 Mev) are presented. As expected, by increasing the particle energy, the benefits of our proposed technique becomes less. However, since the particles with smaller energy are much more plentiful, the overall mitigation is mostly determined with respect to the reduction in particles with smaller energies. Also, it could be observed that the reduction ratio in 15 nm is slightly less than that of 45 nm. This could be explained by the average number of affected cells at these two technology nodes. As shown in Figure 3.17.c, the average number of affected cells almost is doubled for all particle energies from 45 nm to 15 nm. Since the failure probability is computed with respect to all cells and our technique mitigates the errors for a subset of the cells, the failure probability reduction decreases by increasing the number of cells.

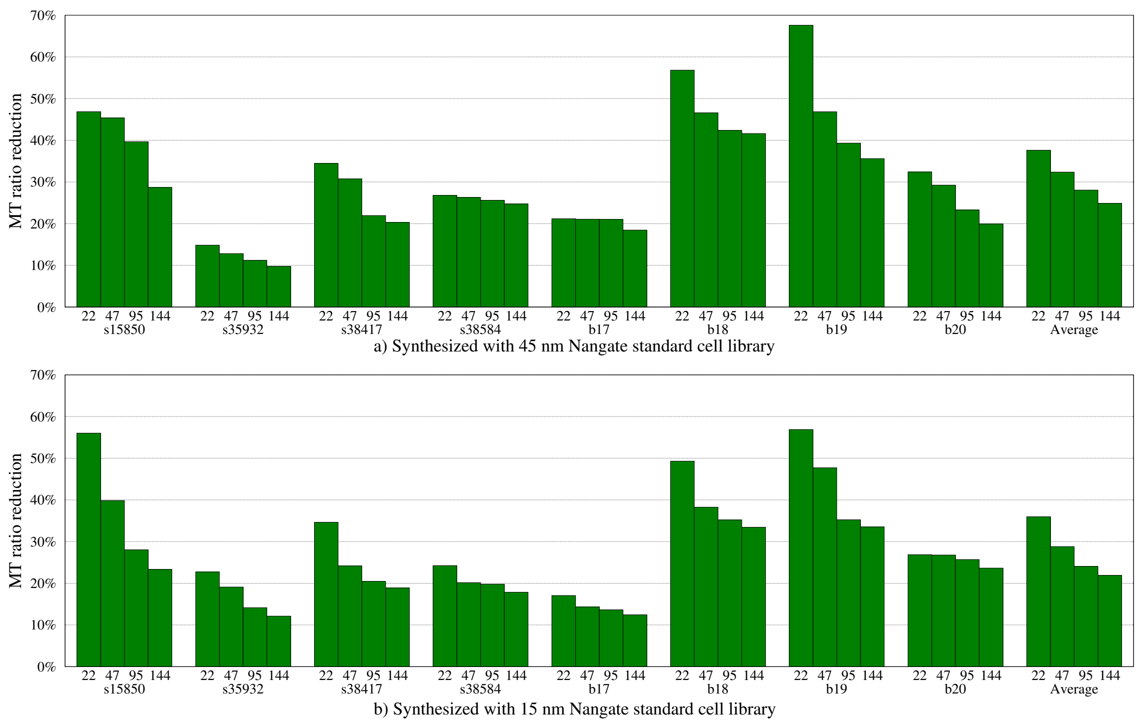


Figure 5.5: MT failure probability reduction for benchmark circuits synthesized with respect to a) 45 nm and b) 15 nm Nangate standard cell library

Overheads

The proposed technique only redistributes the existing whitespace in each row, hence, does not impose any area overhead. Additionally, it satisfies the given timing constraints similar to the MET-unaware placement. This is mainly because the locations of the cells are not fixed in MT-aware placement and could be changed when some of the constraints are not satisfied. During the experiments, the maximum operational frequency is obtained in MT-unaware approach for all benchmark circuits. Then, by putting the same timing constraints,

Runtime

The proposed technique not only increases the runtime of the placement phase due to re-distribution of the whitespaces, but also affect the runtime of the routing phase as trial routing has to be repeated after adjusting locations of the cells. The runtime of the place & route process for MT-unaware (conventional) and the proposed MT-aware placement are reported in Table 5.1. The MT-aware placement has 27.8% longer runtime compared to the conventional approach, however, as it can be seen, this overhead is relatively constant for small and large circuits, hence, it is scalable to industrial-size circuits.

5.2 Low-Cost MBU Detection in Embedded Memories

The most common technique to mitigate MBUs is to physically interleave several words in a way that each MBU affects one bit of several words rather than several bits in the same

Table 5.1: Comparison of runtime MT-unaware (conventional) and MT-aware Place & Route (P&R) [Seconds]

Benchmark	MT-unaware P&R (SoC Encounter) Complete Execution	MT-aware P&R Initial P&R	MT-aware P&R Our Optimization	MT-aware P&R Final P&R	Runtime Overhead
s15850	189	102	3	134	26.5%
s35932	610	329	17	469	33.6%
s38417	614	297	23	362	11.1%
s38584	657	333	31	532	33.3%
b17	858	492	29	601	20.7%
b18	2,243	1,215	173	1,516	29.5%
b19	4,783	2,529	309	3,602	34.6%
b20	3,545	2,199	110	2,410	33.1%
Average					**27.8%**

word. The combination of physical interleaving and single error correction could significantly reduce the SER [18, 130], however, such a technique is not applicable to all memories due to the aspect ratio (i.e. floor-planning), the access latency, and power consumption issues [130]. Therefore, sophisticated error detection and correction techniques are required to harden such memory arrays against MBUs.

The MBU correction in memory arrays has been extensively studied over the past few years. However, there exists some cases where error detection is sufficient to guarantee the correct functionality of the system, i.e. costly and sophisticated error correction is not needed. For example, upon a detection of an error in a cache with the write-through policy, the erroneous block could be invalidated to recover the correct contents from the lower level cache by issuing a cache miss [88]. Another example of such cases are erasure code-based error correction techniques [84, 85] where error detection and recovery are completely isolated for achieving lower costs. In such scenarios, a low-cost MBU detection technique becomes an indisputable part of the reliable system design.

The *virtually interleaved parity* technique has been employed in several studies [84, 83, 85] to detect MBUs where physical interleaving is not possible. In this technique, each word has several parity bits and the bits with a constant distance form a parity group. If the number of parity bits would be large enough, this technique is able to detect all MBUs. However, the area and power overheads of this technique increases in direct proportion to the largest MBU size [84] which is growing by the technology downscaling [3].

In this section, we introduce an error detection technique called *stepped parity.* Unlike the interleaved parity, in the proposed technique each data bit is involved in the computation of several parity bits. The parity bits are organized in a way that each MBU incident could be detected by at least one parity bit which results in a higher error detection capability. The experimental results with respect to MBU patterns obtained for a 45 nm SRAM array show that the stepped parity technique provides higher error detection capability compared to the virtually interleaved parity technique while it imposes considerably smaller area and power overheads.

5.2.1 Stepped Parity

Main Idea

An MBU incident affects the memory cells in a localized manner, hence, the maximum distance of the affected bits is bounded by a certain number which is technology dependent [3, 32]. In addition, the probability of having MBUs with smaller number of cells is typically higher as the particles with smaller energies are typically much more plentiful [136]. Thus, a cost-efficient error detection technique has to put more effort for detecting most common small MBU patterns.

In parity-based error detection techniques, if an even number of bits in one parity group is affected by an MBU incident, that MBU cannot be detected by that parity group. Therefore, an MBU incident is detected if and only if it affects an odd number of bits in at least one parity group. In virtually interleaved parity, each bit of the data is covered by one and only one parity group. The virtually interleaved parity with n parity bits cannot detect some of the MBU patterns spreading over more than n bits. Some examples of such patterns for virtually interleaved parity with two parity bits are shown in Figure 5.6.a. As it can be seen, these parity bits have two affected bits in one word and have considerably high occurrence probabilities. Therefore, the virtually interleaved parity fails to detect such MBU patterns (see Figure 5.6) as it is not designed to detect patterns with higher occurrence probability.

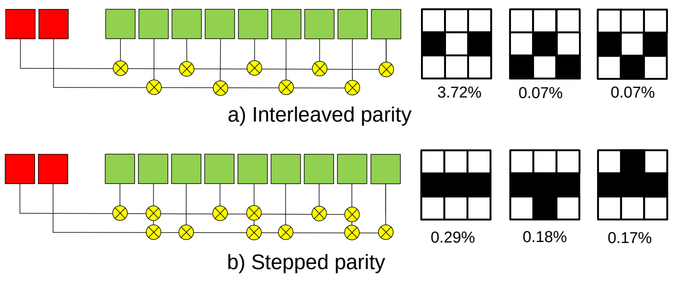


Figure 5.6: The parity organization and most frequent undetectable MBU patterns for a) interleaved parity and b) stepped parity with two parity bits

The main idea behind the proposed stepped parity technique is to associate several parity bits to each bit in a way that each MBU incident could be detected by at least one of these parity bits. Figure 5.6 illustrates the organization of parity bits for virtually interleaved parity and stepped parity with two parity bits. As it can be seen, in the stepped parity, there are data bits which are associated to both parity bits. Although in the stepped parity adjacent cells are associated to the same parity bit, all double adjacent errors could be detected according to the non-common parity bit. This technique is also able to detect double erroneous cells with one unaffected cell in between, whereas interleaved parity fails to do so. In contrast, the interleaved parity could detect three adjacent errors while such an error affects even number of bits in both parity bits in the proposed stepped parity and hence is undetectable. The most common undetectable MBU patterns and their corresponding occurrence probabilities for both error detection techniques are shown in Figure 5.6. As it can be seen, the patterns that cannot be detected by the interleaved parity has considerably higher overall occurrence probability (i.e. ~6X) compared to that of our proposed stepped parity technique.

Formal Presentation

The stepped parity could be uniquely represented as $S(P,T)$ where P is the number of parity bits and T is the *parity thickness* of the code. In this technique, the number of parity bits associated to each bit is defined as parity thickness of that bit. The parity thickness of the code is defined as the maximum parity thickness of all data bits and represents the maximum number of parity bits associated to each bit and could be in the range of $1 \leq T \leq P$.

In this technique, the i^{th} parity bit for a word with n data bits could be computed as:

$$p_i = \bigoplus_{t=0}^{T-1} \bigoplus_{k=0}^{\lfloor \frac{n-t-i}{P+T-1} \rfloor} d_{i+k\cdot(P+T-1)+t} \quad , \qquad n \geq P \tag{5.2}$$

Three examples of stepped parity with four parity bits ($P = 4$) and parity thickness of 1, 2, and 4 ($T = 1, 2, 4$) are depicted in Figure 5.7. As it can be seen, by increasing the parity thickness, the number of required XOR operation grows which means larger encoding/decoding circuitry is needed. On the other hand, such an increase improves the error detection capability. For example, there are $P + T - 2$ unaffected cells between two erroneous cells in the smallest double error MBU pattern which cannot be detected using stepped parity $S(P,T)$. Considering the fact that the probability of having double error significantly reduces by having more unaffected cells in between, a significant improvement is obtained by increasing the parity thickness.

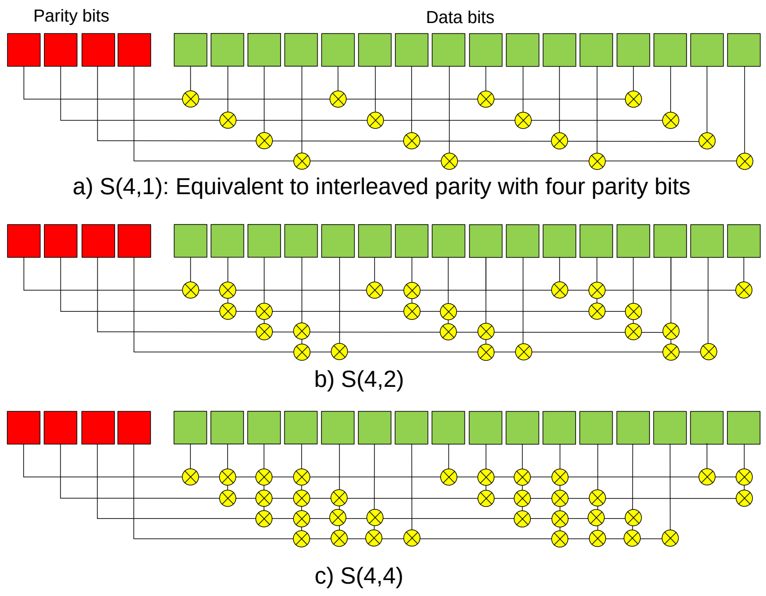

Parity bits
Data bits
a) S(4,1): Equivalent to interleaved parity with four parity bits
b) S(4,2)
c) S(4,4)

Figure 5.7: Examples of stepped parity with four parity bits and parity thickness a) one b) two, and c) four

It is worth to mention that $S(P,1)$ is equivalent to the virtually interleaved parity with P parity bits. This could be clearly seen in Figure 5.7.a and also by assuming $T = 1$ in Equation 5.2:

$$p_i = \bigoplus_{k=0}^{\lfloor \frac{n-i}{P} \rfloor} d_{i+kP} \quad , \qquad n \geq P \tag{5.3}$$

5.2.2 Experimental Results

In order to show the effectiveness of the proposed technique, its MBU detection capability as well as the area and the power overheads are compared with the virtually interleaved parity. The area, latency, and power consumption for a memory array and its corresponding encoding/decoding circuitry are obtained with respect to Nangate 45 nm standard cell library and UMC memory maker in a similar technology node.

Error Correction Capability

Table 5.2 demonstrates the MBU detection coverage for different number of parity bits and thickness. The results are obtained by injecting one million MBU patterns with respect to their occurrence provabilities in a memory array with size of 1024×32 bits. According to this table, the virtually interleaved parity ($S(P, 1)$, see the first column of the results) requires 6 bits ($S(6, 1)$) to reach the detection coverage of 100% while the stepped parity could obtain the same coverage with only four parity bits with parity thickness of 3 (i.e. $S(4, 3)$).

