STUDY OF SINGLE EVENT TRANSIENT ERROR MITIGATION

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Electrical and Computer Engineering
University of Saskatchewan
Saskatoon

By
Hao Xie

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Electrical and Computer Engineering
University of Saskatchewan
57 Campus Drive,
Saskatoon, Saskatchewan
Canada, S7N 5A9

ABSTRACT

Single Event Transient (SET) errors in ground-level electronic devices are a growing concern in the radiation hardening field. However, effective SET mitigation technologies which satisfy ground-level demands such as generic, flexible, efficient, and fast, are limited. The classic Triple Modular Redundancy (TMR) method is the most well-known and popular technique in space and nuclear environment. But it leads to more than 200% area and power overheads, which is too costly to implement in ground-level applications. Meanwhile, the coding technique is extensively utilized to inhibit upset errors in storage cells, but the irregularity of combinatorial logics limits its use in SET mitigation. Therefore, SET mitigation techniques suitable for ground-level applications need to be addressed.

Aware of the demands for SET mitigation techniques in ground-level applications, this thesis proposes two novel approaches based on the redundant wire and approximate logic techniques.

The Redundant Wire is a SET mitigation technique. By selectively adding redundant wire connections, the technique can prohibit targeted transient faults from propagating on the fly. This thesis proposes a set of signature-based evaluation equations to efficiently estimate the protecting effect provided by each redundant wire candidates. Based on the estimated results, a greedy algorithm is used to insert the best candidate repeatedly. Simulation results substantiate that the evaluation equations can achieve up to 98% accuracy on average. Regarding protecting effects, the technique can mask 18.4% of the faults with a 4.3% area, 4.4% power, and 5.4% delay overhead on average. Overall, the quality of protecting results obtained are 2.8 times better than the previous work. Additionally, the impact of synthesis constraints and signature length are discussed.

Approximate Logic is a partial TMR technique offering a trade-off between fault coverage and area overheads. The approximate logic consists of an under-approximate logic and an over-

approximate logic. The under-approximate logic is a subset of the original min-terms and the over-approximate logic is a subset of the original max-terms. This thesis proposes a new algorithm for generating the two approximate logics. Through the generating process, the algorithm considers the intrinsic failure probabilities of each gate and utilizes a confidence interval estimate equation to minimize required computations. The technique is applied to two fault models, Stuck-at and SET, and the separate results are compared and discussed. The results show that the technique can reduce the error 75% with an area penalty of 46% on some circuits. The delay overheads of this technique are always two additional layers of logic.

The two proposed SET mitigation techniques are both applicable to generic combinatorial logics and with high flexibility. The simulation shows promising SET mitigation ability. The proposed mitigation techniques provide designers more choices in developing reliable combinatorial logic in ground-level applications.

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor, Dr. Chen. Without his valuable instruction and inspiration, this thesis would not have been completed. Also, I am grateful for his helpful guidance on how to become a real academic and how to think critically. His encouragement and patience helped me overcome numerous difficulties through my Master's program.

Dr. Adrian Evans gave me a great deal of technical and academic advice. His passion about research deeply inspired me. His profound knowledge of radiation hardening greatly benefited my research project.

I would also like to acknowledge my committee members Professor Gokaraju, Professor Yang and Professor Bui for the constructive feedback and suggestions which made my thesis better in many ways.

Finally, all my success would be impossible without the unconditional love and support from my family.

CONTENTS

Permission to Use i

Abstract ii

Acknowledgements iv

Contents v

List of Tables vii

List of Figures viii

List of Abbreviations ix

1 Introduction 1
- 1.1 Introduction . . . 1
- 1.2 Motivation . . . 6
- 1.3 Objectives . . . 8
- 1.4 Thesis Organization . . . 10

2 Overview of SET Mitigation Techniques 12
- 2.1 Package and Chip Level . . . 12
- 2.2 Core and Module Levels . . . 13
- 2.3 Cell and Process Levels . . . 14
- 2.4 Block Level . . . 15
  - 2.4.1 Coding Techniques . . . 16
  - 2.4.2 Redundant Logic Techniques . . . 16
  - 2.4.3 Rewiring Techniques . . . 17

3 Low Cost Mitigation of Logic Faults with Redundant Wires 19
- 3.1 Signature Based Method to Estimate Controllability and Observability . . . 19
  - 3.1.1 Signature . . . 19
  - 3.1.2 Controllability . . . 21
  - 3.1.3 Observability . . . 22
- 3.2 Redundant Wires . . . 24
  - 3.2.1 Logic Implications . . . 24
  - 3.2.2 Redundant Wires . . . 24

3.2.3 Circuit Modification . . . 25
3.3 Algorithm for Evaluating and Selecting Logic Implications . . . 28
3.3.1 Identification . . . 28
3.3.2 Evaluation . . . 32
3.3.2.1 Evaluation of Zone A . . . 34
3.3.2.2 Evaluation of Zone B . . . 36
3.3.2.3 Evaluation of Zone C . . . 38
3.3.2.4 Evaluation of Overlap Between Implications . . . 40
3.3.3 Selection . . . 41
3.3.4 Summary . . . 43
3.4 Simulation Results . . . 44
3.4.1 Implication Identification . . . 44
3.4.2 Protective Effect Evaluation . . . 46
3.4.3 Protection Results . . . 47
3.4.4 Impact of Synthesis Constraints . . . 49
3.4.5 Impact of Signature Length . . . 50
3.5 Conclusions . . . 51

4 New Approximate Logic based Approaches for Synthesis of Redundant Combinatorial Logic for Selective Fault Tolerance 53
4.1 Fault Model and Evaluation Method . . . 54
4.2 Proposed Algorithm . . . 55
4.3 Simulation Results . . . 59
4.3.1 Simulation Results with SA Fault Model . . . 60
4.3.2 Simulation Results with SET Fault Model . . . 60
4.3.3 Comparing Results between Fault Models . . . 60
4.3.4 Effect of Logic Sharing on Fault Tolerance . . . 62
4.3.5 Results Prediction . . . 64
4.4 Conclusions . . . 65

5 Summary, Conclusion and Future Work 66
5.1 Summary . . . 66
5.2 Conclusion . . . 68
5.3 Future Work . . . 68

LIST OF TABLES

3.1 Observability of Nodes in Example Circuit 23
3.2 Implication Lookup Table 31
3.3 Results of Identification Algorithm 45
3.4 Results of Evaluation Algorithm 47
3.5 Protection Results 50
3.6 Impact of Synthesis Constraints 51
3.7 Impact of Vector Size on Quality of Results 52

4.1 Logic Cubes for Example Circuit 56
4.2 Results after First Round 58
4.3 Summary of SA and SET Results for Benchmark Circuits ($N_{max} = 400$) 63

LIST OF FIGURES

1.1 Charge generation and collection phases in a reverse-biased junction caused by the passage of a high-energy particle. . . . 2
1.2 An air shower created by the collision between a primary cosmic particle and a molecule in the air. . . . 3
1.3 Three intrinsic masking mechanisms. . . . 5
1.4 Prediction of soft error rate against clock frequency[1]. . . . 8

2.1 Summary of SET mitigation techniques. . . . 13
2.2 Triple Modular Redundancy. . . . 14
2.3 The structure of SOI technology. . . . 15
2.4 Basic structure of EDAC code approach implementation 17
2.5 The basic structure of the approximate logic technique. . . . 18

3.1 Example circuit with random generated signatures. . . . 20
3.2 Example circuit with implications. . . . 25
3.3 Netlist manipulation to insert redundant wires. . . . 26
3.4 Zones for protection effect evaluation. . . . 34
3.5 Example of new gates in Zone C 39
3.6 High-level Algorithm Flow. . . . 43
3.7 Failure rate reduction for benchmark S713. . . . 48
3.8 Failure rate reduction for benchmark S1488. . . . 49

4.1 Approximate Logic Functions 53
4.2 Layout of INV and XOR Gate 54
4.3 Transient Sensitivity of 28nm Gates. . . . 54
4.4 Karnaugh Map Examples 55
4.5 High-Level Algorithm Flow. . . . 57
4.6 Area, Power Overheads vs FIT Reduction Results with SA Fault Model (SA) . . . 61
4.7 Area, Power Overheads vs FIT Reduction Results with SET Fault Model. . . . 62
4.8 SET Masking Effect of Redundant Logic Generated Using SA Fault Model (s386 and s1488). . . . 63
4.9 Per Gate - Fault Model Correlation (sao2). . . . 64
4.10 Effect of Logic Sharing (F, H) for sao2 and inc. . . . 64
4.11 Cube Size Distribution for sao2, inc and s1488. . . . 65

LIST OF ABBREVIATIONS

BDD	Binary Decision Diagram
CEC	Concurrent Error Correction
CED	Concurrent Error Detection
CPU	Central Processing Unit
DRAM	Dynamic Random-Access Memory
EDAC	Error detection and correction
EPP	Error Propagation Probability
FIT	Failure In Time, one FIT is equivalent to one failure in one billion hours
FR	Fault Reduction
GMR	Generalized Modular Redundancy
IC	Integrated Circuit
MA	Mandatory Assignment
NP	A complexity class used to describe certain types of decision problems in computational complexity theory
QoR	Quality of Result
SA	Stuck-At
SAT	Satisfiability
SEE	Single Event Effect
SER	Soft Error Rate
SET	Single Event Transient
SEU	Single Event Upset
SIA	Semiconductor Industry Association
SOI	Silicon On Insulator
SOP	Sum Of Product
SRAM	Static Random-Access Memory
TMR	Triple Modular Redundancy

CHAPTER 1

INTRODUCTION

1.1 Introduction

When an integrated circuit (IC) is operating in a radiation environment, it is vulnerable to a strike from various energetic particles, such as protons, neutrons, alpha particles and other heavy ions. As a high-energy particle penetrates through IC materials, extra electrons and holes are generated along the track due to the ionization effect, as shown in Figure 1.1a. These excess carriers will then be collected by the sensitive node nearby in two stages, ion drift (Figure 1.1b) and ion diffusion (Figure 1.1c), and if the charge collected exceeds the charge threshold of the given node, the state of the node is erroneously altered. These adverse interactions between high-energy particles and semiconductor material are denoted as Single Event Effects (SEEs).

Cosmic rays are one of the major radiation sources. They are high-energy particles originating from a sun particle event or other astrophysical processes outside the solar system. Cosmic rays have two phases called primary cosmic rays and secondary cosmic rays. Primary cosmic rays are the original product of astrophysical processes which consist of protons (85%), alpha particles (14%) and other heavier nuclei (1%)[2]. They exist primarily in the space environment. In 1975, Binder et al. identified that primary cosmic rays were the main culprit of single event errors observed in a satellite integrated circuits [3]. This is the first published observation of SEEs. Secondary cosmic rays are a product of collisions that occur when primary cosmic rays enter the atmosphere. The interaction between high-energy particles and molecules in the air generate an air shower of various secondary lighter particles, including neutrons, pions, positrons and muons, as shown in Figure 1.2[4]. The flux of secondary particles has been discovered to be dependent on

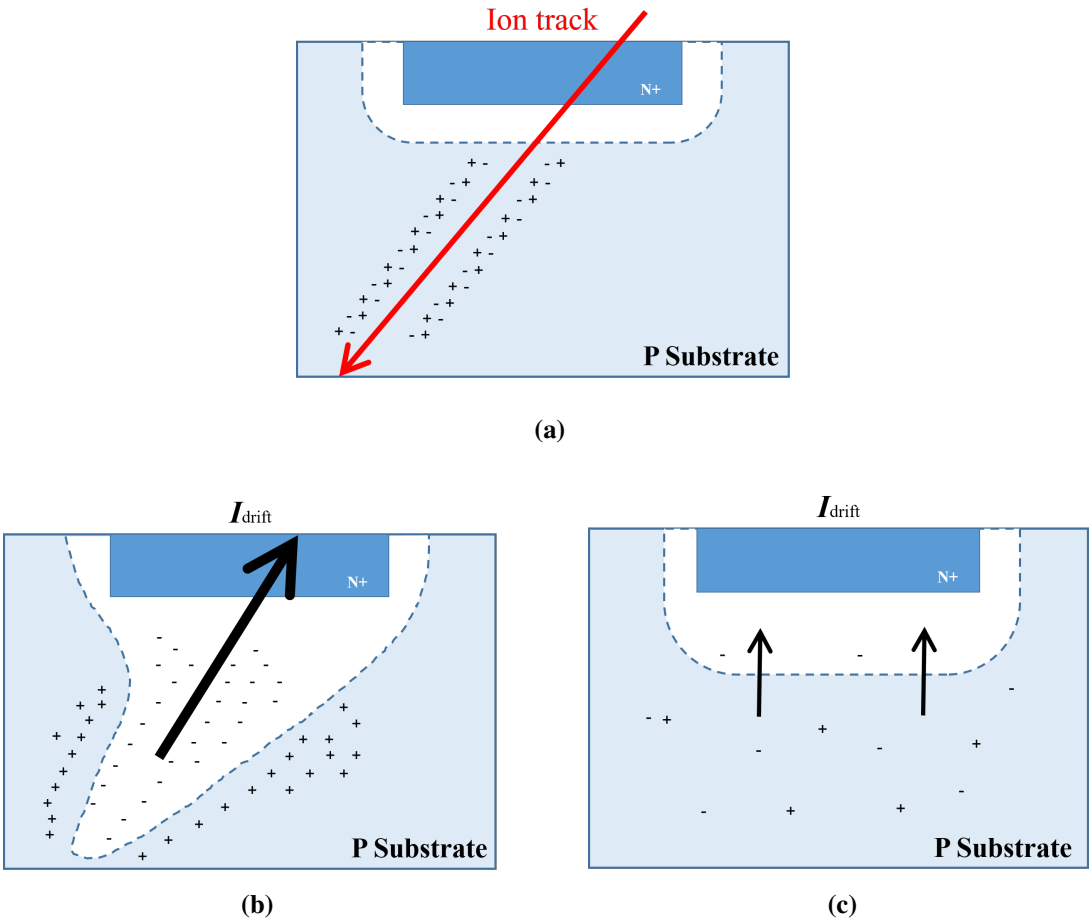


Figure 1.1: Charge generation and collection phases in a reverse-biased junction caused by the passage of a high-energy particle.

both altitude and longitude[5]. The strength of the radiation generally increases with the altitude and longitude. In 1995, Taber and Normand summarized and released a series of experimental results of SEEs at different flight altitudes, and indicated that the range of upset rates on avionics overlaps with the range measured in low-earth orbit [6].

Unpurified packaging materials are another radiation source. The radioactive elements residing in the unpurified materials were found occasionally to emit alpha particles. Alpha particles are +2 particles consisting of two protons and two neutrons. They have a relatively low penetration depth and can be stopped by a few centimeters of air. However, the alpha particles emitted by unpurified packaging materials can easily impact the chips within the package and cause soft errors like SEEs. The first observation of SEEs caused by unpurified materials dates back to the late 1970s in Intel's

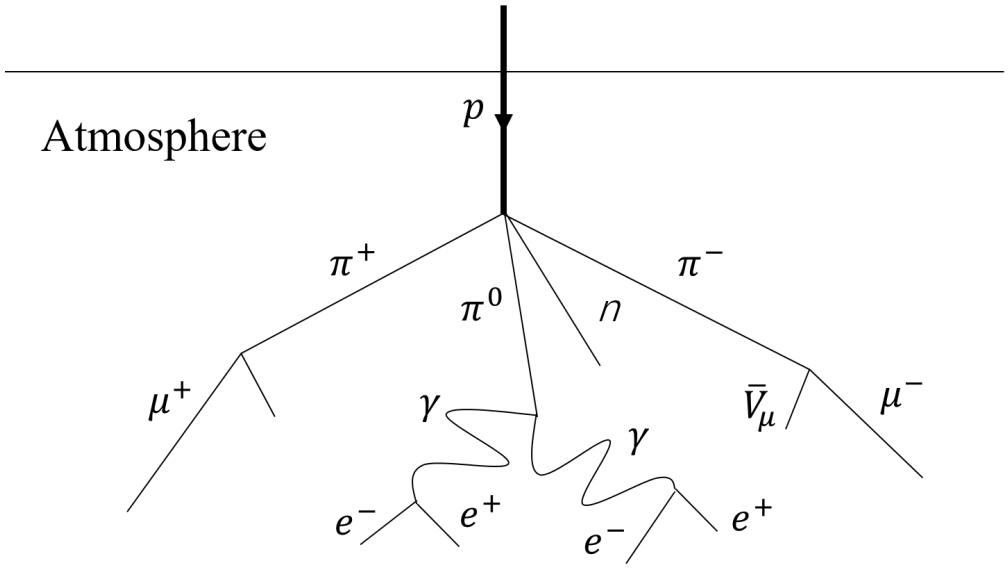


Figure 1.2: An air shower created by the collision between a primary cosmic particle and a molecule in the air.

new Dynamic Random-Access Memory (DRAM) chips [7, 8].

In addition, facilities like nuclear reactors and particle accelerators are artificial radiation sources. Nuclear reactions mainly produce gamma radiation and neutron particles. Particle accelerators produce high-energy protons and electrons which will interact with each other and create radioactive secondary particles. Thus, sensors and control circuits operating in these facilities are highly susceptible to a variety of radiation effects.

To summarize, primary cosmic rays are the major threat to circuits operating in a space environment. Within the atmosphere, primary cosmic particles interact with molecules in the air, forming a cascade of lighter particles. Even though the energy of these secondary particles is attenuated with the distance of penetration, there are still some energetic particles able to reach the surface of the Earth. Thus, devices located from the ground up to flight altitudes are all affected by secondary cosmic rays. Additionally, unpurified packaging materials are an insidious source of radiation, threatening both ground and space devices. In facilities like nuclear reactors and particle accelerators, radioactive particles produced during operation are also a potential threat to nearby devices.

Non-destructive SEEs, which are called soft errors, are classified into two categories based

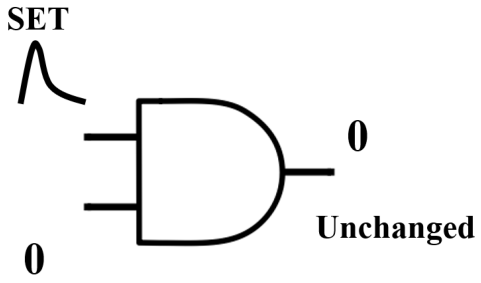
on the area struck by radiation particles: Single Event Upset (SEU) and Single Event Transient (SET). If an energetic particle strikes the sensitive nodes of an element in Dynamic Random-Access Memories (DRAMs), Static Random-Access Memories (SRAMs), latches or flip-flops, and the induced voltage or current transient changes the state of the element, this incorrect state will be preserved until the next writing operation. This phenomenon is referred to as an SEU. Based on the result of a strike, an SEU error can be categorized into one of four groups: masked, corrected, detected or silent [9, 10]. Without protection, the damage of a silent error may vary from a single undesired operation to as serious as a system lockup.

On the other hand, if a radiation particle strikes a sensitive node in combinatorial logic, the formed transient current will propagate along the sensitive paths and finally be latched by sequential elements. This phenomenon is referred to as an SET. For a long time, however, an SET was considered to be a less challenging single event effect than an SEU due to three intrinsic masking mechanisms found in combinatorial circuits, which can inhibit an SET from being latched and which are described later.

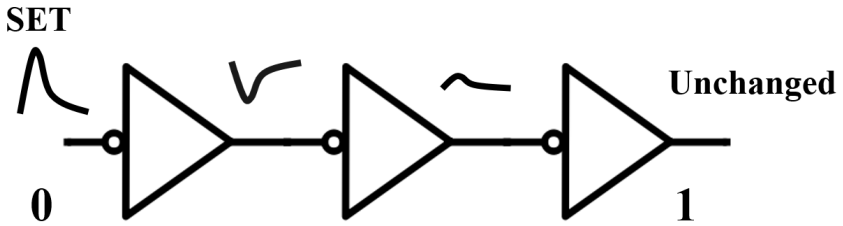
Logical Masking: Logical masking is a mechanism utilizing the concepts of dominant value and don't-care in logical operations. If it is the case that when an input of a gate holds a particular value (1 or 0), and it follows that the output of the gate is always either 1 or 0, notwithstanding the values of other side inputs, then this specific value is denoted as the dominant value and the status of the dominated side inputs are denoted as don't-care. For example, if an SET pulse propagates to an input of an AND gate, but another input occupies the dominant value (0 in this case), the output of the gate will not be affected; therefore, the SET will be masked (Figure 1.3a).

Electrical Masking: Each gate has a logic transition time. If the width of an SET pulse is shorter than this time, the amplitude of the pulse will be attenuated while passing through the gate. Then, this reduced amplitude will decrease the pulse width, and eventually, the pulse will fade away as it propagates, as shown in Figure 1.3b. Generally, pulses wider than the transition time will not be attenuated, pulses shorter than half the time will be eliminated and pulses in-between will be attenuated to some extent [11].

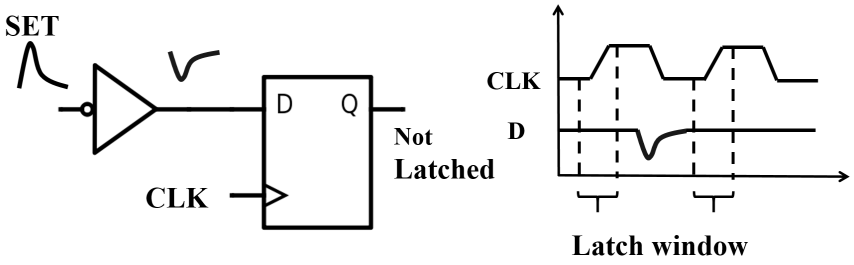
A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1



(a) Logical masking



(b) Electrical masking



(c) Temporal masking

Figure 1.3: Three intrinsic masking mechanisms.