Table 5.2: Error detection coverage of stepped parity

		Parity thickness (T)				
		1	2	3	4	5
# of parity bits (P)	1	69.234%				
	2	91.430%	97.238%			
	3	99.687%	99.694%	99.914%		
	4	99.981%	99.995%	**100.000%**	100.000%	
	5	99.996%	100.000%	100.000%	100.000%	100.000%
	6	**100.000%**	100.000%	100.000%	100.000%	100.000%

As it can be seen in Table 5.2, the error detection coverage typically grows by increasing the parity thickness. Basically such increase improves the error detection for small size MBUs at the expense of reduced detection coverages for larger MBU patterns. As mentioned earlier, larger patterns are less likely to occur and hence the detection coverage grows. However, there might be some cases in which slight reduction in detection coverage is observed.

Area Overhead

For obtaining the area overhead of the proposed stepped parity technique, the area of memory arrays with various number of parity bits is extracted from UMC memory maker [137]. The area for a memory array with size of 1024×32 is 26,811 μm^2. The area for memory with different number of parity bits is shown in Table 5.3. As seen, the area overhead for each parity bit is about 2.5% which is smaller than the overhead in the number of bits ($\frac{1}{32} = 3.1$). This is mainly due to the fact that the address decoder of the memory is not affected by such an increase.

The number of XOR gates in the encoding/decoding circuitry for stepped parity with different number of parity bits and parity thickness is presented in Table 5.4. As it can be observed, the number of required XOR bits sub-linearly grows by increasing the parity thickness. Although there is significant increase in the number of XOR gates and hence the size of the encoding/decoding circuitry, this circuitry is just a small part of the memory and its impact

Table 5.3: Memory area for different number of parities

# of parity bits (P)	Memory area Absolute[μm^2]	Memory area Overhead
1	27,486	2.5%
2	28,161	5.0%
3	28,835	7.5%
4	29,510	10.0%
5	30,185	12.6%
6	30,859	15.1%

Table 5.4: Number of XOR gates required by S(P,T) encoder/decoder circuitry

# of parity bits (P)	Parity thickness (T) 1	2	3	4	5
1	31				
2	30	41			
3	29	45	54		
4	28	47	**59**	70	
5	27	48	64	75	85
6	**26**	49	66	80	87

Table 5.5: Overall area overhead of S(P,T) compared to non-protected memory

# of parity bits (P)	Parity thickness (T) 1	2	3	4	5
1	2.79%				
2	5.30%	5.4%			
3	7.81%	7.9%	8.0%		
4	10.32%	10.5%	**10.6%**	10.7%	
5	12.82%	13.0%	13.2%	13.2%	13.3%
6	**15.3%**	15.5%	15.7%	15.8%	15.8%

on the overall area overhead is minimal. In order to investigate this fact, we estimated the absolute area of the encoding/decoding circuitry according to the area of an XOR gate by considering a reasonable whitespace.

The area overhead of the memory with error detection capability (including both memory array and encoding/decoding circuitry area) with respect to the non-protected memory for different cases are reported in Table 5.5. As it can be seen, by increasing the parity thickness, the area overhead does not significantly increases, however, it grows linearly by increasing the number of parity bits. Hence, from the area perspective, it is worth to reduce the number of parity bits at the expense of more complicated encoding/decoding circuitry.

In order to reach the detection coverage of 100%, the virtually interleaved parity requires six parity bits and imposes 15.33% area overhead ($S(6,1)$ in Table 5.5), however, the stepped parity with four parity bits and parity thickness of three could reach the same goal with only 10.59% area overhead. This indicates that the proposed technique reduces the amount of area overhead by 30.9% compared to the virtually interleaved parity.

The overall area overheads of the memory array and the encoding/decoding circuitry for both stepped parity and virtually interleaved parity applied on different memory configurations are illustrated in Figure 5.8. For both error detection techniques, the minimum number of parity bits required to provide the full detection coverage is considered, i.e. four bits for the stepped parity and six bits for the virtually interleaved parity. The results show that the area overhead reduction compared to the virtually interleaved parity for a memory with 512 words is 24%-28%. The ratio of reduction slightly increases for larger memory sizes and saturates for memory arrays with more than 4096 word on about 32%-33%.

Power Overhead

The power overhead of the stepped parity is computed with respect to the static and dynamic power consumption of the memory array and the encoding/decoding circuitry. The switching activity information (i.e. the read/write access rate and average number of flips in the address

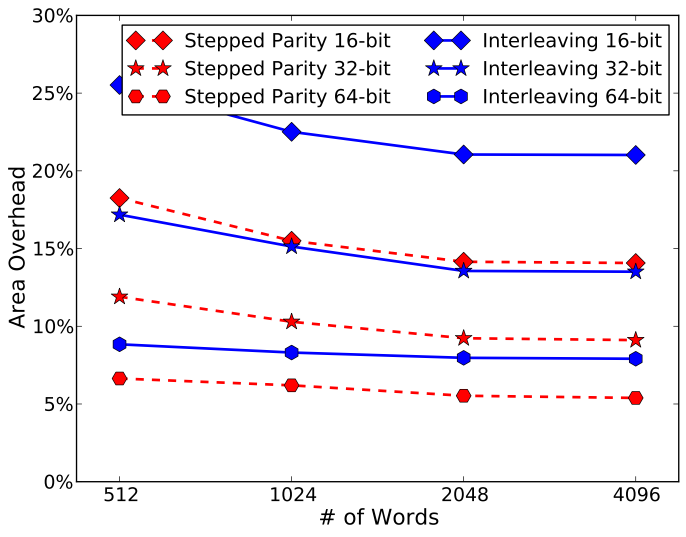


Figure 5.8: Comparison of area overhead of stepped parity and virtually interleaved parity for providing 100% error detection coverage (i.e. S(4,3) vs. S(6,1))

and data bits) is obtained by monitoring the data cache of the OR1200 processor where *stringsearch, bitcounts, basicmath*, and *qsort* workloads from MiBench benchmark suite [117] are executed. The size of the data cache is 1024×32 bits and its power consumption is 95.6 μW where no protection technique is applied.

Table 5.6: Overall power overhead of S(P,T) compared to non-protected memory

# of parity bits (P)	Parity thickness (T) 1	2	3	4	5
1	5.12%				
2	7.70%	8.13%			
3	11.57%	12.20%	12.55%		
4	15.45%	16.20%	**16.65%**	17.08%	
5	19.33%	20.14%	20.77%	21.20%	21.59%
6	**23.20%**	24.09%	24.76%	25.30%	25.58%

The power overhead of the memory array for different configuration of the stepped parity is reported in Table 5.6. As it can be seen, similar to the area overhead, the power overhead strongly depends on the number of parity bits, however, it also considerably grows by increasing the parity thickness. The proposed stepped parity technique reduces the power overhead by 28.2% compared to virtually interleaved parity (from 23.20% to 16.65%).

The power overhead imposed by the stepped parity ($S(4,3)$) and the virtually interleaved parity ($S(6,1)$) for different memory configurations are reported in Figure 5.9. We assumed that the switching activity information is similar to 1024×32 memory array. As shown, the proposed technique can considerably reduce the power overhead for all cases. The amount of reduction is higher for the memories with more number of words as the area overhead imposed by additional XOR gates would be smaller compared to the gain achieved by eliminating the additional two parity columns in virtually interleaved parity.

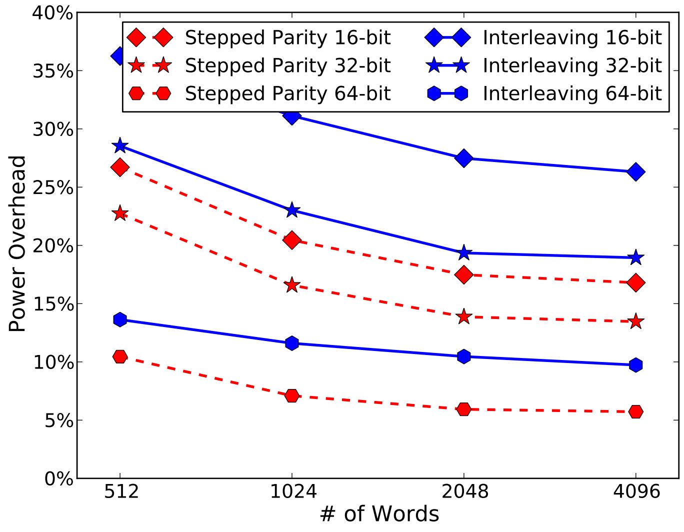


Figure 5.9: Comparison of power overhead of stepped parity and virtually interleaved parity for providing 100% error detection coverage (i.e. S(4,3) vs. S(6,1))

Table 5.7: Memory access latency for different number of parities

# of parity bits (P)	Latency [ns]
1	1.620
2	1.623
3	1.627
4	1.630
5	1.633
6	1.637

Table 5.8: Depth of XOR tree in S(P,T) encoder/decoder circuitry

# of parity bits (P) \ Parity thickness (T)	1	2	3	4	5
1	5				
2	4	5			
3	4	4	5		
4	3	4	**5**	5	
5	3	4	4	4	5
6	**3**	4	4	4	5

Table 5.9: Overall latency overhead of S(P,T) compared to non-protected memory

# of parity bits (P) \ Parity thickness (T)	1	2	3	4	5
1	12.24%				
2	10.02%	12.43%			
3	10.26%	10.26%	12.67%		
4	8.04%	10.45%	**12.86%**	12.86%	
5	8.23%	10.63%	10.63%	10.63%	13.05%
6	**8.47%**	10.88%	10.88%	10.88%	13.29%

Latency Overhead

Increasing the number of parity bits has a negligible impact on the access latency of the memory array (see Table 5.7), however, the delay of the encoding and decoding circuity considerably affects the overall latency. In order to estimate the latency of the encoding/decoding circuity for each configuration of the stepped parity, the corresponding circuity is synthesized with Synopsys Design Compiler with respect to 45 nm Nangate standard cell library. In all cases, the objective was to minimize the latency, hence, all the synthesized netlists were XOR trees with minimum depth (i.e. balanced).

The overall delay overheads of the different configurations of the stepped parity are reported in Table 5.9 for a memory array of 1024×32 bits. The delay of the encoding/decoding circuitry, and hence the overall delay overhead are linearly dependent on the depth of the XOR tree.

As reported in this table, unlike area and power overheads, the latency overhead reduces by increasing the number of parity bits as fewer data bits are used for computing each parity bit. The results show that the virtually interleaved parity and stepped parity impose 8.47% and 12.86% latency overhead compared to the non-protected memory array, respectively. The additional latency overhead of the stepped parity is mainly due to the larger depth of the XOR tree in the encoding circuitry.

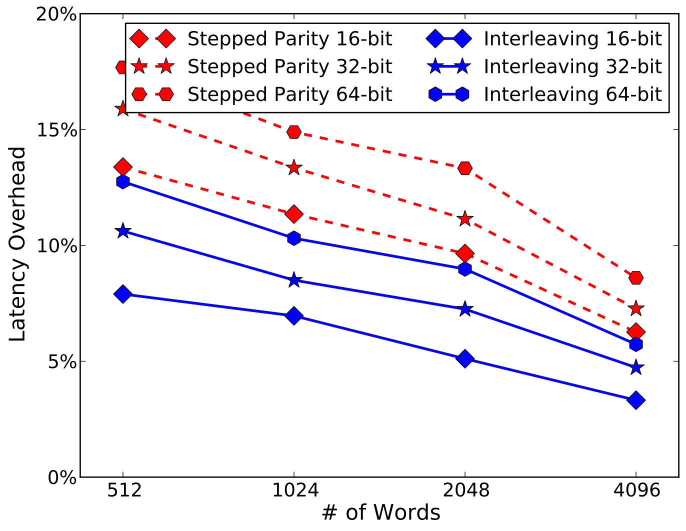


Figure 5.10: Comparison of latency overhead of stepped parity and virtually interleaved parity for providing 100% error detection coverage

In case the additional latency of the memory array increases the number of cycles required for each read/write access, it leads to a performance penalty at system level. Otherwise, such an increase has no impact at the higher levels. For example, the maximum frequency of the OR1200 processor after synthesizing it with 45 nm Nangate and performing place & route optimization is 894 MHz. This indicates that the minimum clock period for this processor is 1.118 ns. The delay of the data cache of this processor (1024×32 bits) is 1.617 ns, hence, each access to this cache unit requires two cycles. For this particular memory array, the access latency after applying the stepped parity or the virtually interleaved parity is still less than two cycles. Therefore, for this case, an appropriate error detection technique could be selected with respect to the area and power overheads of the memory array. In contrast, if the additional latency leads to an excessive memory access cycle in first level caches, techniques with higher area and power overhead but smaller latency are more beneficial to avoid the performance penalty at system-level.

Figure 5.10 shows the latency overhead of the proposed technique for different memory sizes. As shown, the latency overheads of both techniques are reduced by increasing the number of words. Also, the latency overhead of the stepped parity for all cases is higher than the virtually stepped parity, however, the absolute difference between these two techniques is at most 78 ps. Hence, it is very likely that the number of clock cycles per access would be the same for these techniques.

5.3 Low-cost MBU Correction in Configuration Frames

In order to meet the ever increasing performance and power demands, FPGAs are typically fabricated using the most advanced technology nodes. Recently, FPGAs based on 14 nm technology with more dense integration schemes such as 3D die stacking have been announced [138, 139]. In such small device geometries, a single radiation-induced particle strike is likely to affect several adjacent cells in a memory array leading to a MBU [116, 13]. Considering the fact that the MBU rate in nanoscales is comparable with that of the SEU [3, 32], an appropriate scheme is required to detect and correct multiple errors in memory arrays. More specifically, SRAM-based FPGAs are more prone to soft errors as a particle strike in a configuration frame[2] has a permanent impact on the functionality of the mapped design [140]. Since the configuration frames constitute the majority of SRAMs in an FPGA device (e.g. more than 80% for Xilinx Virtex-6 VLX240T), mitigation of MBUs in configuration frames is of decisive importance.