Temporal Masking: When an SET arrives at the input of a memory element, its pulse has to cover completely the latching window so that this SET can be latched. In other words, it has to arrive before the setup time and last until the hold time; otherwise it will be stopped (Figure 1.3c). Buchner et al. proposed the concept of "window of vulnerability" to define this time interval when the memory element is sensitive to SET pulses [12].

Thus, for an SET induced error to be latched, the radiation particle must strike the sensitive area of a node and collect sufficient charge to generate an SET pulse of sufficient duration and amplitude. There also must exist an active path allowing the pulse to reach storage, and the storage must be

within the window of vulnerability to latch the error. Thus, the probability of SET induced soft error P_{SET} can be evaluated by Equation 1.1 [13, 14]

$$P_{SET} = (Flux) \times \sum_{i=1}^{n} A_i Q_i P_{prop} \times T_{mask} \tag{1.1}$$

where $Flux$ is the flux of high-energy particles, which is independent of technologies, A_i is the proportion of sensitive area of a node i, Q_i is the probability that sufficient charge at node i can be collected to form an SET pulse with sufficient amplitude and duration to survive from the electrical masking, P_{prop} is the probability that the SET pulse can propagate to the storage cells, and T_{mask} is the probability that the SET is latched.

1.2 Motivation

Moore's Law projects that the performance of an integrated circuit will double approximately every 18 months. This proved to be accurate for almost 50 years until the technology scaled down to 22nm around 2012 [15] and progress was close to saturation. This performance improvement resulted technically from the scaling of transistors. As feature size shrinks, transistor density of a chip increases, which allows integrating more functional units into one chip; the power consumption and transition delay also decreases, allowing the emergence of high performance and low energy IC designs. While these advancements enormously contributed towards the wide application of digital devices in a variety of fields, scientists also observed a rising susceptibility to radiation effects, including SETs.

As technology scales, a single transistor's physical area shrinks, which dramatically increases the density of transistors on a chip. In 1978, the Intel 8086 processor, which gave rise to the famous x86 architecture, was first introduced. It was the most advanced processor at that time, was produced with a 3000nm process, and had an average of 88 transistors per square mm. Today, the Intel Broadwell-EP Xeon 22-core processor with the 14nm process has reached an astounding 15.8 million transistors per square mm. Meanwhile, the smaller spacing between devices leads to lower critical charge threshold and larger sensitive area per unit transistor. The increases in transistor

density and sensitive area increase A_i in Equation 1.1, and thus, makes integrated circuits more susceptible to SET effects .

At the same time, technology scaling is weakening combinatorial logic's natural immunity. As briefly introduced in Section 1.1, three intrinsic masking mechanisms inhibit the SET effect. However, technology scaling, especially energy and frequency scaling, has shown varying degrees of influence on these masking mechanisms. First, the scaling of operating frequencies, i.e., more state transitions per unit time, increases the chance that an SET pulse is latched, and consequently, reduces the effect of temporal masking. Electrical masking is also weakened by technology scaling. SET pulses narrower than the unit transition time are likely to be attenuated during propagation or to be filtered thoroughly before latching. However, technology scaling leads to faster transition time on each unit, which in turn allows more SET pulses to propagate through the data path. Experiments have indicated that modern circuit designs cannot rely on natural electrical masking any longer [16]. Unlike temporal and electrical masking, logical masking is rarely influenced by scaling. This is because logical masking is independent of the technology process and is only relevant to a circuit's own logic.

According to the evaluating model represented by Equation 1.1, the SET error rate is mainly determined by five factors: $Flux$, A_i, Q_i, P_{prop} and T_{mask}. $Flux$ and P_{prop} are independent of technology scaling. The ratio of sensitive area A_i increases with a transistor's dimension scaling and density increase. Meanwhile, the weakening of electrical and temporal masking exacerbate Q_i and T_{mask}. Overall, all these changes have rendered increasing SET error rates as technology advances.

Decades of experiment and observation have confirmed the aforementioned deduction. The first SET observation dates back to the early 80's [17] when unit dimensions of main technologies were within a range of microns. During that period, the charge deposit phenomenon was only observed occasionally on limited parts of circuits with exposure to relatively high radioactive energies, and thus, it was not regarded as a major threat by most academics [18]. Following the development of fast circuits (up to 100Mbps) in the 90's, the increasing SET errors became a growing concern to developers of space applications. In 1997, Buchner et al. made the predic-

tion that SET errors will dominate soft errors as frequency increases, as shown in Figure 1.4[1]. Subsequently, an increasing number of studies set out to reveal the characteristics and the trends of SETs, and they all reported the growing threat of SETs in reliability issues[19, 20, 21, 22]. In 2009 and 2010, Sridharan et al. conducted a study to evaluate the soft error on a supercomputer and the surprising result was that the detected error rate on the unprotected devices reached as high as 350 per minute[23]. As of today, the Semiconductor Industry Association (SIA) has clearly identified transient errors in combinatorial logic as a major threat and a huge challenge to robust system design in the future[24].

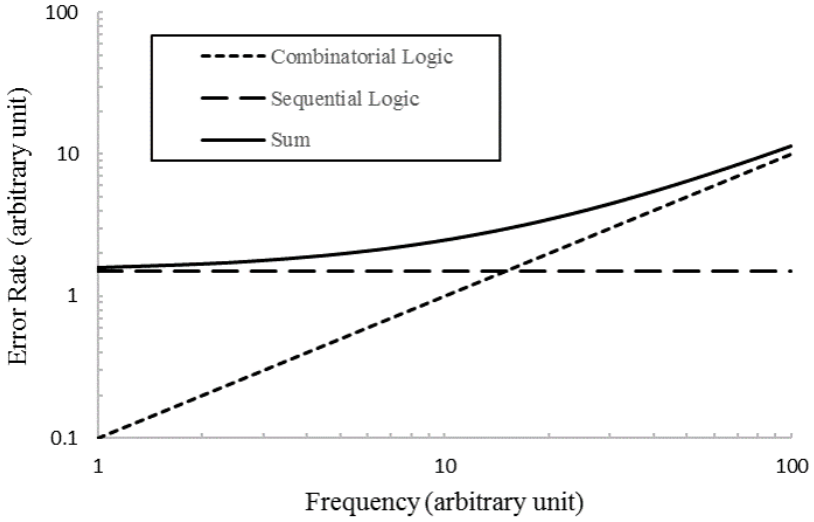


Figure 1.4: Prediction of soft error rate against clock frequency[1].

1.3 Objectives

Well aware of radiation problems including SETs, researchers have put in great effort to make electronic devices resistant to errors caused by high-energy particles, a field known as radiation hardening. Originally, radiation hardening was mainly aimed towards devices operating in space or in high altitude flight, and around nuclear reactors and particle accelerators, where high-energy particles are more intense and reliability is strictly required[25]. Radiation hardening techniques involve all aspects of design, process and testing. Redundancy is a hardening theory widely utilized in component and system design, e.g., Triple Modular Redundancy (TMR). There are also

special processing techniques, like Silicon On Insulator (SOI), that are used to make the sensitive area resistant to collecting charges. Additionally, space and nuclear-related projects may enclose their critical circuits in a shield made of lead or other dense materials. With these technologies, radiation-tolerant devices can resist radiation thousands of times stronger than ground-level commercial devices can [26, 8, 27]. However due to the considerable time and effort required to develop and test a radiation-tolerant design, the performance of radiation-hardened devices is typically two to four generations behind that of commercial devices.

With technology scaling, the sensitive area of a transistor increases, and the critical charge for both combinatorial and sequential logic decreases, which makes new electronic devices more susceptible to SEEs. Without protection, advanced computer chips can easily have an error rate in excess of 50,000 FIT/chip, where one FIT (failure in time) is equivalent to one failure in one billion hours. This rate is higher than the sum of all other reliability issues [24]. The increasing threat makes radiation tolerance a necessity for designs even in applications at ground level. Previously, ground-level design was mainly focused on protecting memory and registers, because they utilize a significant portion of chip area and have dominant error rates. However, as discussed above, with technology and frequency scaling, combinatorial logic errors may eventually dominate, negating the effort of strategies focused on hardening conventional memory[13, 28]. Thus, developers need to reallocate resources in both directions in order to achieve a balanced protecting effect in future designs.

In order to withstand severe radiation environments, space and nuclear related designs have to expend tremendous area, power and delay overheads on radiation hardening. For example, the classic TMR needs more than 200% in area and power overheads as well as two extra layers of delay. It also takes considerable extra time for developing and testing. Nevertheless, all these demanding requirements may not be suitable for ground-level applications. First, ground-level environment is not as severe as environment in space or around nuclear reactors. Then, instead of pursuing extreme reliability, ground-level applications focus more on efficiency in radiation protection. A balance between the protecting effect and overheads needs be achieved. Furthermore, ground-level commercial applications are responsive to the newest technology, and thus, there is

limited time for developing and testing the application of radiation hardening techniques. As a result, a lightweight SET mitigation technique with the following features is the most desirable solution for ground-level radiation applications:

Flexible trade-off: The trade-off between the protecting effect and cost overheads is adjustable. The adjustable trade-off allows designers to achieve different SET coverage under various design constraints.

High efficiency: The technique can achieve remarkable coverage with relatively low overheads.

General purpose: The technique should be applicable to most combinatorial circuits. The generality makes it possible to integrate the technique into automatic design flow with which developers are able to utilize the mitigation technique without a complete knowledge of design.

Fast computation: The technique should be able to find a result close to optimal within reasonable computing time. The fast computation guarantees the feasibility of the technique.

In this thesis, I will propose two generic and high efficient SET error mitigation techniques based on the redundant theory. Since the techniques are aimed at generic combinatorial logic, they can be utilized to most of digital designs. Additionally, the mitigation techniques protect the target design by enhancing the logical masking effect, which is not affected by the technology or radiation environment. The effectiveness of the proposed algorithms is tested by fault injection simulations.

1.4 Thesis Organization

Chapter 1 introduces the basic concepts around Single Event Transient effects. As technology advances, the growing trend of SET errors is explained, which is the motivation for designing soft error mitigation techniques for ground-level applications. The author then presents the objectives of this thesis based on the special demands of ground-level applications.

Chapter 2 reviews state-of-the-art SET mitigation techniques. The techniques are classified based on different levels of design granularity, ranging from the chip level to the process level.

At the package and chip levels, shielding and package purifying are briefly introduced. At the core and module levels, Triple Modular Redundancy and its extensions are discussed. At the cell and process levels, techniques focusing on increasing the resupplying current and limiting charge collection are discussed. Three main block level mitigation approaches, i.e., coding, redundant logic and rewiring, are demonstrated and their features and limits are discussed.

Chapter 3 presents a new algorithm for quickly selecting redundant wires for the purpose of masking logic faults. The approach is general purpose and can be applied to any combinatorial circuit. Since the technique is based on increasing the logical masking of faults, it is thus independent of technology scaling. In Section 3.1, the author briefly reviews the notions of signatures, controllability and observability. Then, in Section 3.2, logic implications, redundant wires and techniques for modifying circuits in order to insert redundant wires are discussed. Following this, in Section 3.3, the author presents the new contribution, a novel algorithm for identifying, evaluating and selecting implications. The simulation results are described in Section 3.4. This section also includes a study of the influence of the synthesis constraints used in the original circuit.

Chapter 4 presents an enhanced approximate logic algorithm that quickly computes a series of redundant logic functions with increasing fault-masking coverage. In Section 4.1, two different fault models, stuck-at model and transient model, are introduced. In Section 4.2, the author describes the proposed algorithm, and in Section 4.3, simulation results for several benchmark circuits evaluated for stuck-at (SA) and transient faults are presented.

Chapter 5 summarizes the two presented SET mitigation techniques and outlines the contributions and future directions.

CHAPTER 2

OVERVIEW OF SET MITIGATION TECHNIQUES

Studies on SET mitigation techniques have been conducted at different levels of design granularity, and researchers have proposed a variety of techniques spanning from the chip level down to the process level, as shown in Figure 2.1. Generally, SET mitigation techniques are classified into two types: fault avoidance and fault correction. Fault avoidance involves isolating radiation sources from devices or constraining the amplitude or width of the pulse so as to minimize the occurrence of the SET actively. Techniques at the package/chip level and the cell/process level mainly fall into this type. On the other hand, fault correction method focuses on maximizing the effects of the three masking mechanisms, which decreases the probability of the fault being latched. Techniques of this type are mainly at the module and block levels that are closely related to the design logic. In this section, a brief overview of the state of the art in SET mitigation is presented.

2.1 Package and Chip Level

Protons and neutrons are two primary sources of radiation particles from cosmic rays[2]. Shielding is the most straightforward and widely applied strategy to protect digital circuits against these particles. For different radiation sources, various shielding materials and strategies are used. For example, an aluminum shielding in the range of 100-250 mils can achieve good protection against protons in a space environment [8]. As for neutrons, materials with low quantivalence have been found to be more effective[4]. Thick shields are required against high energetic particles; nonetheless, it is challenging to eliminate all particles thoroughly. Moreover, this technique is

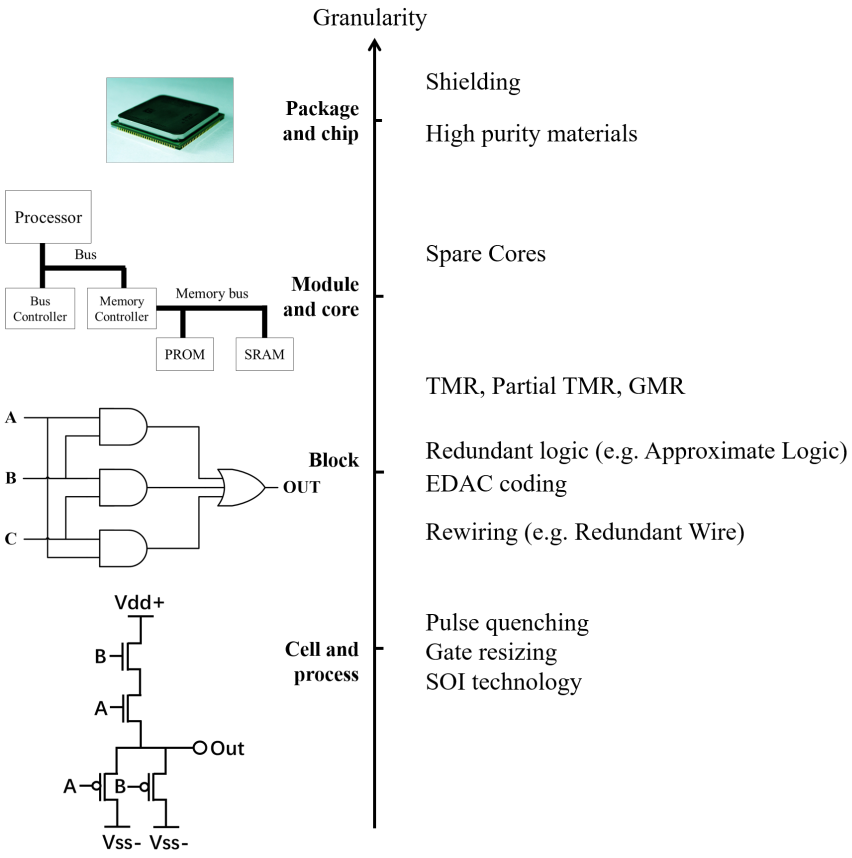


Figure 2.1: Summary of SET mitigation techniques.

not applicable to most ground-level commercial applications. Besides protons and neutrons, alpha particles are another major culprit of SETs, and impure package materials are the primary source of them. To control alpha particle emission, manufacturers adopt high-purity materials and go through dedicated processes during manufacturing. These measures have dramatically decreased the number of SETs induced by alpha particles.

2.2 Core and Module Levels

At the core or module level, Triple Modular Redundancy (TMR) is the most well-known and widely applied SET mitigation technique[26]. As is shown in Figure 2.2, the TMR method initially triplicates the target hardware module and then feeds the outputs of these three modules into a voter circuit. The voter circuit takes the majority result as the final output, which ensures that all single-

bit errors are corrected at the final output. Moreover, multi-bit errors can also be masked if they do not affect the same outputs on more than one copy. Nevertheless, TMR requires more than 200% area and power overhead, which is too costly for many ground-level applications. Consequently, a selective TMR scheme is proposed in [29]. Instead of protecting the whole module, the scheme identifies and triplicates only the part of the module with high error susceptibility. In [30], a module used to select the final result based on the history index is proposed to replace the majority voter. Another recent study [31] has looked at a generalization of the TMR called General Modular Redundancy (GMR). In this technique, the TMR is relaxed by ignoring the output combinations with a low probability of occurrence. However, the experimental results indicate the overheads are still not far below the TMR.

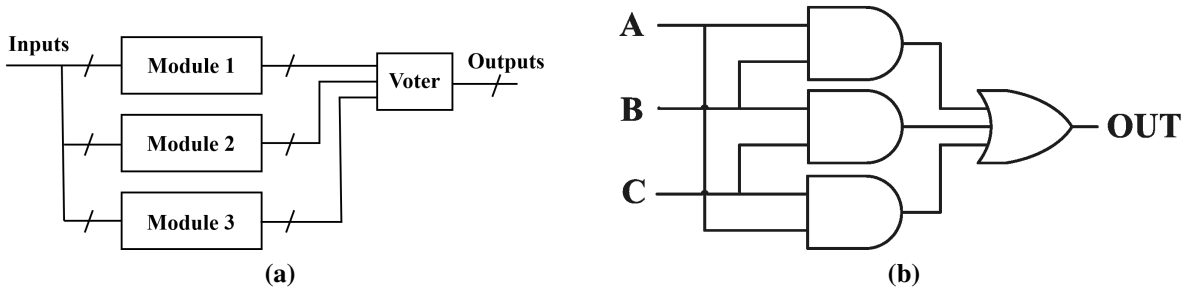


Figure 2.2: Triple Modular Redundancy.

2.3 Cell and Process Levels

At the cell level, researchers have put in great effort to adjust the parameters of layouts such as transistor sizing and spacing. For example, gate resizing techniques prove to be effective in fault avoidance because increasing the gate size can enhance the resupplying current, and in turn, decrease the recovery time of transient pulses [32, 33]. In [34, 35], a method to minimize SETs by replacing high-vulnerability nodes with alternative but insensitive implementations is proposed. In [36], Jonathan et al. reported a new mechanism in which charge collection happened simultaneously on adjacent nodes, behaving in such a way as to quench the voltage transient. This pulse quenching mechanism can effectively constrain the width of SET pulses.

At the process level, studies mainly focus on how to limit the charge collection process at SET origins, since the charge collection process is responsible for the voltage pulse generation and directly influences the magnitude of the transient pulse. Mavis et al. proposed that by delicately selecting node structure and materials, both width and amplitude of SET pulses can be effectively limited[37]. Silicon on insulator (SOI) is a processing technology which places a layer of silicon oxide between substrate and transistors, as demonstrated in Figure 2.3. This isolated structure dramatically reduced the charge collected form adjacent materials. Experimental results has shown that the SOI process has higher radiation tolerance than conventional bulk process[27].

However, the cell and process level techniques require either delicate layout designing or specific technologies, which usually conflicts with ground-level application's need for fast design and high-cost performance.

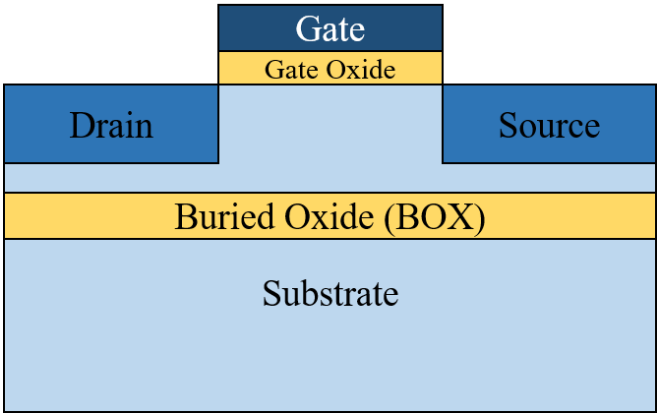


Figure 2.3: The structure of SOI technology.

2.4 Block Level

At the block level, the most common mitigation strategy is redundancy. Either physical or logical redundancy protects the target circuit by comparing the circuit output with pre-generated redundant information.

2.4.1 Coding Techniques

Error detection and correction (EDAC) codes are a major class of redundant techniques. The coding technique has been widely used in the digital communication area to ensure the correctness of the message during transmission. The parity bit, for example, is the simplest coding form with the ability to detect single-bit errors, and Hamming codes are a coding family with the ability to correct single-bit errors and detect double-bit errors. Besides digital communication, the coding technique has also been applied to and found effective in memory designs due to the memory's regular structure. However, to protect the irregular combinatorial logic, the coding technique usually requires a code generation logic called prediction logic (Figure 2.4). The cost of this prediction logic highly depends on the target logic, the coding method as well as the coverage requirement. In [38, 39, 40, 41], a series of Concurrent Error Detection (CED) methods based on the parity code is proposed. On the other hand, Lo et al. applies both error detection and correction codes to arithmetic circuits[42]. A customized Concurrent Error Correction (CEC) code is presented to balance the protective coverage against the overhead incurred[43].