Several schemes have been presented to address the increasing soft error concern in FPGA configuration frames. The main objective of these schemes is to reduce error latency, and hence, to avoid error accumulation within configuration frames. Costly modular redundancy [141, 142, 143] is a conventional technique to tolerate soft errors in both configuration frames and functional logic. However, accumulated errors in both data and configuration bits dramatically limit the mean time to failure of such schemes [127]. Another technique is to optimize the configuration frame circuitry for soft errors as detailed in [144, 145]. However, such hardening techniques are not implemented in existing FPGA devices because of their excessive area overheads. Therefore, a low-cost solution is required to correct erroneous configuration frames during operation. The combination of configuration scrubbing and ECCs is an effective solution to detect and correct radiation-induced transient errors in configuration bits. In this regard, SEU tolerant scrubbing scheme [146, 147, 148, 149] is very well studied in the literature, and it is already included in Xilinx and Altera design flows [150, 151].

There are also few schemes to specifically address MBUs in FPGA devices. The scheme proposed in [152] employs the two dimensional hamming code in each configuration frame to correct MBUs. In [153], another scheme based on interleaved single error correction hamming code has been presented which could correct up to four adjacent error bits. In addition, Xilinx offers a soft error mitigation controller (IP block) based on *Cyclic Redundancy Code* (CRC) and ECC which can correct up to two adjacent erroneous cells in each configuration frame [154]. Recent experiments reveal that the number of affected bits by an MBU incident grows as technology scales [3]. Therefore, a more complicated ECC with very high area overhead is required to efficiently address the increase in MBU incidence rate.

Based on the fact that error detection can be done at much lower cost than error correction [77], we propose an MBU detection technique to detect the erroneous configuration frame during configuration scrubbing. Once an error is detected, by taking advantage of erasure codes and using the data stored in a redundant frame, the correct contents of the affected configuration frame is reconstructed. For detecting MBUs in the configuration frames of the FPGA, we propose a low-cost technique namely *Interleaved n Dimensional (InD) parity.* The interleaving distance of this technique is optimized based on the actual MBU patterns and their respective probabilities obtained from a detailed technology dependent analysis. Furthermore,

[2]A configuration frame is the smallest addressable chunk of a configuration memory

by carefully dividing the frames into several clusters, the proposed technique can detect and correct an MBU affecting several adjacent configuration frames.

For the case study on a Xilinx FPGA device, the proposed InD parity technique is able to detect 100.00% of the soft errors, and all detected errors are successfully recovered by means of erasure codes. Our proposed scheme only occupies 3.3% of logic blocks and 0.9% of *Block RAMs* (BRAMs[3]) out of the available resources in the employed FPGA device. The results also reveal that our proposed scheme provides the highest recovery coverage[4] with considerably less area overhead compared to the existing solutions at the expense of a negligible increase in the Mean Time To Repair (MTTR)[5] (3.75%).

5.3.1 Erasure Codes

Our proposed error correction scheme is based on the concept of erasure codes [155, 156]. An optimal erasure code is a data recovery technique which transforms m blocks into $m+n$ blocks such that the original m blocks can be reconstructed from any arbitrary set of m blocks among $m+n$ coded blocks (demonstrated in Figure 5.11). This data recovery technique is widely implemented in reliable storage devices (hard disks and CDs) [157], error correction in cache arrays [86], multimedia multi-casting, and signal transfer protocols [158].

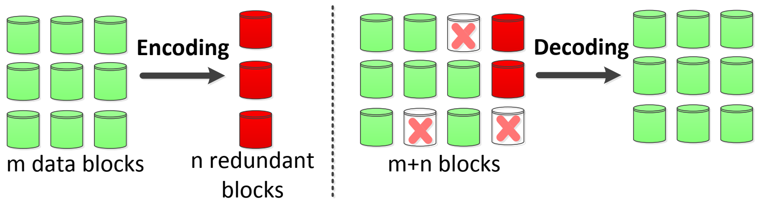


Figure 5.11: Encoding and decoding of erasure codes

A variety of erasure codes with different recovery coverage, area overhead, and encoding/decoding complexity are proposed in the literature [155, 156]. The area overhead of an erasure code is expressed as the ratio of redundant blocks to the data blocks (i.e. n/m). The recovery coverage of an erasure code is defined as the maximum number of erasures that could be tolerated. For the optimal erasure codes, the recovery coverage is equal to the number of redundant blocks (i.e. n).

Erasure codes are not proposed to detect or correct errors rather to retrieve the original blocks when a subset of blocks is not available (i.e. erased). However, once an error is detected in some blocks, by assuming that those blocks are not available, the original blocks could be recovered by means of an erasure code.

5.3.2 MBU Detection

Considering the fact that the entire configuration frame could be recovered using an erasure code, the identification of the exact location of erroneous bits is not of our interest, rather

[3]BRAMs are pre-wired ASIC-like RAMs available to logic fabric through read/write ports

[4]Recovery coverage is the percentage of the errors that can be successfully detected and recovered to the original state

[5]Mean Time to Repair is the average time required to detect and correct an error

a low-cost error detection technique with a very high detection coverage is required. In this regard, we present a very efficient error detection coding technique called InD parity for MBU detection in the configuration frames. The error detection capability of this technique in the configuration frame for two particular cases where N=2 and N=3 are investigated in details.

InD Parity

In memory arrays of typical microprocessors such as cache units, the parity bits employed for error detection in each memory entry (i.e. word) are computed during each memory access. Hence, from the performance perspective, it is crucial that each memory entry has its own error detection coding. Otherwise, during each memory access, all other entries that are involved in the computation of the corresponding common parity bit(s) must be accessed as well. However, the errors in FPGAs are detected during the periodic scrubbing where the error checking unit can access the entire contents of a configuration frame. Therefore, a parity bit could be computed based on multiple entries.

In the proposed error detection technique, we exploit the fact that the sizes of large MBU patterns are typically much smaller than the size of a configuration frame. Since an MBU incident affects several bits in a localized manner, the bits which are located far enough cannot be simultaneously affected with one MBU incident. Therefore, having separate parities for such bits in configuration frames neither increases the error detection capability nor improves the performance, rather only imposes unnecessary area overhead.

In order to increase the cost-efficiency of error detection for the configuration frames, we introduce the idea of InD parity. The main idea is to use the same parity bit for the bits that are separated by a constant distance to minimize the area overhead (i.e. interleaved parity). In addition, the parity bits are distributed at several dimensions to increase the detection coverage with respect to probable MBU patterns as there are some MBU patterns that cannot be detected using one or two dimensions, no matter how many parity bits are employed. The number of parity bits in each dimension theoretically has to be at most equal to the largest MBU spread on that dimension. However, in practice, large MBU patterns are typically detected using the parity bits on the other dimensions. Consequently, the number of parity bits required by this technique is always smaller than this theoretical limit.

It should be mentioned that the proposed InD parity technique is conceptually different from the existing error detection techniques used for an interleaved memory such as cache units [130, 18]. In the latter case, each word has its own error detection code and also, the memory words are physically interleaved to reduce the probability of having more than one erroneous bit in each word for an MBU incidence. However, this does not reduce the area overhead of the error detection technique. In contrast, the proposed InD parity coding is a *virtual* interleaving technique which does not require any changes to the configuration frame structure. In addition, in the case of a large configuration frame, the number of redundant bits could be drastically reduced in comparison to the complete interleaved parity coding.

In order to clearly show the detection capability of the proposed technique, the organization of parity bits as well as the error detection capability of the proposed interleaved two and three dimensional (I2D and I3D) parity schemes are detailed first and then the generalization of this technique in higher dimensions is discussed.

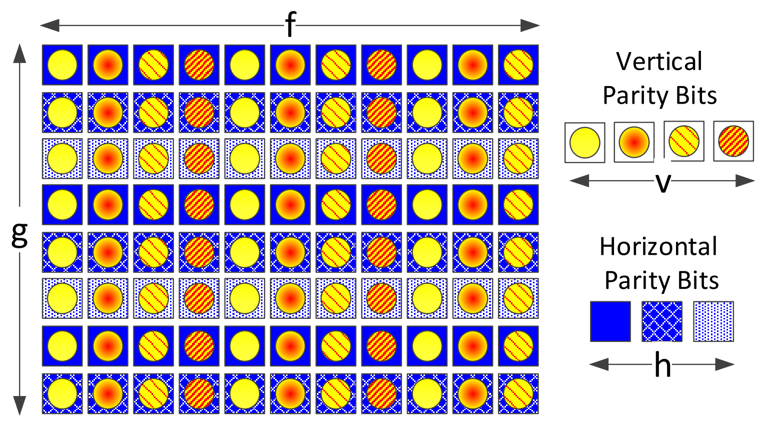


Figure 5.12: An example of I2D with vertical and horizontal distance of 4 and 3, respectively

I2D Parity

In the complete (traditional) 2D parity, a parity bit is associated for each row (column) which is constructed by XORing all the bits in that particular row (column). In the I2D parity technique, each horizontal (vertical) parity bit is the XOR of the bits in multiple rows (columns). More specifically, all rows (columns) that are separated by a constant distance of h (v) form an *interleaving group* and all the bits within that interleaving group are covered by only one horizontal (vertical) parity bit. For a $f \times g$ memory array, the complete 2D parity have $f + g$ parity bits while I2D parity requires only $h + v$ parity bits. Additionally, the number of XOR operations in both schemes are of the same order. These are formulated in Table 5.10.

Table 5.10: Parity computation for complete 2D and I2D

	2D parity	I2D parity
Horizontal Parity	$p_i^{row} = \bigoplus\limits_{j=1..f} a_{i,j}$	$p_i^{row}[1..h] = \bigoplus\limits_{\substack{j=1..f \\ x=0..\lfloor g/h \rfloor}} a_{i+xh,j}$
Vertical Parity	$p_j^{col} = \bigoplus\limits_{i=1..g} a_{i,j}$	$p_j^{col}[1..v] = \bigoplus\limits_{\substack{i=1..g \\ y=0..\lfloor f/v \rfloor}} a_{i,j+yv}$
No. of Parity Bits	$f + g$	$v + h$

Figure 5.12 depicts an example of I2D coding with the horizontal and the vertical interleaving distance of 3 and 4, respectively. In this example, first, second and third horizontal parity bits are the parity of all bits belonging to {1,4,7}, {2,5,8}, and {3,6} rows, respectively.

Both the horizontal and the vertical interleaving distance directly affect the detection coverage of the I2D parity technique. The only cases that I2D cannot detect an MBU incident is when the MBU affects an even number of bits involved in the computation of a certain parity bit while it is not detected by other parity bits. However, most of such cases are detectable by the complete 2D parity technique. Figure 5.13 illustrates three examples of such MBU patterns which cannot be detected by I2D with horizontal and vertical distance of 3 and 4, respectively.

Since the size of MBU patterns (in worst case, 24 bits are affected in our experiments) is considerably smaller than the size of a configuration frame (e.g. each configuration frame of a Virtex-6 device comprises of 81×32 bits), in case the horizontal and the vertical interleaving distance would be large enough, I2D can provide exactly the same MBU detection coverage

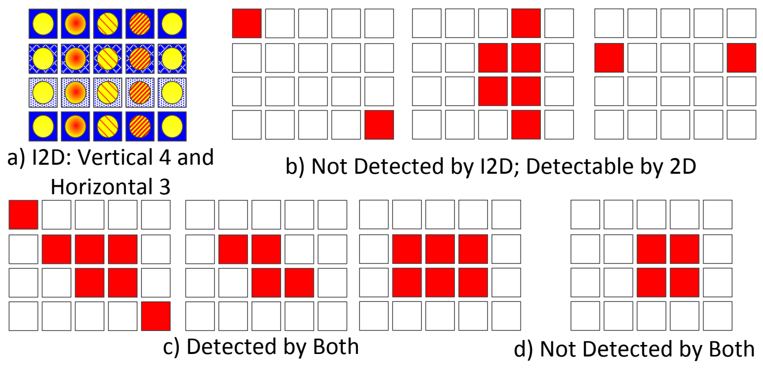

a) I2D: Vertical 4 and Horizontal 3
b) Not Detected by I2D; Detectable by 2D
c) Detected by Both
d) Not Detected by Both

Figure 5.13: Examples of MBU patterns for comparison of detection capability of 2D and I2D (vertical and horizontal distance of 4 and 3, respectively)

as a complete 2D parity technique. This is because the erroneous bits would be covered by at least one interleaving group. Therefore, the horizontal and the vertical interleaving distance have to be determined according to the possible MBU patterns.

The impact of the horizontal and the vertical interleaving distance on the error detection coverage are described in Table 5.11. The results presented in this table are extracted by analyzing each 45 nm MBU pattern (Figure 3.4 and Figure 3.5) and investigating the detection capability of I2D with a given interleaving distance for that pattern. As shown in Table 5.11, the error detection coverage grows by increasing the interleaving distances and saturates for the vertical and horizontal distance of 5. This detection coverage is exactly the same number that the complete 2D parity can detect for these MBU patterns, however, our I2D technique needs only 10 parity bits for error detection whereas the complete 2D parity requires 113 parity bits for each Xilinx Virtex-6 configuration frame.

Table 5.11: Error detection coverage vs. interleaving distance for I2D parity

	v=0	v=1	v=2	v=3	v=4	v=5
h=0	0.000%	68.960%	85.513%	89.990%	90.223%	90.278%
h=1	68.960%	68.960%	85.513%	89.990%	90.223%	90.278%
h=2	81.797%	81.797%	91.827%	95.590%	95.827%	95.862%
h=3	85.216%	85.216%	94.937%	98.688%	98.937%	98.960%
h=4	85.625%	85.625%	95.004%	98.752%	99.208%	99.286%
h=5	85.659%	85.659%	95.287%	99.038%	99.287%	**99.301%**

I3D Parity

In general, there might be some MBU patterns which affect an even number of cells in each row and column. This kind of patterns cannot be detected by the I2D parity technique since parity bit can only detect an odd number of erroneous bits. Examples of MBU patterns, in this technology, with significant occurrence probabilities that cannot be detected either by I2D or traditional 2D parity technique are shown in Figure 5.14. In order to detect such MBUs, a more powerful MBU detection technique is required. In this regard, we propose I3D parity technique which has an additional set of parity bits for diagonals.

Computation of the horizontal and the vertical parity bits in I3D parity follows the same rules explained for I2D parity. Similar to the interleaving technique employed in I2D parity to reduce

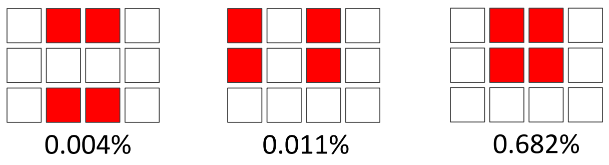


Figure 5.14: Three major MBU patterns that cannot be detected by either I2D or traditional 2D parity and their occurrence probability

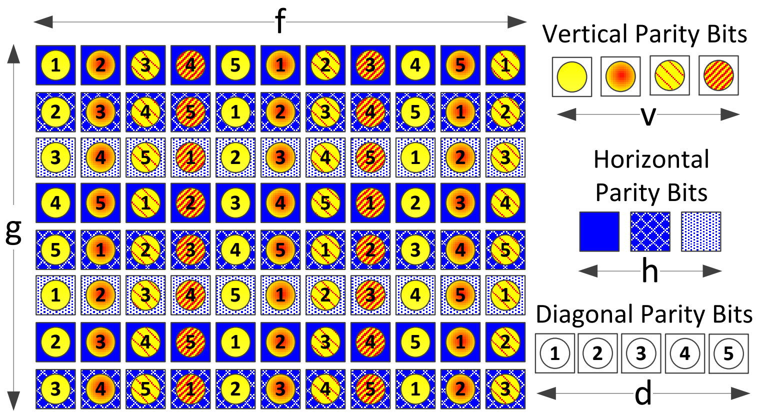


Figure 5.15: An example of I3D with vertical, horizontal, and diagonal distance of 4, 3, and 5, respectively

the number of horizontal and vertical parity bits, in I3D parity, several interleaving groups are formed for the diagonals as well. Then for each group only one parity bit is computed. In order to uniformly distribute d diagonal parity bits for a configuration frame of size $g \times f$, the interleaving group for the bit position of (i, j) could be computed by $(i + f \cdot (j - 1))$ mod d. An example of the I3D parity technique with the vertical, the horizontal, and the diagonal interleaving distance of 4, 3, and 5, respectively, is depicted in Figure 5.15.

The error detection coverage of I3D parity for different horizontal, vertical, and diagonal interleaving distances are reported in Table 5.12. As it can be seen, the maximum error detection coverage of this coding technique is 100% and could be achieved with at least 10 parity bits (h=5, v=2, d=3). Although the number of parity bits is much less than the complete 2D parity (i.e. 113), this detection technique is able to detect all MBUs.

Comparison of I2D and I3D

The number of parity bits required by I2D to reach its maximum detection coverage is equal to that of I3D (both require 10 parity bits). However, the error detection coverage of I2D is less than that of I3D. The maximum error detection coverages for various number of parity bits employed in I2D and I3D parity techniques are demonstrated in Figure 5.16. This is obtained by comparing the detection coverage of all possible cases that result in the desired number of parity bits. For instance, for 4 parity bits in I2D parity, we need to compare the detection coverage for five cases which are (v=4,h=0), (v=3,h=1), (v=2,h=2), (v=1,h=3), and (v=0,h=4). As it can be seen in this figure, the I3D parity always provides higher detection coverage than I2D parity when the number of parity bits is more than 2. Although the I3D

Table 5.12: Interleaving distance vs. error detection coverage for I3D parity

		v=0	v=1	v=2	v=3	v=4	v=5
d=0	h=0	0.000%	68.960%	85.513%	89.990%	90.223%	90.278%
	h=1	68.960%	68.960%	85.513%	89.990%	90.223%	90.278%
	h=2	81.797%	81.797%	91.827%	95.590%	95.827%	95.862%
	h=3	85.216%	85.216%	94.937%	98.688%	98.937%	98.960%
	h=4	85.625%	85.625%	95.004%	98.752%	99.208%	99.286%
	h=5	85.659%	85.659%	95.287%	99.038%	99.287%	99.301%
d=1	h=0	68.960%	68.960%	85.513%	89.990%	90.223%	90.278%
	h=1	68.960%	68.960%	85.513%	89.990%	90.223%	90.278%
	h=2	81.797%	81.797%	91.827%	95.590%	95.827%	95.862%
	h=3	85.216%	85.216%	94.937%	98.688%	98.937%	98.960%
	h=4	85.625%	85.625%	95.004%	98.752%	99.208%	99.286%
	h=5	85.659%	85.659%	95.287%	99.038%	99.287%	99.301%
d=2	h=0	84.988%	84.998%	89.735%	95.496%	95.477%	95.510%
	h=1	84.988%	84.998%	89.735%	95.496%	95.477%	95.510%
	h=2	89.715%	89.715%	89.736%	95.513%	95.478%	95.513%
	h=3	93.452%	93.452%	93.545%	99.301%	99.283%	99.301%
	h=4	93.547%	93.547%	93.545%	99.301%	99.287%	99.301%
	h=5	93.549%	93.549%	93.545%	99.301%	99.287%	99.301%
d=3	h=0	92.604%	92.604%	98.145%	98.037%	98.289%	98.309%
	h=1	92.604%	92.604%	98.145%	98.037%	98.289%	98.309%
	h=2	98.128%	98.128%	99.659%	99.387%	99.659%	99.870%
	h=3	98.801%	98.801%	99.650%	99.379%	99.652%	99.652%
	h=4	98.838%	98.728%	99.981%	99.743%	**100.000%**	**100.000%**
	h=5	99.015%	99.015%	**100.000%**	99.798%	**100.000%**	**100.000%**

parity technique provides more detection coverage with the same number of bits, it occupies more logical resources on the FPGA. This is because more parity bits are required to be stored for the third dimension (diagonal parity) and the complexity of the controller unit increases for the generation of such additional parity bits.

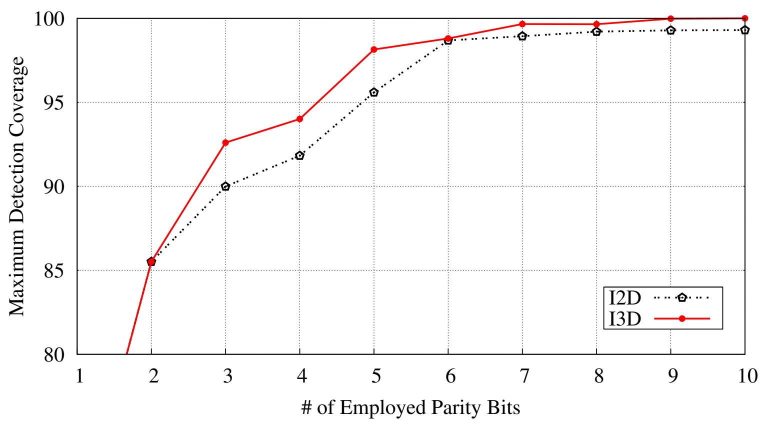


Figure 5.16: Maximum detection coverage obtained by I2D and I3D parity for different number of parity bits

5.3.3 Recovery Based on Erasure Codes

For error detection, we propose to implement a scrubber unit which periodically checks the parity bits of the configuration frames for possible errors. Upon a detection of an error, by assuming that the erroneous frame is erased, its contents are recovered using an erasure code.

The proposed scheme does not require any modification to the architecture of existing FP-

GAs. It can be mapped as a soft module alongside with the user design into the FPGA. This way, the scrubber can work in parallel with the user application.

Effective Erasure Code for Error Recovery in Configuration Frames

There are plenty of erasure codes with different characteristics in the literature. An effective erasure code should be selected in the context of the error recovery for the configuration frames. Since soft error occurrence rate is relatively small and scrubbing is continuously performed to detect and recover possible errors[6], it is unlikely to have multiple erroneous configuration frames in each scrubbing iteration. In addition, the encoding time of the erasure code is not a major issue in this context as it is done only once in advance during the design mapping to the FPGA device. In contrast, a high decoding time prolongs the error recovery process. Therefore, an erasure code with one redundant block and short decoding time satisfies our requirements.

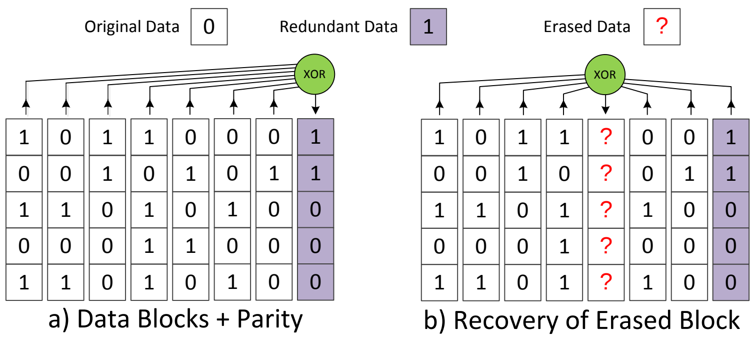


Figure 5.17: Encoding and decoding of parity-based erasure codes

The parity based erasure code [155] is an optimal erasure code with one redundant block ($n = 1$; a $m + 1$ erasure code), hence, it can recover from all cases with one erased block. In this technique, the redundant block stores the parity of data in all other blocks (Figure 5.17.a). In case of an erasure, the contents of the erased block could be simply computed by the parity of the other m blocks (Figure 5.17.b).

The factor m determines the trade-off between the area overhead and the encoding/decoding time. If all configuration frames in a large device are protected with only one erasure block, the error recovery time would be considerably high as all frames have to be read to compute the contents of the erased configuration frame. In order to reduce the error recovery time, FPGA frames could be divided into several clusters, each of which has its own redundant block (see Figure 5.18). In summary, a smaller m shortens the encoding/decoding time at the expense of higher area overhead.

It is worth to mention that, during the recovery process, the entire affected configuration frame is reconstructed which leads to MBU correction of that particular configuration frame. While erasure codes can completely reconstruct an erroneous configuration frame which has multiple errors, it fails when MBUs span across several configuration frames. In order to avoid such cases, frames for different clusters are appointed in a way that the frames from one cluster are not physically adjacent in the FPGA device. Therefore, in case an MBU affects adjacent

[6]The overall SER of all configuration frames of the Xilinx Virtex-6 VLX240T device is estimated with [39] to be 5.11 errors per 10^5 hours while the scrubbing iteration for the same device is about 18.7 ms

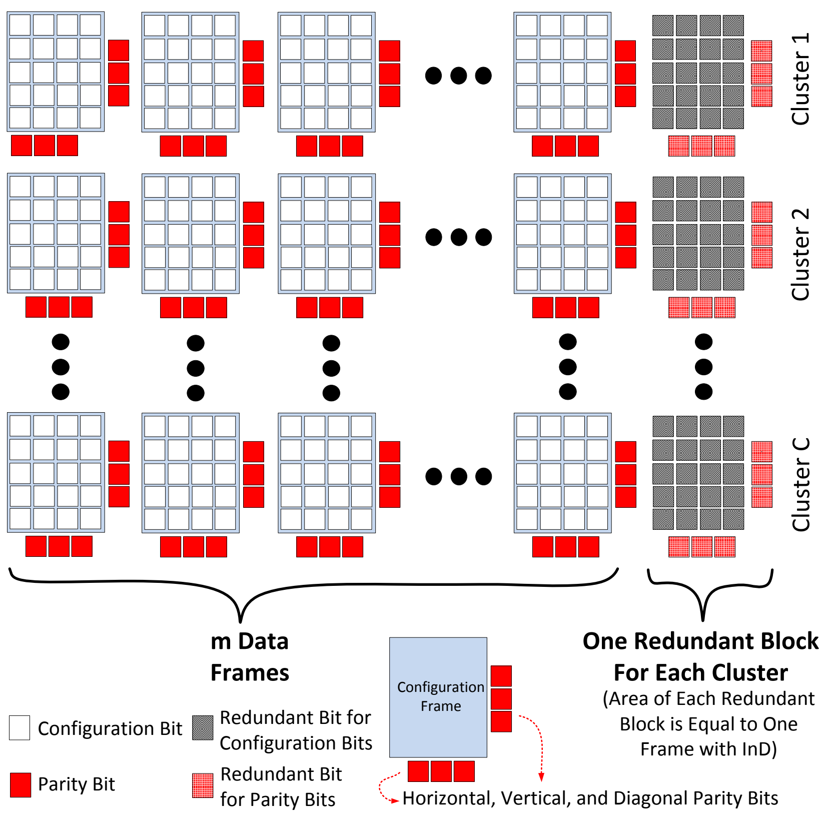

Cluster 1
Cluster 2
Cluster C
m Data Frames
One Redundant Block For Each Cluster
(Area of Each Redundant Block is Equal to One Frame with InD)
Configuration Frame
Configuration Bit
Redundant Bit for Configuration Bits
Parity Bit
Redundant Bit for Parity Bits
Horizontal, Vertical, and Diagonal Parity Bits

Figure 5.18: Configuration frame protection using proposed erasure code-based approach

frames, it affects only one frame from each cluster which could be recovered based on the redundant block in the same cluster.

The proposed technique is also applicable to FPGAs with partial reconfiguration capability. The partial reconfiguration is the ability to dynamically modify a subset of configuration frames by downloading their contents while the remaining logic continues to operate without interruption. In such designs, in each reconfiguration phase, in addition to the configuration frames, the corresponding erasure frames have to be updated as well.