2.4.2 Redundant Logic Techniques

Redundant logic is another major class of mitigation techniques. A typical example is the TMR method discussed on page 13. At the block level, in order to reduce TMR's high overhead, researchers proposed a group of partial TMR techniques named approximate logic. Approximate logic consists of an approximation of the on-set and the off-set of the target combinatorial logic. The generated approximations and the original logic will be connected to a checker logic. The checker is a two-layer logic with an AND/OR structure, and its output is the final output of the circuit, as shown in Figure 2.5. Compared to the conventional TMR, the approximate logic technique has more flexibility over cost control, whereas selecting a good approximation requires skillful maneuvers.

The approximate logic technique was first published in [44]. It uses Binary Decision Diagrams (BDDs) to represent the target circuit and generates the approximations by repeatedly pruning

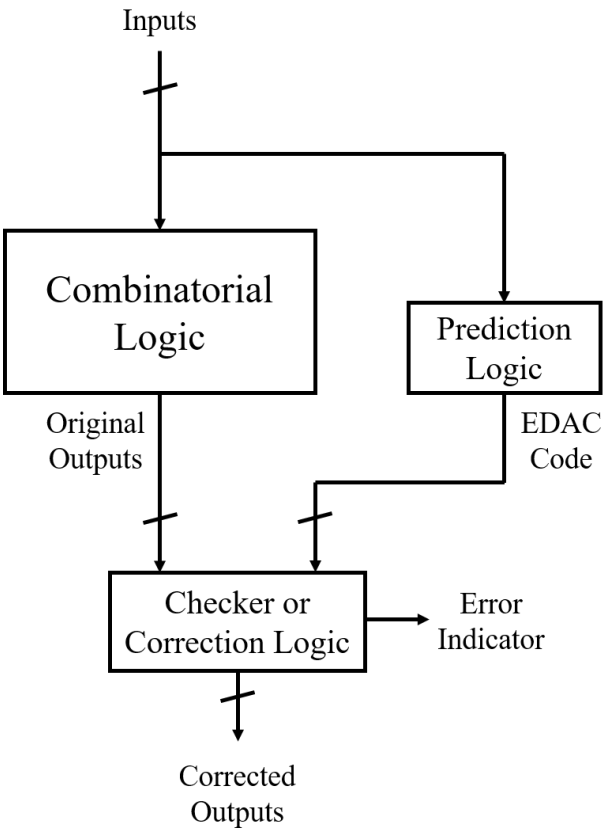


Figure 2.4: Basic structure of EDAC code approach implementation

the longest branch from the BDD tree. It assumes pruning the longest branch will remove the logic taking the most overhead while losing the least coverage. However, this technique does not scale well to large circuits since the BDD representation is not capable of handling large circuits. Choudhury et al. solved this limitation by utilizing a circuit partitioning technique[45, 46]. On the other hand, a technique which prunes the original circuit netlist and gives fault coverage estimation dynamically is proposed by [47, 48]. While this method results in better scalability, the flexibility may be limited by the structure of the original netlist. In addition, the approximate logic is capable of masking timing faults [49].

2.4.3 Rewiring Techniques

Unlike the coding and the approximate logic methods generating redundancy on logic blocks, the rewiring technique focuses on partially restructuring the circuit to increase the overall soft error

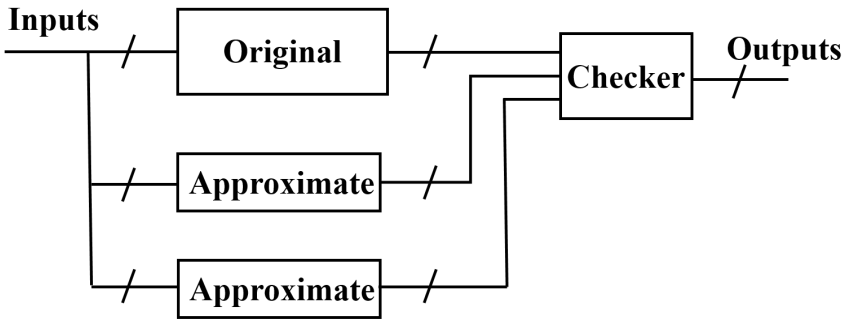


Figure 2.5: The basic structure of the approximate logic technique.

rate (SER) resilience. The restructuring process mainly falls into two categories: local rewriting and redundant wire addition, where the local rewriting refers to partially changing the circuit connections while keeping the circuit's original functionality, and the redundant wire addition, as the name indicates, focuses on adding redundant wires to protect the critical nodes identified. The local rewriting method results in less area overhead than the redundant wire addition method, while its SER mitigation benefit is relatively lower. Since the rewiring technique operates on a subtler scale than the other two techniques, it is more flexible and efficient, which is particularly suitable to ground-level applications with moderate reliability requirements.

The largest difficulty in implementing the rewiring technique is identifying the critical nodes given a complex netlist. In [50], the redundant wire addition method was first applied to soft error mitigation. Its identification algorithm is based on fault injection simulations, which is known for demanding tremendous simulations, and thus, is quite computationally expensive. In addistion, its results are inaccurate due to the inevitable compromise made with the computational limitation and the unexamined side effects induced by the newly added wires. In [51, 52, 53], techniques based on local rewriting with more efficient identification algorithms are presented. However, they all lack an algorithm which thoroughly and effectively takes all factors (like logical masking benefits as well as newly induced faults after addition or restructuring) into consideration, and which gives an accurate estimate about the gross benefits achieved in order to guide the SER hardening process.

CHAPTER 3

LOW COST MITIGATION OF LOGIC FAULTS WITH REDUNDANT WIRES

Chapter 2 reviewed state-of-the-art SET mitigation techniques classified based on the design granularity. In this and next chapters, two novel block-level mitigation methods belonging to rewiring and redundant logic techniques separately are presented.

3.1 Signature Based Method to Estimate Controllability and Observability

For each internal node of a digital circuit, the difficulty of controlling the specific logic value from circuit inputs and observing the value from circuit outputs are defined as controllability and observability, respectively. These two metrics are closely related to and widely applied in circuit testability analysis[54, 55, 56, 57, 58]. Likewise, the algorithms for evaluating and selecting redundant wires rely on accurate determination of controllability and observability metrics for the circuit nodes. In this project, a signature based method which is capable of accurately and efficiently estimating these metrics is introduced. In this chapter, we first review the notion of signatures and then present how they can be used in the computation of controllability and observability metrics.

3.1.1 Signature

The probability of faults propagating through a combinatorial network depends on the state of both inputs and nodes in a circuit. In some cases, probabilities are used to deduce the state

of nodes; however, the drawback of this approach is that the correlation between the states on different nodes is lost. Thus, when calculating the state probability of a node combination, these approaches can easily lead to a significant difference between estimated and real values. In order to reduce the error, more complicated probability reasoning methods are required.

A logic signature is a randomly generated sequence of logic values on all the nodes of a network, and these signatures can be used to deduce the expected state of an individual node or the expected state of a combination of multiple nodes. As shown in [52], logic signatures can be effective in analyzing masking effects.

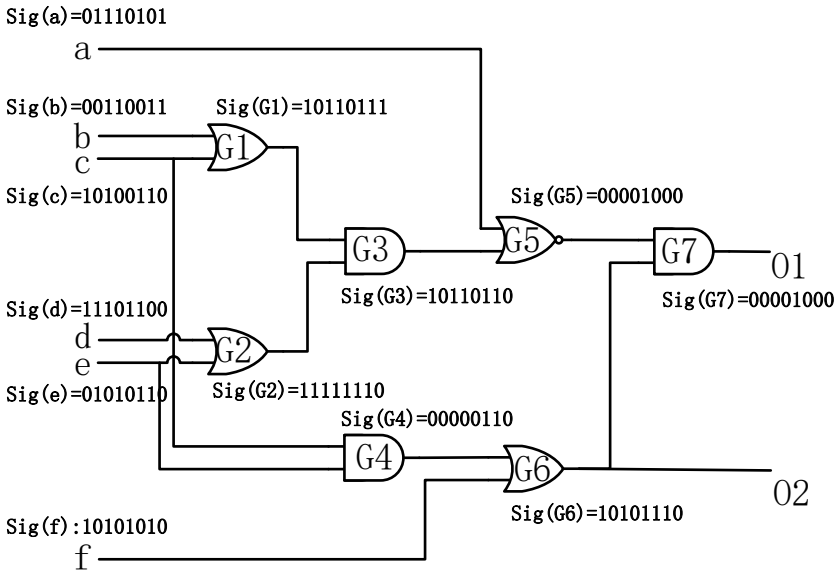


Figure 3.1: Example circuit with random generated signatures.

A K-bit signature of a node n in a circuit C is a K-bit sequence of logic values appearing at node n corresponding to a sequence of K input vectors (see Figure 3.1). The signature provides information about the probability of a node having the logic value of 0 or 1. However, the k^{th} bit in the signature of one node is logically consistent with the k^{th} bit in the signature of another node, which facilitates reasoning about implications. A K-bit signature at node n can be represented as shown in Equation 3.1.

$$Sig_K(n) = F_n(I_1) \cdot F_n(I_2) \cdots F_n(I_K) \tag{3.1}$$

where I_i indicates the i^{th} random input vector, and $F_n(I_i) \in 0,1$ is the local value at n when the

i^{th} input vector is applied.

The algorithm used to generate K-bit signatures for all the nodes in a circuit C is shown in Algorithm 1. *SortTopological()* sorts all the nodes in topological sequence, so that the nodes can be evaluated in a single pass from input to output. *RandomSignal()* generates a random value for the specified input. *EvaluateParallel()* computes the logic value for all of the input values in parallel using the techniques originally described in [59]. The complexity of generating K-bit signatures for a circuit with N nodes is $O(N \cdot K)$.

Algorithm 1 : Signature Generation

```
function GENERATESIGNATURES(C,K) SORTTOPOLOGICAL(C_Nodes)
    for j = 1 to k do
        for i in C_Inputs do
            I_j(i) = RANDOMSIGNAL
        end for
    end for
    for c in Nodes_sorted do
        c = EVALUATEPARALLEL(I_1 ··· I_k )
    end for
end function
```

3.1.2 Controllability

In a given logic circuit, the 1(0) controllability of a given node is directly correlated to the percentage of input vectors justifying the logical value 1(0) on the given node. In this paper, we use a K-bit vector $cc_v(n)$ to estimate the v controllability of node n where $v \in 0,1$. When $cc_v(n)$'s i^{th} bit is 1, it means the value of node n is set to v under the i^{th} input vector; otherwise it is 0. Thus, the computation of the v controllability of a node n is achieved by counting the number of 1s in the the K samples within the vector $cc_v(n)$. Moreover, with the definition of signatures, approximating $cc_v(n)$ with K-bit signatures is straightforward, as shown in Equation 3.2.

$$cc_v(n) \approx \begin{cases} Sig_k(n) & v = 1 \\ \overline{Sig_k(n)} & v = 0 \end{cases} \tag{3.2}$$

3.1.3 Observability

Depending on the input vector, there may or may not exist sensitized paths from a given node, n, to at least one of the primary outputs. In this paper, we use $ob_{T,X}(n)$ to represent the observability of node n to any of the target nodes in the targeting set T while the sensitized path does not pass through any of the nodes in the exclusion set X. When $ob_{T,X}(n)$'s i^{th} bit is 1, it means that under the i^{th} input vector, the value at node n is observable on at least one of the target nodes in T without passing through any nodes in the exclusion set X; otherwise, it is equal to 0. With this generalized definition, the traditional observability of node n can be represented as $ob_{O,\{\}}(n)$, where O stands for all the primary outputs and the exclusion set X is empty. In [52], an algorithm for computing the traditional observability $ob_{O,\{\}}(n)$ is proposed, and here, we extended that algorithm to compute the generalized concept of observability proposed.