Storing Redundant Data

The proposed scheme generates error detection codes (InD parity bits) for each configuration frame and also a redundant erasure block for each cluster. This additional data has to be stored in BRAMs of the FPGA device or in the spare bits of the configuration frames. In case the parity bits are stored in BRAMs, one important concern is not to store parity bits of different frames from one cluster very close together because an MBU might affect parity bits of several frames in that cluster. In such a scenario, affected bits cannot be recovered as the number of erroneous frames is more than the available erasure blocks (only one erasure block for each cluster). A similar issue also exists for storing error detection data and the erasure block of the same cluster, i.e. if an MBU affects both the erasure block of a cluster as well as the parity bits of one frame in the same cluster, the recovery is not possible.

Considering these important issues, it is essential to interleave the redundant data in a way that for each cluster at most either error detection parity bits of one frame or the redundant erasure block could be affected. Hence, the size of the clusters have to be larger than one, to facilitate an interleaving distance of at least one among data from each cluster.

Error Detection and Recovery Unit

The *Error Detection and Recovery* (EDR) unit periodically performs configuration scrubbing to detect possible erroneous frames. This unit reads a configuration frame word by word and performs the InD error detection for that frame. Upon a detection of an error, this unit restores the contents of the erroneous frame according to the contents of the remaining frames in the same cluster. It is worth to mention that the EDR unit not only scrubs the configuration frames but also scrubs the redundant erasure blocks stored in BRAMs for detecting possible errors. Otherwise, an accumulated error in a redundant block prevents from reconstructing the correct data when a configuration frame from the same cluster becomes erroneous in the future.

The scrubbing rate is determined by the desired level of reliability as well as the radiation-induced SER. For example, SER at altitude of 40,000 feet is about 500X more than that of the terrestrial environment [159], hence, a higher scrubbing rate is required for avionic systems to provide the same level of reliability.

Soft errors in the configuration frames or the functional logic of the EDR unit might affect its functionality. Therefore, it is essential to protect this unit against this kind of errors. In order to address this, we protect this unit using a TMR implementation. Since the size of the EDR unit is relatively small (it occupies less than 1% of the all resources in a Xilinx Virtex-6 FPGA device), the area overhead imposed by triplication (2%) is not significant. An important point to note is that since the EDR periodically scrubs the entire configuration frames of the device, it can also self-heal the soft errors occurring in the configuration frames pertaining to its own circuitry.

Once an error is detected in a configuration frame, the following steps required to be performed by the EDR unit for recovering the correct contents of that frame:

1. The EDR unit creates a temporary block and initializes all its bits with zero.
2. The EDR unit determines the erroneous configuration frame cluster and reads all the configuration frames of the erroneous cluster one by one by excluding the erroneous configuration frame. The recovery unit computes the XOR of all bits with their corresponding bits in the temporary block and replaces the new value in the temporary block. Once all frames in the cluster are processed, the temporary block contains the parity of all error-free frames in the same cluster.
3. The temporary block is written into the erroneous frame in the FPGA device.

Since the likelihood of having two erroneous frames from the same cluster in one scrubbing iteration is negligible[7], such cases could be safely ignored.

The objective of this scheme is to address the permanent effect of radiation-induced soft

[7]The SER of an SRAM cell is 6.93×10^{-13} per hour in 45 nm technology [39]. For a configuration size of 81×32, cluster size of 50, and scrubbing rate of 18.7 ms, the probability of having one error in one cluster is 2.33×10^{-11}, while the probability of having more than one error in the same cluster in one scrubbing period is 2.66×10^{-22}.

errors only in the configuration frames. However, errors might still affect the functionality of the mapped design from the error occurrence time until the recovery operation is performed. Therefore, a roll-back (e.g. checkpointing) [160, 161] or roll-forward (e.g. TMR) [161, 162] error correction technique is also required to mask errors propagated to the mapped design during this time interval.

5.3.4 Case Study: Implementation on a Xilinx FPGA Device

As a proof of concept, we implemented the proposed technique on a Xilinx Virtex-6 VLX240T device. In this section, we explain the implementation flow validated by a fault injection experiment. Then, we discuss the recovery time and area overhead trade-off in details and quantitatively compare the proposed scheme with existing solutions.

Implementation Flow

Xilinx FPGAs offer several interfaces to read and modify the configuration frames [163]. *Internal Configuration Access Port* (ICAP) is one of the fastest interfaces which operates at frequency of 100 MHz. In this work, this interface has been used as it is within the FPGA fabric and could be controlled by a custom hardware block.

We wrote a reconfigurable implementation of the EDR unit in Verilog which accepts the overall number of frames and the number of clusters as input. This implementation is completely independent of the user design and needs to be merged with the latter before mapping into the FPGA device.

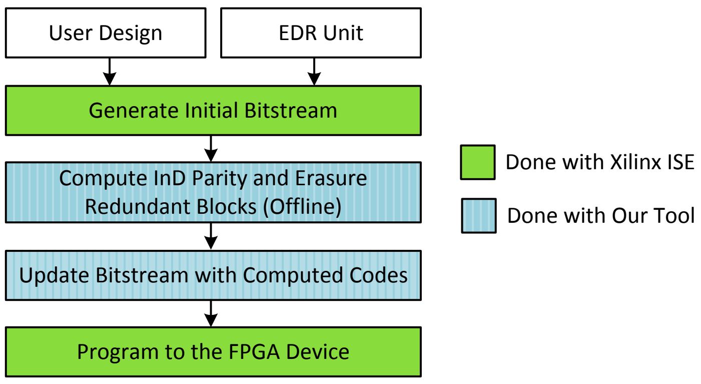

User Design
EDR Unit
Generate Initial Bitstream
Compute InD Parity and Erasure Redundant Blocks (Offline)
Update Bitstream with Computed Codes
Program to the FPGA Device
Done with Xilinx ISE
Done with Our Tool

Figure 5.19: Implementation flow of the proposed scheme to a Xilinx FPGA device

The implementation flow of our proposed MBU mitigation scheme is described in Figure 5.19. In the first step, the EDR unit as well as the user design are given to Xilinx ISE to build the initial bitstream. Once the initial bitstream is ready, the InD parity codes for the configuration frames are obtained by extracting the configuration frame data from the bitstream. At the same time, considering contents of all frames in each cluster, the redundant erasure block for that cluster is computed. Afterwards, the bitstream is updated by mapping the redundant erasure blocks into the idle BRAM units and parity bits into the spare bits of the configuration frame. Then, the updated bitstream is programmed into the FPGA.

Validation with Fault Injection

In order to validate the error detection and recovery capabilities of the proposed technique, we performed a fault injection experiment on the configuration bits of the Xilinx Virtex-6 VLX240T device. The fault injection is done by reading a randomly selected configuration frame using the ICAP interface. Afterwards, a random MBU pattern is applied to a random location of the frame and eventually the modified contents is written back to the configuration frame using the same interface. We have injected 1,000 errors on the configuration frames and initiated one iteration of scrubbing. At the end of the scrubbing period, we read the contents of the erroneous frame and compared its contents with the golden state before fault injection. The experimental result shows that the proposed technique is able to detect and correct all injected errors.

Area and Power Overhead

In the employed Xilinx Virtex-6 XLV240T device, there are 28,464 configuration frames each of which comprising of 81 32-bit words. In addition, each configuration frame has 16 spare bits for the Xilinx SEU correction mechanism [164]. As mentioned in Section 5.3.2, I3D parity requires 10 bits to reach its maximum detection coverage (i.e. 100%) and these bits could be stored in these 16 spare bits of the corresponding frame without any other storage mechanism such as BRAMs. Please note that the complete 3D parity scheme requires $2\times(81+32)=226$ bits for each configuration frame which has to be stored in a BRAM unit.

While the I3D parity bits can be stored in the spare bits of the same configuration frame, the redundant erasure blocks have to be stored in FPGA BRAMs. This is because a redundant erasure block occupies the same size of a configuration frame ($81\times32+16$), and the number of erasure blocks is linear to the number of employed clusters. Therefore, our scheme requires a separate storage medium such as BRAMs for storing erasure blocks. There are 416 BRAM units with a size of 36 Kb available in the employed Virtex-6 VLX240T device. In our proposed scheme, some of these BRAM units can be exploited for storing the erasure blocks. Figure 5.20 demonstrates the number of BRAM units required for storing erasure blocks for different number of clusters. According to the figure, it is evident that even for 50 clusters, which can significantly reduce the recovery time, the number of BRAMs required to store the redundant erasure blocks is only 4. This is less than 1% of the total BRAMs in the employed device. The entire EDR unit with built-in TMR and I3D detection technique occupies 1,219 (3.3%) CLBs in the employed Xilinx Virtex-6 VLX240T device.

We also obtained the power consumption overhead due to usage of the EDR unit alongside the user application. It is worth to mention that the EDR unit scrubs all configuration frames (whether used or not) for errors because an error in an unused configuration frame may cause a short circuit and damages the device. As a result, the power consumption of this unit is independent of the mapped design size. However, the power overhead of the EDR unit increases by the size of the mapped design as dynamic power consumption grows. Considering almost constant power consumption for the EDR unit, the maximum overhead could be obtained with respect to the case that no design is loaded to the FPGA device. The experiment carried out using the Xilinx XPWR tool shows that the power of the employed FPGA device without any design loaded is 2.87 watts, while mapping only the EDR unit to this FPGA increase the power consumption to 3.02 watts which shows power overhead of 5.2%. This small power overhead for the case where no design is loaded is due to high static power consumption of FPGA device. Furthermore, an experiment has been carried out for a subset of ITC'99 benchmark circuits mapped to this FPGA device, together with the EDR unit. The results show that the power consumption of the FPGA device increases on average only by 4.9% for applying proposed erasure code based protection scheme.

Detection and Recovery Time

The EDR unit periodically scrubs the entire configuration frames of the FPGA device through the ICAP interface. The FPGA device employed in this work has 28,464 configuration frames which have to be scrubbed by the EDR unit. Our experiments reveal that in case there is just one cluster employed for the entire device, the error recovery time from an erroneous frame is about 18.7 ms. However, as shown in Figure 5.20, by increasing the number of clusters, the recovery time drastically reduces at the expense of a linear increase in the BRAM area.

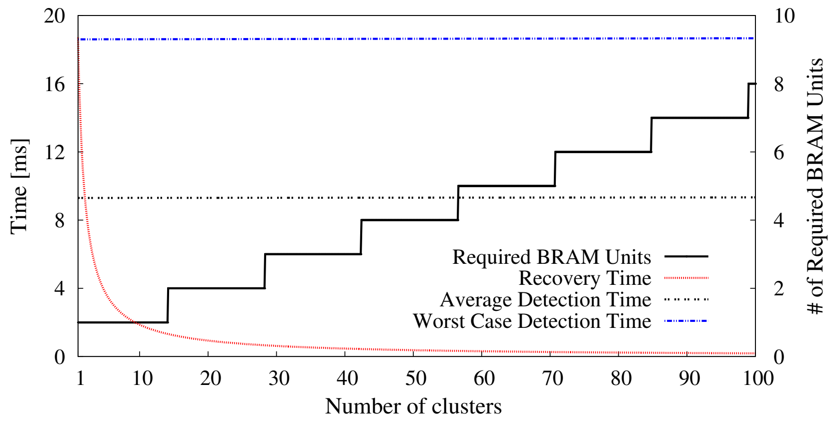


Figure 5.20: Error detection and correction time vs. number of required BRAMs for different cluster sizes

The error detection time in the proposed scheme is a function of the scrubbing time (the read-back time of entire configuration frames). The worst case scenario occurs when an MBU occurs in a frame right after it is scrubbed. In this case, the error remains in that frame for one complete scrubbing period (18.7 ms). However, the average MBU detection time is half of this period. Both the worst case and the average error detection time are strongly dependent on the number of clusters, however, since the number of clusters is negligible compared to the number of configuration frames, the slight increase in these factors is not distinguishable in Figure 5.20.

5.3.5 Comparison with the Previous Work

In order to clearly demonstrate the benefits and drawbacks of the proposed scheme, it is extensively compared with state-of-the-art solutions. The error recovery coverage of each scheme is obtained with respect to the MBU patterns obtained for a 45 nm technology considering the error correction capability of that particular scheme.

The detectability of each MBU pattern can be deterministically obtained for the 2D Hamming code [152] and Xilinx SEU correction [150]. However, Xilinx CRC+ECC [154] can prob-

Table 5.13: Comparison of different configuration frame soft error mitigation schemes

Scheme	Require External Storage	# of BRAMS	Average Error Detection Time [ms]	Average Error Recovery Time[ms]	Recovery Coverage
Xilinx SEU Correction [150]	No	0	9.342	0	51.718%
Xilinx CRC+ECC [154]	No	0	9.342	0	61.101%
Xilinx CRC+Reload [154]	Yes	0	9.342	18,7	100.000%
2D Hamming Code [152]	No	519	9.342	0	91.652%
I2D Parity + Erasure Code (50 Clusters)	No	**4**	**9.343**	**0.351**	**99.301%**
I3D Parity + Erasure Code (50 Clusters)	No	**4**	**9.343**	**0.351**	**100.000%**

abilistically detect specific patterns, hence, we performed a simulation-based MBU injection experiment to extract error detection probabilities for these schemes.

Xilinx has implemented three different schemes for soft error detection and correction in the configuration frames. The first scheme is based on a hamming code of distance 3 and is able to correct SEUs [150]. The second one utilizes a combination of CRC and ECC and is able to correct two adjacent cells in one word as well [154]. These two schemes are able to only correct 51.72% and 61.10% of the soft errors, respectively. The third scheme exploits CRC for error detection, however, it cannot identify the erroneous configuration frames. In this approach, after error detection, the contents of all configuration frames are reloaded from an external non-volatile storage. Although this technique can successfully detect and correct all MBUs, it has a long recovery time as it reloads the contents of all frames. In addition, the external storage imposes some additional costs to the system.

The MBU mitigation scheme presented in [152] exploits a 2D hamming code to correct MBUs. In the employed FPGA device which has a frame size of 81×32, horizontal and vertical hamming codes need 81×6 and 32×7 bits, respectively, which means a total of 710 hamming bits for each frame. In this scheme, for protecting all configuration frames more than 19.27 Mb memory is required which could be translated into 519 BRAM units. Although this scheme provides a very high error coverage, the employed FPGA device has only 416 BRAMs and there are not enough BRAM units to store all of the required hamming bits. Furthermore, this scheme cannot correct some of MBUs in the hamming data which significantly reduces its correction coverage compared to our proposed scheme.

Our proposed scheme with I2D parity detection and 50 clusters needs only four BRAM units and is able to correct 99.30% of the soft errors. By employing the proposed I3D parity technique, the error correction probability increases to 100.00%. As explained earlier, for both detection techniques, the parity bits could be stored in the redundant bits of the corresponding frame, and hence, do not impose any additional resource overhead. The proposed scheme only occupies 1% of the available BRAMs for storing erasure frames and is a good candidate to be used alongside with very large designs. Besides the area overhead for storing the error detection and correction data, all schemes have a common requirement for performing the scrubbing. Since it is a common requirement among all schemes, it is not reported in Table 5.13.

As reported in Table 5.13, the error detection time of all the schemes are almost the same. Indeed, all the schemes read the entire configuration frames using the ICAP interface and perform the error checking in parallel with reading frames, hence, the type of coding scheme does not affect the timing. The slight increase in the error detection time of our proposed scheme is due to the time required for scrubbing 50 additional erasure blocks employed for clusters. The error recovery time of the previous schemes is negligible as the error correction data is already loaded to the scrubber unit. Thus, the only added time would be due to the writing of corrected contents to the erroneous frame. In contrast, our proposed scheme requires some additional time to read all frames in the affected cluster and compute the erased frame contents. However, the sum of the error detection and the error recovery time determines the *Mean Time To Repair* (MTTR) which is the average time required by the system to return to its normal operation. The MTTR of previous schemes is 9.343 ms while our scheme has an MTTR of 9.694 ms. The 3.75% overhead in MTTR is reasonable considering the high error recovery coverage and the low area overhead of the proposed scheme.

Although the Xilinx CRC+Reload scheme can provide 100% recovery coverage, it requires an additional external non-volatile memory for storing contents of configuration memory. Our proposed technique eliminates the need for such an external memory. On the other hand, it significantly reduces MTTR compared to that scheme.

An important point is that although these schemes remove the permanent effect of soft errors from configuration frames, the errors could affect the mapped design functionality until the end of recovery operation. Since all these schemes have the same behavior from this perspective, this is not included in the comparison.

5.4 Conclusions

Three complementary MET/MBU mitigation techniques were presented in this chapter. The experimental results show that the proposed techniques could mitigate errors to a large extent. In more details:

- The proposed MT-aware placement algorithm could optimizes the locations of the cells in the layout with respect to the MT vulnerability. Experimental results show that the proposed layout-based modeling and mitigation approaches have modest runtime and are scalable for industrial-size circuits.
- The proposed MBU detection technique exploits two facts: 1) The cells affected by an MBU indecent are always very close to each other 2) likelihood of MBU patterns with smaller number of affected cells is typically higher. This technique considerably reduces the area and power overheads compared to the existing solutions.
- The proposed erasure code-based scheme is implemented as a generic soft core alongside with the user design and does not require any changes to the existing FPGA architecture. Compared to the previous solutions, our scheme provides the highest level of MBU protection at very low costs with a negligible recovery time. The implementation results reveal that the proposed scheme occupies only 1% of memory and 3% of logic resource on Xilinx Virtex-6 device.

6 Concluding Remarks

6.1 Summary and Conclusions

This thesis tackles the challenge of radiation-induced soft error in nanoscale technology nodes. Soft error is a physical phenomenon, however, it could be propagated and affect the output of running workloads at application-level. As a result, accurate soft error modeling requires a very fast and accurate cross-layer approach.

In this thesis, a comprehensive soft error analysis for an embedded processor using an efficient cross-layer modeling approach was presented. The proposed cross-layer modeling can accurately model the intrinsic SER as well as all the masking factors up to the application level. By having an appropriate model of abstraction at each step, errors were accurately propagated and their application-level effects were investigated. The results showed that selective protection is crucial for low-cost SER mitigation of memory arrays, flip-flops, and combinational gates. Furthermore, the impact of frequency and voltage scaling on the overall SER as well as the SER contributions of clock and reset tree were studied using this platform. These results can assist circuit designers to employ effective hardening techniques to reduce the SER with minimum impact on other design constraints.

By careful analysis of the results of SER of analysis, several SER mitigation potential were identified and addressed by proposing novel low-cost solutions:

- For SER mitigation in random logic, an effective multiple transient mitigation technique at the physical design step was presented which optimizes the locations of the cells in the layout with respect to the soft error vulnerability. Experimental results showed that the proposed layout-based mitigation technique has no area and performance overhead and is scalable for industrial-size circuits.
- In order to address the growing concern of radiation-induced MBUs in memory arrays, a cost-efficient error detection technique called stepped parity was presented. This technique exploits the fact that the cells affected by an MBU indecent are always very close to each other. The comparison of error detection capability as well as the area and power overheads of the stepped parity with the the state of the art demonstrates the cost-efficiency of the proposed technique.
- The configuration frames are the most vulnerable resources on the FPGA fabric to soft errors as they constitute the majority of the FPGA memory bits and once affected by soft errors, the functionality of the mapped design is permanently changed. In order to address this issue, a cost-efficient scheme based on erasure codes for MBU detection and correction in the configuration frames of SRAM-based FPGAs was presented. This scheme was implemented as a generic soft core alongside with the user design and does not require any changes to the existing FPGA architecture. Compared to the previous solutions, our scheme provides the highest level of MBU protection at very low costs

with a negligible recovery time.

6.2 Outlook

It is predicted that by further technology scaling to sub 10 nm region, the transistors become vulnerable to new types of particles such as proton- and muon-induced radiation strikes. The proposed cross-layer soft error analysis platform could be easily tuned to model new sources by incorporating the corresponding device-level models. Furthermore, the ratio and size of MBUs/METs will grow by further technology scaling, making our proposed MET/MBU mitigation techniques for logic, memory arrays, and FPGA more attractive.

Although this thesis concentrates on soft error modeling and mitigation, the proposed solutions are not restricted to this specific fault models and could be used for other kinds of transient faults. For instance, retention failures, read disturb, and write failures are common transient errors in spintronic-based memory array. These errors could be analyzed using our proposed fault injection platform and fault injection acceleration technique. In addition, the proposed stepped parity technique could be used to detect such errors. Another example is *Random Telegraph Noise* (RTN) of CMOS circuits in near-threshold computing which changes the gate delay for a very short period of time. The impact of RTN could be analyzed using our cross-layer approach by exploiting an appropriate fault model in device- and circuit-level analysis. In this context, we already exploited the platform and techniques presented in this thesis to evaluate some other reliability issues. Few examples of such evaluations are presented in [165, 166, 167, 168, 169].

Bibliography

[1] A. Dixit and A. Wood. The impact of new technology on soft error rates. In *International Reliability Physics Symposium*, pages 5B–4, 2011.

[2] R. Harada, Y. Mitsuyama, M. Hashimoto, and T. Onoye. Neutron induced single event multiple transients with voltage scaling and body biasing. In *International Reliability Physics Symposium*, pages 3C–4, 2011.

[3] E. Ibe, H. Taniguchi, Y. Yahagi, K.i. Shimbo, and T. Toba. Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule. *IEEE Transactions on Electron Devices*, 57(7):1527–1538, 2010.

[4] C. Rusu, A. Bougerol, L. Anghel, C. Weulerse, N. Buard, S. Benhammadi, N. Renaud, G. Hubert, F. Wrobel, T. Carriere, and R. Gaillard. Multiple event transient induced by nuclear reactions in cmos logic cells. In *IEEE International On-Line Testing Symposium*, pages 137–145, 2007.

[5] Altera Corporation. *Quartus II Handbook Version 11.0.* Altera, 2012.

[6] R. Aitken, E.H. Cannon, M. Pant, and M.B. Tahoori. Resiliency challenges in sub-10nm technologies. In *VLSI Test Symposium*, pages 1–4, 2015.

[7] J.-L. Autran and D. Munteanu. *Soft Errors: From Particles to Circuits*, volume 39. CRC Press, 2015.

[8] K. Reick, P.N. Sanda, S. Swaney, J.W. Kellington, M.J. Mack, M.S. Floyd, and D. Henderson. Fault-tolerant design of the ibm power6 microprocessor. *IEEE Micro*, 28(2):30–38, 2008.

[9] C.-Y. Cher, K.P. Muller, R.A. Haring, D.L. Satterfield, T.E. Musta, T.M. Gooding, K.D. Davis, M.B. Dombrowa, G.V. Kopcsay, R.M. Senger, et al. Soft error resiliency characterization and improvement on ibm bluegene/q processor using accelerated proton irradiation. In *International Test Conference*, pages 1–6, 2014.

[10] S.S. Mukherjee, C. Weaver, J. Emer, S.K. Reinhardt, and T. Austin. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In *International Symposium on Microarchitecture*, page 29, 2003.

[11] N. Seifert, B. Gill, K. Foley, and P. Relangi. Multi-cell upset probabilities of 45nm high-k+ metal gate sram devices in terrestrial and space environments. In *International Reliability Physics Symposium*, pages 181–186, 2008.

[12] R. Naseer and J. Draper. Parallel double error correcting code design to mitigate multi-bit upsets in srams. In *European Solid-State Circuits Conference*, pages 222–225, 2008.

[13] M. Ebrahimi, H. Asadi, and M.B. Tahoori. A layout-based approach for multiple event transient analysis. In *Design Automation Conference*, pages 1–6, 2013.

[14] M. Ebrahimi, H. Asadi, R. Bishnoi, and M. Tahoori. Layout-based modeling and mitigation of multiple event transients. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015.

[15] M. Ebrahimi, R. Seyyedi, L. Chen, and M.B. Tahoori. Event-driven transient error propagation: A scalable and accurate soft error rate estimation approach. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 743–748, 2015.

[16] M. Ebrahimi, A. Mohammadi, A. Ejlali, and S.G. Miremadi. A fast, flexible, and easy-to-develop fpga-based fault injection technique. *Microelectronics Reliability*, 54(5):1000–1008, 2014.

[17] M. Ebrahimi, N. Sayed, M. Rashvand, and M.B. Tahoori. Fault injection acceleration by architectural importance sampling. In *International Conference on Hardware-Software Codesign and System Synthesis*, 2015.

[18] M. Ebrahimi, A. Evans, M.B. Tahoori, R. Seyyedi, E. Costenaro, and D. Alexandrescu. Comprehensive analysis of alpha and neutron particle-induced soft errors in an embedded processor at nanoscales. In *Design, Automation and Test in Europe Conference*, pages 1–6, 2014.

[19] M. Ebrahimi, A. Evans, M.B. Tahoori, E. Costenaro, D. Alexandrescu, V. Chandra, and R. Seyyedi. Comprehensive analysis of sequential and combinational soft errors in an embedded processor. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015.

[20] B. Gill, N. Seifert, and V. Zia. Comparison of Alpha-particle and Neutron-induced combinational and sequential logic error rates at the 32nm technology node. In *International Reliability Physics Symposium*, pages 199–205, 2009.

[21] NN Mahatme, S Jagannathan, TD Loveless, LW Massengill, BL Bhuva, S-J Wen, and R Wong. Comparison of combinational and sequential error rates for a deep submicron process. *IEEE Transactions on Nuclear Science*, 58(6):2719–2725, 2011.

[22] N. Seifert, P. Shipleg, M.D. Pant, V. Ambrose, and B. Gill. Radiation-induced clock jitter and race. In *International Reliability Physics Symposium*, pages 215–222, 2005.

[23] M. Cabanas-Holmen, E.H. Cannon, A.J. Kleinosowski, J. Ballast, J. Killens, and J. Socha. Clock and reset transients in a 90 nm RHBD single-core Tilera processor. *IEEE Transactions on Nuclear Science*, 56(6):3505–3510, 2009.

[24] M.B. Tahoori M. Ebrahimi. Stepped parity: A low-cost multiple bit upset detection technique. In *International Test Conference*, 2015.

[25] M. Ebrahimi, P.M.B. Rao, R. Seyyedi, and M.B. Tahoori. Low-cost multiple bit upset correction in sram-based fpga configuration frames. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2015.

[26] J.-C. Laprie. Dependable computing and fault-tolerance. *Digest of Papers Fault Tolerant Computing Systems*, pages 2–11, 1985.

[27] M. Pizza, L. Strigini, A. Bondavalli, and F. Di Giandomenico. Optimal discrimination between transient and permanent faults. In *International High-Assurance Systems Engineering Symposium*, pages 214–223, 1998.

[28] T.C. May and M.H. Woods. A new physical mechanism for soft errors in dynamic memories. In *International Reliability Physics Symposium*, pages 33–40, 1978.

[29] S. Kumar, S. Agarwal, and J.P. Jung. Soft error issue and importance of low alpha solders for microelectronics packaging. *Reviews on Advanced Material Science*, 34:185–202, 2013.

[30] J.F. Ziegler and W.A. Lanford. Effect of cosmic rays on computer memories. *Science*, 206(4420):776–788, 1979.

[31] J.F. Ziegler. Terrestrial cosmic rays. *IBM journal of research and development*, 40(1):19–39, 1996.

[32] D. Radaelli, H. Puchner, S. Wong, and S. Daniel. Investigation of multi-bit upsets in a 150 nm technology SRAM device. *IEEE Transactions on Nuclear Science*, 52(6):2433–2437, 2005.