This algorithm takes three arguments, C, T and X, as described above and is shown in Algorithm 2.

Algorithm 2 : Observability Calculation

```
function COMPUTEOBS(C,T,X) SORTREVERSETOPOLOGICAL(C_Nodes)
    for t in T do
        ob_{T,X}(t) = All ONES
    end for
    for n in C_Nodes do
        if n ∈ X then
            ob_{T,X}(n) = All ZEROES
        else
            for f in Fanouts of n do
                ob_{T,X}(n) |= COMPUTEOBSCOND(n, f)
            end for
        end if
    end for
end function
```

The method *SortReverseTopological()* sorts all the nodes in circuit C in a reverse topological sequence. The method *ComputeObsCond(n,f)* computes the conditions when node n is observable through direct fanout f. It first decides the non-dominant values at the side inputs of n. Then it gets the signatures of those side inputs and, if the non-dominant value is 0, inverts the corresponding

signature. Finally, those signatures are ANDed with the observability of node *f*.

Table 3.1: Observability of Nodes in Example Circuit

	signature	$ob_{\{O1,O2\},\{\}}$	$ob_{\{O1,O2\},\{G5\}}$
a	01110101	00001000	00000000
b	00110011	00001000	00000000
c	10100110	11011100	01010100
d	11101100	10000000	00000000
e	01010110	00000110	00000100
f	10101010	11111001	11111001
G1	10110111	10001010	00000000
G2	11111110	10000010	00000000
G3	10110110	10001010	00000000
G4	00000110	01010101	01010101
G5	00001000	10101110	00000000
G6	10101110	11111111	11111111
G7	00001000	11111111	11111111

Consider the six inputs, two outputs circuit shown in Figure 3.1. A set of randomly generated signatures and the resulting observability computations are shown in Table 3.1. Two sets of observability are computed, both targeting the same primary outputs but the second one has an excluded gate, *G3*. The $ob_{\{O1,O2\},\{\}}(c)$, for instance, is derived by computing the observability through nodes *G1* and *G4* respectively and then ORing the two results. The $ob_{\{O1,O2\},\{\}}(G1)$ is 10001010 and since *G1* is an *OR* gate whose non-dominant value is 0, the signature of *b* is inverted to 11001100. ANDing these two gives the observability of *c* at the path through *G1* which is 10001000. Similarly, the observability of c at the path through *G4* is 01010100, and finally, ORing these two gives $ob_{\{O1,O2\},\{\}}(c)$ which is 11011100.

If instead we want to compute the observability excluding gate G3, $ob_{\{O1,O2\},\{G3\}}(c)$, since the path through *G1* is blocked by the *G3* node and is no longer observable, $ob_{\{O1,O2\},\{G3\}}(c)$ is equal to the observability of *c* at the path through *G4,* which is 01010100.

Similar to Algorithm 1, the observability algorithm traverses the circuit in a single pass from output to input, and for each node in the circuit, a constant number of basic signature operations is performed. Thus the complexity of computing the observability for a circuit with N nodes is $O(N \cdot K)$.

The application of controllability and the generalized observability will be described later where they are used for selecting implications.

3.2 Redundant Wires

The proposed logic fault mitigation methodology is based on the logic coverage provided by redundant wires. In this chapter, we review the basic notions of logic implications and redundant wires, and how to modify circuits in order to insert redundant wires.

3.2.1 Logic Implications

An implication is a relationship that exists in a circuit between two nodes, called the source (S) and the target (T). If it is the case that whenever S is true (or false), and it follows that T is always either true or false, then there exists a 1-to-1 implication from S to T. Logic implications have been widely used in areas like probabilistic analysis [48], logic optimization [60, 61], design verification [62] and test-pattern generation [63]. In this project, logic implications are used to guide the selection of redundant wires.

Let us consider the example circuit from Figure 3.1, but re-draw it to illustrate the implications in Figure 3.2. It can be seen that when $G3 = 1$, then $G5 = 0$ because *G5* is a *NOR* gate and the dominant value of its inputs is 1. This implication is denoted as $(G3, 1) \rightarrow (G5, 0)$.

3.2.2 Redundant Wires

A redundant wire, as the name implies, is a functionally redundant connection whose logic is fully covered by the rest of the circuit, and thus, the addition of a redundant wire preserves the functionality of the original circuit. As proposed in [50], implications can be used to identify and generate redundant wires. The way redundant wires are inserted in the circuit is introduced in Section 3.2.3. Like many spare redundancy methods used to mask faults, the redundant wire is able to mask logical faults under its coverage from being observed on primary outputs. A model for evaluating the protective effect of a redundant wire is proposed in Section 3.3.2.

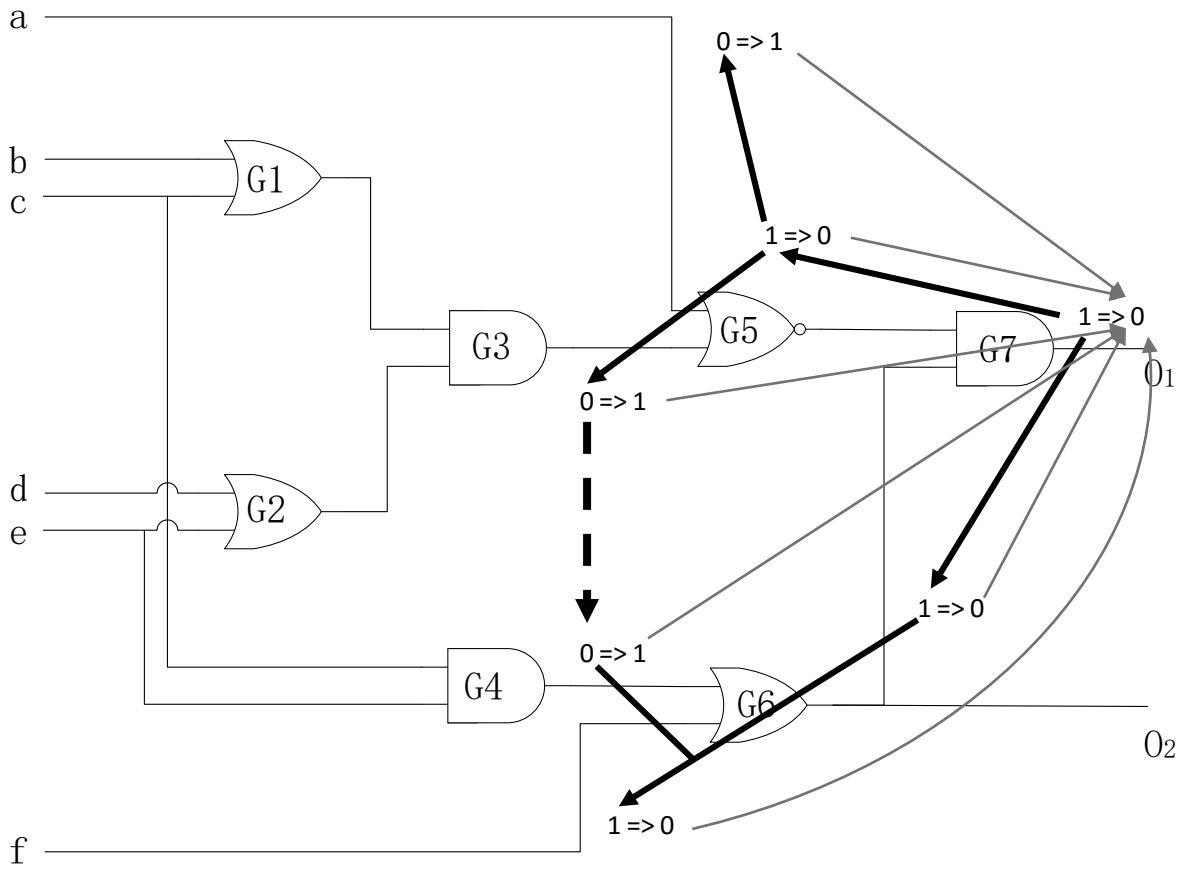


Figure 3.2: Example circuit with implications.

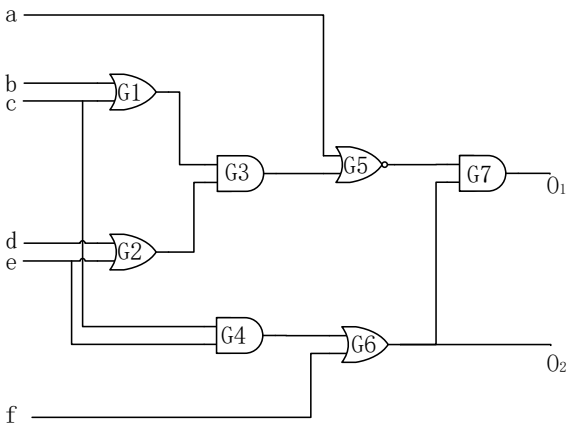
Let us once again consider the example circuit shown in Figure 3.2. It can be seen that when $G3 = 1$, then $G5 = 0$ and $G7 = 0$. Let us suppose that when $G3 = 1$ and $G6 = 1$, there is a fault at *G5* that causes a bit flip $(0 \rightarrow 1)$. This fault would propagate to the output *O1* and generate an incorrect output value (1 instead of 0). If, however, a redundant wire were connected from *G3* to the input of *G7*, this fault would be masked at *G7* and the outputs would be correct.