[33] D. Giot, P. Roche, G. Gasiot, J.-L. Autran, and R. Harboe-Sorensen. Heavy ion testing and 3-d simulations of multiple cell upset in 65 nm standard srams. *IEEE Transactions on Nuclear Science*, 55(4):2048–2054, 2008.

[34] J.A. Maestro and P. Reviriego. Study of the effects of mbus on the reliability of a 150 nm sram device. In *IEEE Design Automation Conference*, pages 930–935, 2008.

[35] J. Furuta, K. Kobayashi, and H. Onodera. Impact of cell distance and well-contact density on neutron-induced multiple cell upsets. In *International Reliability Physics Symposium*, pages 6C–3, 2013.

[36] R. Baumann. Soft errors in advanced computer systems. *IEEE Design & Test of Computers*, 22(3):258–266, 2005.

[37] I. Chatterjee, B. Narasimham, N.N. Mahatme, B.L. Bhuva, R.A. Reed, R.D. Schrimpf, J.K. Wang, N. Vedula, B. Bartz, and C. Monzel. Impact of technology scaling on sram soft error rates. *IEEE Transactions on Nuclear Science*, 61(6):3512–3518, 2014.

[38] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Dependable Systems and Networks*, pages 389–398, 2002.

[39] D. Alexandrescu. A comprehensive soft error analysis methodology for SoCs/ASICs memory instances. In *International On-Line Testing Symposium*, pages 175–176, 2011.

[40] D. Rossi, M. Omana, F. Toma, and C. Metra. Multiple transient faults in logic: an issue for next generation ics? In *Proceedings of International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 352–360, 2005.

[41] D.L. Hansen, E.J. Miller, A. Kleinosowski, K. Kohnen, A. Le, D. Wong, K. Amador, M. Baze, D. DeSalvo, M. Dooley, et al. Clock, flip-flop, and combinatorial logic contributions to the seu cross section in 90 nm asic technology. *IEEE Transactions on Nuclear Science*, 56(6):3542–3550, 2009.

[42] K. Cheng, S. Huang, and W. Dai. Fault emulation: A new methodology for fault grading. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems (TCAD)*, 18(10):1487–1495, 1999.

[43] L. Antoni, R. Leveugle, and B. Feher. Using run-time reconfiguration for fault injection applications. *IEEE Transactions on Instrumentation and Measurement*, 52(5):1468–1473, 2003.

[44] D. D. Andres, J. C. Ruiz, D. Gil, and P. Gil. Fault emulation for dependability evaluation of VLSI systems. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, 16(4):422–431, 2008.

[45] L. Sterpone and M. Violante. A new partial reconfiguration-based fault-injection system to evaluate SEU effects in SRAM-based FPGAs. *IEEE Transactions on Nuclear Science (TNS)*, 54(4):965–970, 2007.

[46] S. Hwang, J. Hong, and C. Wu. Sequential circuit fault simulation using logic emulation. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems (TCAD)*, 17(8):724–736, 1998.

[47] P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda, and M. Violante. Exploiting circuit emulation for fast hardness evaluation. *IEEE Transactions on Nuclear Science (TNS)*, 48(6):2210–2216, 2001.

[48] A. Ejlali, S. G. Miremadi, H. Zarandi, G. Asadi, and S. B. Sarmadi. A hybrid fault injection approach based on simulation and emulation co-operation. In *Proceedings of Dependable Systems and Networks (DSN)*, pages 479–488,

2003.

[49] H. R. Zarandi, S. G. Miremadi, and A. Ejlali. Dependability analysis using a fault injection tool based on synthesizability of HDL models. In *Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 485–492, 2003.

[50] M. A. Aguirre, V. Baena, J. Tombs, and M. Violante. A New Approach to Estimate the Effect of Single Event Transients in Complex Circuits. *IEEE Transactions on Nuclear Science (TNS)*, 54(4):1018–1024, 2007.

[51] L. Entrena, M. Garcia-Valderas, R. Fernandez-Cardenal, M. Portela Garcia, and C. Lopez-Ongil. SET emulation considering electrical masking effects. *IEEE Transactions on Nuclear Science (TNS)*, 56(4):2021–2025, 2009.

[52] L. Entrena, M. Garcia-Valderas, R. Fernandez-Cardenal, A. Lindoso, M. Portela, and C. Lopez-Ongil. Soft error sensitivity evaluation of microprocessors by multilevel emulation-based fault injection. *IEEE Transactions on Computers*, 61(3):313–322, 2012.

[53] B. Zhang, W. Wang, and M. Orshansky. Faser: Fast analysis of soft error susceptibility for cell-based designs. In *International Symposium on Quality Electronic Design*, pages 755–760, 2006.

[54] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transaction on Computers (TC)*, 35(8):677–691, 1986.

[55] N. Miskov-Zivanov and D. Marculescu. Circuit reliability analysis using symbolic techniques. *IEEE Transaction on CAD of Integrated Circuits and Systems (TCAD)*, 25(12):2638–2649, 2006.

[56] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, and J. P. Hayes. Accurate reliability evaluation and enhancement via probabilistic transfer matrices. In *Proceedings of Design, Automation and Test in Europe Conference (DATE)*, pages 282–287, 2005.

[57] G. Norman, D. Parker, M. Z. Kwiatkowska, and S. K. Shukla. Evaluating the reliability of nand multiplexing with prism. *IEEE Transaction on CAD of Integrated Circuits and Systems (TCAD)*, 24(10):1629–1637, 2005.

[58] D. Bhaduri, S. K. Shukla, P. Graham, and M. Gokhale. Scalable techniques and tools for reliability analysis of large circuits. In *Proceedings of IEEE International Conference on VLSI Design (VLSID)*, pages 705–710, 2007.

[59] N. Miskov-Zivanov and D. Marculescu. Modeling and optimization for soft-error reliability of sequential circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(5):803–816, 2008.

[60] G. Asadi and M. B. Tahoori. An analytical approach for soft error rate estimation in digital circuits. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 3, pages 2991–2994, 2005.

[61] H. Asadi, M.B. Tahoori, M. Fazeli, and S.G. Miremadi. Efficient algorithms to accurately compute derating factors of digital circuits. *Microelectronics Reliability*, pages 1215–1226, 2012.

[62] L. Chen, M. Ebrahimi, and M.B. Tahoori. Cep: correlated error propagation for hierarchical soft error analysis. *Journal of Electronic Testing*, 29(2):143–158, 2013.

[63] F. Kriebel, S. Rehman, D. Sun, P.V. Aceituno, M. Shafique, and J. Henkel. Acsem: accuracy-configurable fast soft error masking analysis in combinatorial circuits. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 824–829, 2015.

[64] A. Biswas, P. Racunas, R. Cheveresan, J. Emer, S.S. Mukherjee, and R. Rangan. Computing architectural vulnerability factors for address-based structures. In *International Symposium on Computer Architecture*, pages 532–543, 2005.

[65] N.J. George, C.R. Elks, B.W. Johnson, and J. Lach. Transient fault models and avf estimation revisited. In *Dependable Systems and Networks (DSN)*, pages 477–486, 2010.

[66] N.J. Wang, A. Mahesri, and S.J. Patel. Examining ace analysis reliability estimates using fault-injection. In *ACM SIGARCH Computer Architecture News*, volume 35, pages 460–469, 2007.

[67] A. Haghdoost, H. Asadi, and A. Baniasadi. System-level vulnerability estimation for data caches. In *Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 157–164, 2010.

[68] Li Tang, Shuai Wang, Jie Hu, and Xiaobo Sharon Hu. Characterizing the l1 data cache's vulnerability to transient errors in chip-multiprocessors. In *Annual Symposium on VLSI (ISVLSI)*, pages 266–271, 2011.

[69] S. Wang. Characterizing system-level vulnerability for instruction caches against soft errors. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, pages 356–363, 2011.

[70] Y. Cheng, M.A. Anguo, and M. Zhang. Accurate and simplified prediction of l2 cache vulnerability for cost-efficient soft error protection. *IEICE Transactions on Information and Systems*, 95(1):56–66, 2012.

[71] P. Montesinos, W. Liu, and J. Torrellas. Using register lifetime predictions to protect register files against soft errors. In *Dependable Systems and Networks*, pages 286–296, 2007.

[72] D. Tang, C. He, Y. Li, H. Zang, C. Xiong, and J. Zhang. Soft error reliability in advanced cmos technologies-trends and challenges. *Science China Technological Sciences*, 57(9):1846–1857, 2014.

[73] A. Sehgal, A. Dubey, E.J. Marinissen, C. Wouters, H. Vranken, and K. Chakrabarty. Redundancy modelling and array yield analysis for repairable embedded memories. In *IEE Proceedings Computers and Digital Techniques*, volume 152, pages 97–106, 2005.

[74] N. Krishnamurthy, A.K. Martin, M.S. Abadir, J. Abraham, et al. Validation of powerpc tm custom memories using symbolic simulation. In *VLSI Test Symposium*, pages 9–14, 2000.

[75] J. Wuu, D. Weiss, C. Morganti, and M. Dreesen. The asynchronous 24 MB on-chip level 3 cache for a dual-core Itanium architecture processor. In *International Solid State Circuits Conference*, 2005.

[76] C.N. Keltcher, K.J. McGrath, A. Ahmed, and P. Conway. The AMD Opteron processor for multiprocessor servers. *IEEE Micro*, 23(2):66–76, 2003.

[77] R.H. Morelos-Zaragoza. *The art of error correcting coding.* John Wiley & Sons, 2006.

[78] P. Reviriego, S. Pontarelli, A. Evans, and J.A. Maestro. A class of SEC-DED-DAEC codes derived from orthogonal latin square codes. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–5, 2014.

[79] S. Cha and H. Yoon. Single-error-correction and double-adjacent-error-correction code for simultaneous testing of data bit and check bit arrays in memories. *IEEE Transactions on Device and Materials Reliability*, 14(1):529–535, 2014.

[80] K. Namba, S. Pontarelli, M. Ottavi, and F. Lombardi. A single-bit and double-adjacent error correcting parallel decoder for multiple-bit error correcting BCH codes. *IEEE Transactions on Device and Materials Reliability*, 2014.

[81] A Neale, M Jonkman, and M Sachdev. Adjacent-MBU tolerant SEC-DED-TAEC-yAED codes for embedded SRAMs. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2015.

[82] E.H. Neto, I. Ribeiro, M. Vieira, G. Wirth, and F.L. Kastensmidt. Using bulk built-in current sensors to detect soft errors. *IEEE Micro*, (5):10–18, 2006.

[83] P. Reviriego and J.A. Maestro. Efficient error detection codes for multiple-bit upset correction in SRAMs with BICS. *ACM Transactions on Design Automation of Electronic Systems*, 14(1):18, 2009.

[84] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe. Multi-bit error tolerant caches using two-dimensional error coding. In *IEEE/ACM International Symposium on Microarchitecture*, pages 197–209, 2007.

[85] X. Jian, J. Sartori, H. Duwe, and R. Kumar. High performance, energy efficient chipkill correct memory with multidimensional parity. *Computer Architecture Letters*, 12(2):39–42, 2013.

[86] A. Banaiyanmofrad, M. Ebrahimi, and N. Dutt F. Oboril, M.B. Tahoori. Protecting caches against multiple bit upsets using embedded erasure coding. In *European Test Symposium*, 2014.

[87] P. Rao, M. Ebrahimi, R. Seyyedi, and M.B. Tahoori. Protecting SRAM-based FPGAs against multiple bit upsets using erasure codes. In *Design Automation Conference (DAC)*, pages 1–6, 2014.

[88] S.S. Mukherjee, J. Emer, and S.K. Reinhardt. The soft error problem: An architectural perspective. In *International Symposium on High-Performance Computer Architecture*, pages 243–247, 2005.

[89] M. Maniatakos, M.K. Michael, and Y. Makris. AVF-driven parity optimization for MBU protection of in-core memory arrays. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1480–1485, 2013.

[90] M. Maniatakos, M.K. Michael, and Y. Makris. Multiple-bit upset protection in microprocessor memory arrays using vulnerability-based parity optimization and interleaving. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2014.

[91] S. Whitaker, J. Canaris, and K. Liu. Seu hardened memory cells for a ccsds reed-solomon encoder. *IEEE Transactions on Nuclear Science*, 38(6):1471–1477, 1991.

[92] Leonard R. Rockett J. An seu-hardened cmos data latch design. *IEEE Transactions on Nuclear Science*, 35:1682–1687, 1988.

[93] T. Calin, M. Nicolaidis, and R. Velazco. Upset hardened memory design for submicron cmos technology. *IEEE Transactions on Nuclear Science.*, pages 2874–8, 1996.

[94] M. Nicolaidis, R. Perez, and D. Alexandrescu. Low-cost highly-robust hardened cells using blocking feedback transistors. In *VLSI Test Symposium*, pages 371–376, 2008.

[95] R. Naseer and J. Draper. Df-dice: a scalable solution for soft error tolerant circuit design. In *International Symposium on Circuits and Systems*, 2006.

[96] T.D. Loveless, S. Jagannathan, T. Reece, J. Chetia, B.L. Bhuva, M.W. McCurdy, L.W. Massengill, S.-J. Wen, R. Wong, and D. Rennie. Neutron-and proton-induced single event upsets for d-and dice-flip/flop designs at a 40 nm technology node. *IEEE Transactions on Nuclear Science*, 58(3):1008–1014, 2011.

[97] V.B. Sheshadri, B.L. Bhuva, R.A. Reed, R.A. Weller, M.H. Mendenhall, R.D. Schrimpf, K.M. Warren, B.D. Sierawski, S.-J. Wen, R. Wong, et al. Effects of multi-node charge collection in flip-flop designs at advanced

technology nodes. In *International Reliability Physics Symposium*, pages 1026–1030, 2010.

[98] R. Rajaei, B. Asgari, M. Tabandeh, and M. Fazeli. Design of robust sram cells against single event multiple effects for nanometer technologies. *IEEE Transactions on Device and Materials Reliability*, pages 429–436, 2015.

[99] M. D'Alessio, M. Ottavi, and F. Lombardi. Design of a nanometric cmos memory cell for hardening to a single event with a multiple-node upset. *IEEE Transactions on Device and Materials Reliability*, 14(1):127–132, 2014.

[100] D.G. Mavis and P.H. Eaton. Seu and set modeling and mitigation in deep submicron technologies. In *International Reliability Physics Symposium*, pages 293–305, 2007.

[101] O.A. Amusan, L.W. Massengill, B.L. Bhuva, S. DasGupta, A.F. Witulski, and J.R. Ahlbin. Design techniques to reduce set pulse widths in deep-submicron combinational logic. *IEEE Transactions on Nuclear Science*, 6(54):2060–2064, 2007.

[102] N.M. Atkinson, J.R. Ahlbin, A.F. Witulski, N.J. Gaspard, W.T. Holman, B.L. Bhuva, E.X. Zhang, L. Chen, and L.W. Massengill. Effect of transistor density and charge sharing on single-event transients in 90-nm bulk cmos. *IEEE Transactions on Nuclear Science*, 6(58):2578–2584, 2011.

[103] Q. Zhou and K. Mohanram. Gate sizing to radiation harden combinational logic. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(1):155–166, 2006.

[104] M.R. Choudhury, Q. Zhou, and K. Mohanram. Soft error rate reduction using circuit optimization and transient filter insertion. *Journal of Electronic Testing*, 25(2-3), 2009.

[105] S. Almukhaizim and Y. Makris. Soft error mitigation through selective addition of functionally redundant wires. *IEEE Transactions on Reliability*, 57(1):23–31, 2008.

[106] K. Mohanram and N.A. Touba. Cost-effective approach for reducing soft error failure rate in logic circuits. In *International Test Conference*, 2003.

[107] iRoC Technologies. TFIT: Cell level soft error analysis, http://iroctech.com, 2015.

[108] M. Fazeli, S.N. Ahmadian, S.G. Miremadi, H. Asadi, and M.B. Tahoori. Soft error rate estimation of digital circuits in the presence of multiple event transients. In *Design, Automation and Test in Europe Conference*, pages 1–6, 2011.

[109] N. Miskov-Zivanov and D. Marculescu. Multiple transient faults in combinational and sequential circuits: a systematic approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(10):1614–1627, 2010.

[110] G Hubert, L Artola, and D Regis. Impact of scaling on the soft error sensitivity of bulk, fdsoi and finfet technologies due to atmospheric radiation. *Integration, the VLSI Journal*, 2015.

[111] M. Rardon M. Clarke, D. Hammerschlag and A. Sood. Eliminating routing congestion issues with logic synthesis. In *Cadence White Paper*, 2014.

[112] H Puchner, D Radaelli, and A Chatila. Alpha-particle seu performance of sram with triple well. *IEEE Transactions on Nuclear Science*, 51(6):3525–3528, 2004.

[113] Y-P Fang and Anthony S Oates. Neutron-induced charge collection simulation of bulk finfet srams compared with conventional planar srams. *IEEE Transactions on Device and Materials Reliability*, 11(4):551–554, 2011.

[114] M. Fazeli, S. G. Miremadi, H. Asadi, and S. N. Ahmadian. A fast and accurate multi-cycle soft error rate estimation approach to resilient embedded systems design. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 131–140, 2010.

[115] NN Mahatme, NJ Gaspard, T Assis, S Jagannathan, I Chatterjee, TD Loveless, BL Bhuva, LW Massengill, SJ Wen, and R Wong. Impact of technology scaling on the combinational logic soft error rate. In *International Reliability Physics Symposium*, pages 5F–2. IEEE, 2014.

[116] N. Seifert, B. Gill, S. Jahinuzzaman, J. Basile, V. Ambrose, Q. Shi, R. Allmon, and A. Bramnik. Soft error susceptibilities of 22 nm tri-gate devices. *IEEE Transactions on Nuclear Science*, 59(6):2666–2673, 2012.

[117] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *IEEE International Workshop on Workload Characterization*, pages 3–14, 2001.

[118] A. L Silburt, A. Evans, I. Perryman, S.-J. Wen, and D. Alexandrescu. Design for soft error resiliency in internet core routers. *IEEE Transactions on Nuclear Science*, 56(6):3551–3555, 2009.

[119] H. Cha et al. A gate-level simulation environment for alpha-particle-induced transient faults. *IEEE Transactions on Computers*, 45(11), 1996.

[120] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert. Statistical fault injection: quantified error and confidence. In *Design, Automation & Test in Europe Conference*, pages 502–506, 2009.

[121] V. Ferlet-Cavrois, L.W. Massengill, and P. Gouker. Single Event Transients in Digital CMOS: A Review. *IEEE*

Transactions on Nuclear Science, 60(3):1767, 2013.

[122] R. Rajaraman, J.S. Kim, N. Vijaykrishnan, Y. Xie, and M.J. Irwin. SEAT-LA: a soft error analysis tool for combinational logic. In *VLSI Design*, 2006.

[123] E. Costenaro, A. Evans, D. Alexandrescu, L. Chen, M. Tahoori, and M. Nicolaidis. Towards a hierarchical and scalable approach for modeling the effects of SETs. In *IEEE Workshop on Silicon Errors in Logic-System Effects (SELSE)*, 2013.

[124] S. Kiamehr, M. Ebrahimi, F. Firouzi, and M.B. Tahoori. Chip-level Modeling and Analysis of Electrical Masking of Soft Errors. In *VLSI Test Symposium*, 2013.

[125] C. Lopez-Ongil, L. Entrena, M. Garcia-Valderas, M. Portela, M. A. Aguirre, J. Tombs, V. Baena, and F. Munoz. A unified environment for fault injection at any design level based on emulation. *IEEE Transactions on Nuclear Science (TNS)*, 54(4):946–950, 2007.

[126] A. Mohammadi, M. Ebrahimi, A. Ejlali, and S. G. Miremadi. SCFIT: A FPGA-based Fault Injection Technique for SEU Fault Model. In *Proceedings of Design, Automation and Test in Europe Conference (DATE)*, pages 586–589, 2012.

[127] M. Ebrahimi, S.G. Miremadi, H. Asadi, and M. Fazeli. Low-cost scan-chain-based technique to recover multiple errors in tmr systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(8):1454–1468, 2013.

[128] A. Stuart and J.K. Ord. Kendalls advanced theory of statistics, volume i: Distribution theory. *Arnold, London*, 1994.

[129] Jedec89c, http://www.jedec.org/standards-documents.

[130] S. Baeg, S. Wen, and R. Wong. SRAM interleaving distance selection with a soft error failure model. *IEEE Transactions on Nuclear Science*, 56(4):2111–2118, 2009.

[131] V. Chandra and R. Aitken. Impact of voltage scaling on nanoscale sram reliability. In *Design, Automation and Test in Europe*, pages 387–392, 2009.

[132] B.T. Kiddie and W.H. Robinson. Alternative standard cell placement strategies for single-event multiple-transient mitigation. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 589–594, 2014.

[133] SOC Encounter. http://www.cadence.com, 2015.

[134] J.R. Ahlbin, L.W. Massengill, B.L. Bhuva, B. Narasimham, M.J. Gadlage, and P.H. Eaton. Single-event transient pulse quenching in advanced cmos logic circuits. *Transactions on Nuclear Science*, 56(6):3050–3056, 2009.

[135] D. Yankang, C. Shuming, and L. Biwei. Impact of pulse quenching effect on soft error vulnerabilities in combinational circuits based on standard cells. *Microelectronics Journal*, 44(2):65–71, 2013.

[136] J.-L. Autran, D. Munteanu, G. Gasiot, P. Roche, S. Serre, and S. Semikh. *Soft-error rate of advanced SRAM memories: Modeling and monte carlo simulation.* INTECH Open Access Publisher, 2012.

[137] UMC memory maker, www.umc.com, 2015.

[138] P. Dorsey. Xilinx stacked silicon interconnect technology delivers breakthrough fpga capacity, bandwidth, and power efficiency. *Xilinx White Paper: Virtex-7 FPGAs*, pages 1–10, 2010.

[139] Altera Corp. Meeting the Performance and Power Imperative of the Zettabyte Era with Generation 10. *Altera White Paper WP-01200-1.0*, 2013.

[140] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam. Mitigation of radiation effects in sram-based fpgas for space applications. *ACM Computing Surveys*, 47(2):37, 2015.

[141] C. Carmichael. Triple module redundancy design techniques for virtex fpgas. *Xilinx Application Note XAPP197*, 1, 2001.

[142] Ichinomiya, Y. and Tanoue, S. and Amagasaki, M. and Iida, M. and Kuga, M. and Sueyoshi, T. Improving the robustness of a softcore processor against seus by using tmr and partial reconfiguration. In *Field-Programmable Custom Computing Machines*, pages 47–54, 2010.

[143] B. Pratt, M. Caffrey, J.F. Carroll, P. Graham, K. Morgan, and M. Wirthlin. Fine-grain seu mitigation for fpgas using partial tmr. *Transactions on Nuclear Science*, 55(4):2274–2280, 2008.

[144] S. Srinivasan, A. Gayasen, N. Vijaykrishnan, M. Kandemir, Y. Xie, and M.J. Irwin. Improving soft-error tolerance of fpga configuration bits. In *International Conference on Computer Aided Design*, pages 107–110, 2004.

[145] B.S. Gill, C. Papachristou, and F.G. Wolff. A new asymmetric sram cell to reduce soft errors and leakage power in fpga. In *Design, Automation and Test in Europe Conference and Exhibition*, pages 1–6, 2007.

[146] M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K.A. Label, M. Friendlich, H. Kim, and A. Phan. Effectiveness of internal versus external seu scrubbing mitigation strategies in a xilinx fpga: Design, test, and analysis. *Transactions on Nuclear Science*, 55:2259–2266, 2008.

[147] A. Sari, M. Psarakis, and D. Gizopoulos. Combining checkpointing and scrubbing in fpga-based real-time systems. In *VLSI Test Symposium*, pages 1–6, 2013.

[148] A. Sari and M. Psarakis. Scrubbing-based seu mitigation approach for systems-on-programmable-chips. In *Field-Programmable Technology*, pages 1–8, 2011.

[149] G. A Vera, S.H. Ardalan, X. Yao, and K. Avery. Fast local scrubbing for field-programmable gate array's configuration memory. *Journal of Aerospace Information Systems*, 10(3):144–153, 2013.

[150] Les Jones. Single event upset (seu) detection and correction using virtex-4 devices. *Xilinx Application Note XAPP714*, 1, 2007.

[151] Altera Corp. Enhancing Robust SEU Mitigation with 28-nm FPGAs. *Altera White Paper WP-01135-1.0*, 2010.

[152] Sang Phill Park, Dongsoo Lee, and Kaushik Roy. Soft-error-resilient fpgas using built-in 2-d hamming product code. *Transactions on Very Large Scale Integration (VLSI) Systems*, 20(2):248–256, 2012.

[153] M. Lanuzza, P. Zicari, F. Frustaci, S. Perri, and P. Corsonello. A self-hosting configuration management system to mitigate the impact of radiation-induced multi-bit upsets in sram-based fpgas. In *International Symposium on Industrial Electronics*, pages 1989–1994, 2010.

[154] Xilinx Corp. LogiCORE IP Soft Error Mitigation Controller v3.4. *Product Guide PG036*, 2012.

[155] James S Plank. Erasure codes for storage applications. In *File and Storage Technologies*, pages 1–74, 2005.

[156] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM computer communication review*, 27(2):24–36, 1997.

[157] J.S. Plank, M. Thomason, et al. A practical analysis of low-density parity-check erasure codes for wide-area storage applications. In *Dependable Systems and Networks*, pages 115–124, 2004.

[158] J.M. Park, E.K.P. Chong, and H.J. Siegel. Efficient multicast stream authentication using erasure codes. *ACM Transactions on Information and System Security*, 6(2):258–285, 2003.

[159] C. Hu and S. Zain. Nseu mitigation in avionics applications. *Xilinx Application Note XAPP1073*, pages 1–12, 2011.

[160] K. Rupnow, W. Fu, and K. Compton. Block, drop or roll (back): Alternative preemption methods for rh multi-tasking. In *Field Programmable Custom Computing Machines*, pages 63–70, 2009.

[161] D.K. Pradhan and N.H. Vaidya. Roll-forward and rollback recovery: Performance-reliability trade-off. In *Fault-Tolerant Computing Symposium*, pages 186–195. IEEE, 1994.

[162] M. Ebrahimi, S.G. Miremadi, and H. Asadi. Sctmr: a scan chain-based error recovery technique for tmr systems in safety-critical applications. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 1–4, 2011.

[163] Xilinx Corp. Virtex-6 FPGA Configuration User Guide. *User Guide UG360 (V3.6)*, 2013.

[164] Ken Chapman. Seu strategies for virtex-5 devices. 2010.

[165] M. Ebrahimi, F. Oboril, S. Kiamehr, and M.B Tahoori. Aging-aware logic synthesis. In *International Conference on Computer-Aided Design*, pages 61–68, 2013.

[166] R. Bishnoi, M. Ebrahimi, F. Oboril, and M.B Tahoori. Read disturb fault detection in stt-mram. In *International Test Conference (ITC)*, pages 1–7, 2014.

[167] R. Bishnoi, M. Ebrahimi, F. Oboril, and M.B Tahoori. Architectural aspects in design and analysis of sot-based memories. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 700–707, 2014.

[168] M.S. Golanbari, S. Kiamehr, M. Ebrahimi, and M.B. Tahoori. Aging guardband reduction through selective flip-flop optimization. In *European Test Symposium*, pages 1–6, 2015.

[169] R. Bishnoi, M. Ebrahimi, F. Oboril, and M. Tahoori. Improving write performance for stt-mram. *IEEE Transactions on Magnetics*, 2016.