3.2.3 Circuit Modification

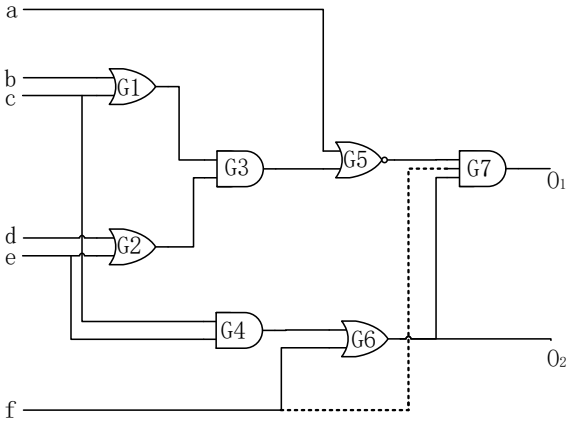
In this section, we discuss how to modify a netlist in order to insert redundant wires, based on a given implication. First, given an implication, the basic idea behind how to insert a redundant wire is to add a connection originating from the source to the input of the target gate. However, in order to preserve the functionality of the original circuit, two necessary conditions must be satisfied. The first condition is that the source of the implication must be out of the fan-out cone of the target. The

reason is straightforward. Suppose the source is a transitive fan-out of the target and it is connected back to the target, a combinatorial loop would be created.

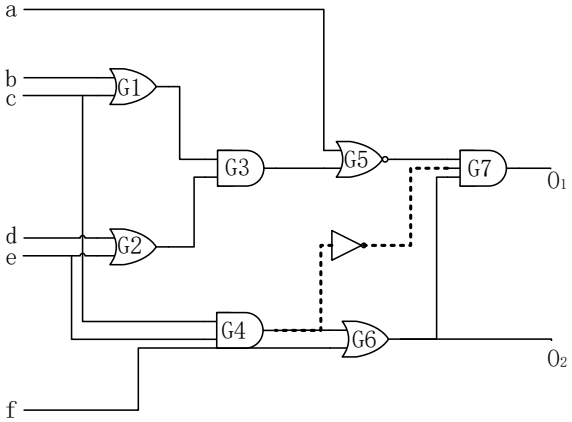
The second condition is that the implicated value must be able to dominate the target. For instance, the value 1 is the dominant value of *OR* and *NAND* gates, and value 0 is the dominant value of *AND* and *NOR* gates. Of course, *XOR* gates have no dominant value and a *NOT* gate has both 1 and 0 as dominant values. This condition combined with the manipulations introduced later guarantee that the redundant wire will always be dominated under fault-free conditions, ensuring that the circuit is not modified.



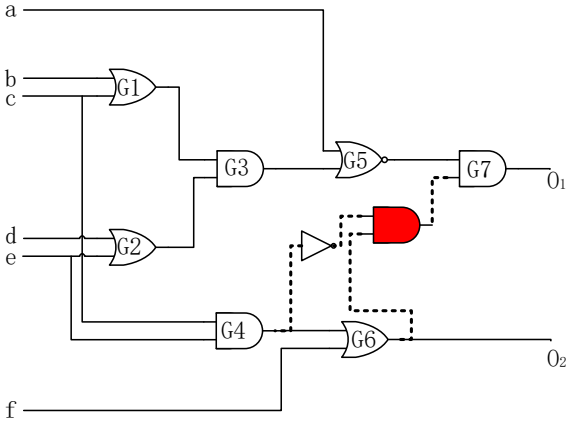
(a) Original circuit



(b) Redundant wire insertion by connecting directly



(c) Redundant wire insertion by adding an inverter



(d) Redundant wire insertion by adding an inverter and a checker

Figure 3.3: Netlist manipulation to insert redundant wires.

For example, in the circuit shown in Figure 3.3, we can identify two implications, $(G4,1) \rightarrow (G7,0)$ and $(G4,1) \rightarrow (G3,1)$. Both of their target gates are *AND* gates, while their implicated values are 0 and 1, respectively. Since the dominant value of an *AND* gate is 0, we can add a

redundant wire by inverting the output of *G4* and adding it to the input of *G7* for the first implication, as shown in Figure 3.3c. But for the second implication, connecting the output of *G4*, whether inverted or not, to the input of *G3* will change the original functionality.

Besides the necessary conditions discussed above, there are two additional factors which influence the addition of redundant wires. The first factor is the implicant value of the implication. If it dominates the target gate, then the source can be connected directly to the input of the target gate. Otherwise, an inverter must be inserted before connecting the source to the target. Figure 3.3b and Figure 3.3c show the insertion of implications $(f,0) \rightarrow (G7,0)$ and $(G4,0) \rightarrow (G7,0)$, which illustrate these two situations. Notice that the second situation introduces a new inverter into the circuit, and consequently, increases the risk of faults. Our evaluation algorithm will take into consideration the new faults that can occur in the gates that are inserted when an implication is mapped.

Another factor is the maximum number of inputs to a single logic gate in the library. When the target gate already has the maximum number of inputs, a gate-separating operation is required. In this case, the target gate is broken into two smaller, equivalent gates. For example, Figure 3.3d, illustrates the insertion of the implication $(G4,1) \rightarrow (G7,0)$, assuming the maximum number of inputs to an *AND* gate is 2. In this case, instead of using a 3-input *AND* gate, we have to separate *G7* into two 2-input *AND* gates and then connect them. Since the prediction of the occurrence and consequence of the gate separation is difficult at the evaluation stage, our evaluation algorithm will ignore the effects caused by this factor. However, during the separating operation, the wires with larger probability to dominate the target will be placed closer to the target, so that the newly added gate will be masked to the utmost, and thus, the influence of the added gate is minimized.

In addition, for implications not meeting the second necessary conditions, our approach is to add a gate as a checker behind the target. This method is also used by [52] and [64]. Depending on the implicated value, either an *AND* gate or an *OR* gate is inserted. Unlike the previous cases, this case introduces one more gate, and hence, leads to extra fault sources as well as larger area, power and timing overheads. Like the situation using inverters, this side effect is considered during the evaluation stage. As a result, when other conditions are equal, implications that do not require

inserting extra gates will be favored over those which require extra gates.

3.3 Algorithm for Evaluating and Selecting Logic Implications

The overall process of protecting a circuit begins with identifying all the implications. Next, we must evaluate the benefit from the implications, and finally, we must select which implications to add. These three steps are described in the following three sections.

3.3.1 Identification

The challenge in using redundant wires to mask faults is first to identify a complete set of implications, and then, to rank them based on their fault-masking capability. In this section, we will focus on the first challenge. Two kinds of identification algorithms have been proposed. [64, 52] present simulation-based approaches. They first run a certain number of random simulations, then using the simulation vectors, they filter out pairs of nodes for which no consistent relation exists. They then use a satisfiability (SAT) solver to verify the remaining pairs. Even though the simulation step narrows down the number of candidate pairs, this strategy still requires a large number of SAT verification operations.

Another approach proposed in [50] is based on justifications. In this methodology, the wires with known values are called justified and those with uncertain values are called unjustified. Justification is the process of inferring the value of an unjustified pin of a gate when the value of some other pins are known and there exists one single possible value for the unjustified pin. Justification can take place in both the forward and backward directions. In terms of using only backward justifications or both, [50] proposed two direct implication methods. Consider *G7* in Figure 3.2 as an example. First, assume that *G7* is set to 1 and then its inputs *G5* and *G6* are justified to 1 because *G7* is an *AND* gate. Further, as *G5* is set to 1, *G3* and *a* are both justified to 0. At this point, no more wires can be justified. Equivalently, by applying the contrapositive, we get implications which could be used for the redundant wire addition. The complexity of forward and backward justification is linear with the number of gates.

However, the direct implication methods could miss a class of implications called indirect implications which cannot be inferred using forward and backward justifications. In the previous example, as *G3* is an *AND* gate and its value is justified to 0, we cannot justify the value of either *G1* or *G2* any longer. Thus, they are considered unjustified. However, if we temporarily set *G1* to 0, we could justify *b*, *c* and *G4* to 0. Similarly, if *G2* is temporarily set to 0 then *d*, *e* and *G4* will be justified to 0. We notice that *G4* is always equal to 0 when *G3* is 0 regardless of the value of *G1* or *G2,* and thus, there exists an indirect implication $(G3,0) \rightarrow (G4,0)$.

During the process of looking for indirect implications, we must temporarily inject logical values at certain nodes in order to conduct further justifications and check the consequences at the end. This kind of technique is called learning, as defined by [63]. In [50], they propose their third variant of identification algorithms based on this recursive learning technique. With unlimited recursion depth, this method has the ability to identify all direct and indirect implications, but its complexity increases exponentially with the recursion depth. In order to trade off for execution time, they limited the recursion depth.

In this section, a novel identification technique utilizing the law of contrapositive and implication lookup tables is proposed. Since implications follow the law of contrapositive, i.e., negating and switching the two parts of an implication is logically equivalent to its origin, an implication can be identified whenever its original or contraposed form is identified. Furthermore, we found that the contraposed forms of most indirect implications are direct implications. For example, in the circuit in Figure 4, $(G3,0) \rightarrow (G4,0)$ is an indirect implication while $(G4,1) \rightarrow (G3,1)$ is a direct implication. Thus, in order to save identified implications, in our approach, shown in Algorithm 3, we introduce an implication lookup table T in which each element represents an implication relationship between two corresponding nodes. The nodes in circuit C are sorted in a topological order. For each node n in this order, we search the lookup table for identified implications which imply node n, like $(m,u) \rightarrow (n,0)$. By applying the contrapositive law, a new implication $(n,1) \rightarrow (m,\bar{u})$ is identified, and thus, $m = \bar{u}$ is justified. After the table has been searched, further direct justifications are applied based on all the justified assignments. As shown in the pseudo-code, the above procedure must be repeated twice, one for implicant $n = 0$ and one for $n = 1$.

Algorithm 3 : Low Effort Implication Identification

```
function IDENTIFYIMPLICATIONSLOWEFFORT(C)
    SORTTOPOLOGICAL(C_Nodes)
    Create and Init Implication Table T
    for n in C_nodes do
        for m prior to n do
            if T[(m = u) → (n = 0)] is true then
                JUSTIFY(m = ū)
                T[(n = 1) → (m = ū)] ← true
            end if
        end for
        Further justifications
        for m prior to n do
            if T[(m = u) → (n = 1)] is true then
                JUSTIFY(m = ū)
                T[(n = 0) → (m = ū)] ← true
            end if
        end for
        Further justifications
    end for
end function
```

Table 3.2 is an example of the implication lookup table derived from the circuit in Figure 3.1, which illustrates how to identify implications with the proposed algorithm. For simplicity, only the referenced items are shown. In the table, each line represents an implicant source and each column represents a conclusion. A value T(true) in a cell means the corresponding implication exists and is identified. At the start, the full table is initialized to false. Based on the topology, the algorithm starts with node c. First, it looks for cells that are set to T in the column with $c = 1$ but there is no such cell at the start. Then, it tries to perform backward and forward justification based on the assignment $c = 0$ and finds the justified assignment $G4 = 0$. Thus, this implication is saved into the cell in the first line ($c = 0$) and fifth column ($G4 = 0$). Similarly, we identify and record the implications with implicants $c = 1$, $e = 0$, and $e = 1$. When it comes to identifying implications with implicant $G4 = 1$, by looking up the fifth column ($G4 = 0$) in the table, the algorithm finds two implications, thus $c = 1$ and $e = 1$ are justified. Based on the justified assignments $G4 = 0$, $c = 1$ and $e = 1$, assignments $G1 = 1$, $G2 = 1$ and $G3 = 1$ are justified and all these implications are marked in the fifth row ($G4 = 0$) of the table. At the end, with this lookup table, indirect

implications $(G3,0) \rightarrow (G4,0)$ can be easily identified by looking up the last column ($G3 = 1$). Finally, all the implications are identified and ready for use in the next phase.

Table 3.2: Implication Lookup Table

		c		e		G4		G1		G2		G3	
		0	1	0	1	0	1	0	1	0	1	0	1
c	0					T							
	1								T				
e	0				T								
	1										T		
G4	0												
	1		T		T				T		T		T
G1	0	T				T						T	
	1												
G2	0			T		T						T	
	1												
G3	0					T							
	1								T		T		

Even though the proposed algorithm (*identifyImplicationLowEffort*) is able to identify efficiently both direct and indirect implications, the results are dependent on the topological sequence, i.e., different topological sequences lead to different results. For instance, in the example circuit, *G3* and *G4* have no dependence upon each other, and thus, their order in the topological sequence is uncertain. If *G3* was prior to *G4*, opposite from the order used in the previous example, the indirect implication will not be identified. To solve this problem, we propose an improved algorithm with the help of a trace-back mechanism (*identifyImplicationHighEffort*, see Algorithm 4). Unlike the previous algorithm which processes all the nodes in a single pass, the improved algorithm maintains a stack. Normally, each node will be processed in the same topological order as in the low-effort algorithm. However, if a new implication is identified, but the implicant node n of its contraposed form has already been processed, the algorithm will push a new task to re-visit node n at the top of the task stack, because this contraposed implication was not identified when n was processed previously. In the next loop, the algorithm will go back and redo the justification with the newly found assignments. Because of the implication lookup table, once an assignment is justified, there is no need to justify it again when the algorithm returns to process a previously

processed node.

The two proposed algorithms both run in linear time with respect to the number of implications in the circuit, since the implication lookup table avoids unnecessary justifications. In terms of space complexity, the implication lookup table is a two-dimension array with the size of $2n \times 2n$, where n is the number of nodes in the circuit. Note also that, unlike other implication identification methods which target only identifying implications with certain assignment as implicant, the proposed methods need first to find out all the implications and then to check the lookup table for implications demanded. This may be unnecessary under certain circumstances, but it is required with our system because it needs to list all the implications for further evaluation and selection.

Algorithm 4 : High-Effort Implication Identification

```
function IDENTIFYIMPLICATIONSHIGHEFFORT(C)
    INITIALIZE(Implication Table T)
    INITIALIZE(Stack S)
    SORTREVERSETOPOLOGICAL(C_Nodes)
    while S not empty do
        n ← S.pop()
        SEARCHIDENTIFIEDIMPLICATION(T, n = 0)
        DIRECTIMPLICATION(n = 0)
        SEARCHIDENTIFIEDIMPLICATION(T, n = 1)
        DIRECTIMPLICATION(n = 1)
        for new implication (n = v) → (m = u) do
            if m < n in topological order &&
                (m = ū) → (n = v̄) not found then
                S.push(m)
            end if
        end for
    end while
end function
```

3.3.2 Evaluation

When using redundant wires to protect logic against transient faults, assessing the protective effect of each implication is one of the core problems. It directly helps guide the candidate selection procedure, and hence, significantly influences the final protective performance. In [64, 50], the assessment of implications is performed by fault injection simulations. However, the execution

time of fault injections depends on many factors, such as the number of input vectors, the number of nodes in the circuit and the number of implications. Only a modest number of simulations can be performed, which will limit the accuracy. On the other hand, in [52, 32, 53], mathematical models for implication evaluation were proposed. In this section, a novel evaluation model quantitatively and systematically taking multiple factors into account is introduced.

Implications can be generally classified into three groups on the basis of the relative position between *S* and *T*. They are:

Group 1: S is within the fan-in cone of T;

Group 2: S and T only share part of their fan-in cones;

Group 3: T is within the fan-in cone of S.

As discussed in Section 3.2.3, implications in the third group are not relevant since adding redundant wires will form combinatorial loops. As for the first two groups, although *S* and *T* have different relative positions, in our evaluation model they are treated equivalently. In our model, a circuit is divided into three zones based on the effects that gates in each zone may bring about. In Figure 3.4, an example implication and its three effect zones are illustrated. *Zone A* is the fan-in cone of the target *T*. Gates in this zone are under the coverage of the implication and are potentially protected by the corresponding redundant wire. *Zone B* includes the source and its transitive fan-ins. Unlike *Zone A*, the gates in this zone may become more sensitive because inserting the redundant wire will introduce extra fan-out paths from them. Also, notice that *Zone A* and *Zone B* are not mutually exclusive. For implications belonging to *Group 1*, *Zone B* is entirely included in *Zone A*. But for implications in *Group 2*, *Zone A* and *Zone B* partially overlap. *Zone C* includes newly added gates due to the redundant wire insertion. Based on the type of the implication to be inserted, it may contain an inverter (NOT), a checker (AND or OR), both or empty. As these newly added gates will introduce new fault sources, this zone has an adverse impact on the final protective effect.

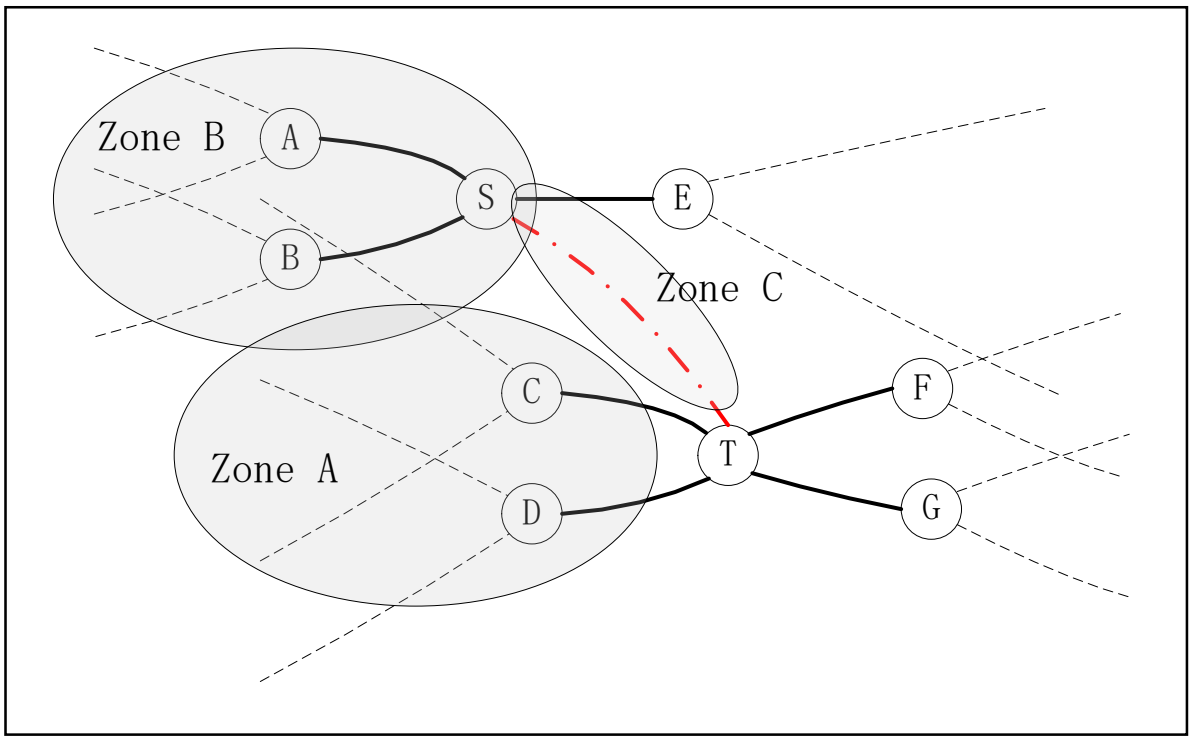


Figure 3.4: Zones for protection effect evaluation.

3.3.2.1 Evaluation of Zone A

Zone A is the transitive fan-in cone of the target. Under different input combinations, different paths within this zone are sensitized and all the gates on the sensitized paths will control the logic value of the target. If a fault occurs in a gate along these paths, it will affect the target and may propagate further causing an error at the primary outputs. However, with a redundant wire originating from a source to this target, this fault will be masked at the target and thus prevented from further propagation. Clearly, the bigger this probability is, the better the candidate redundant wire protects the given gate. Based on these observations, three conditions for a fault occurring at a gate in Zone A to be masked by the corresponding redundant wire can be identified.

AC1: The source must hold the implicant value;

AC2: There must exist at least one sensitized path from the fault to at least one primary output, while all of these paths must pass through the target;

AC3: The fault must not distort the source and the target simultaneously.

The first and the third conditions are straightforward. If the source value is not the implicant value or it is modified by the fault, it cannot dominate the target, and thus, the redundant wire will not stop the fault. The second condition addresses two requirements. First, the existence of sensitized paths guarantees that this fault could be observed in the absence of protection. Otherwise, protecting this fault is unnecessary. Second, if we assume that there exist sensitized paths that do not pass through the target, the fault will propagate through these paths despite the protection provided by the redundant wire.

Given these three conditions, the probability of a fault occurring at gate g to be masked by the implication i can be represented as shown in Equation 3.3.

$$P_{protect}(g,i) = P(AC1 \cap AC2 \cap AC3) \tag{3.3}$$

where $P(AC1 \cap AC2 \cap AC3)$ is the probability of the intersection of all three conditions. Furthermore, the condition *AC1* can be represented by $cc_u(S)$, where u is the implicant value.

As for the condition *AC2*, its probability can be evaluated by $ob_{\{all\,ouputs\},\{\}}(g) - ob_{\{all\,outputs\},\{T\}}(g)$. Here, as defined in Section 3.1, $ob_{\{all\,outputs\},\{\}}(g)$ is a K-bit sequence corresponding to the signature and the i^{th} 1 in this vector means gate g is observable to at least one primary output under the i^{th} input vector of the signature. Similarly $ob_{\{all\,outputs\},\{T\}}(g)$ is a K-bit sequence but the 1s in this vector represent the condition that there exist sensitized paths from gate g to any output without passing through the target T. The subtraction generates a binary sequence with the same length where its i^{th} bit is set to 1 if and only if the i^{th} bit in the first vector is 1 and the corresponding bit in the second vector is 0. Specifically, a 1 in the i^{th} position of this vector means that there exist sensitized paths from gate g to primary outputs and all of these paths pass through the target T.

Finally, the condition *AC3* can be evaluated by $ob_{\{T\},\{\}}(g) - ob_{\{S\},\{\}}(g)$, which represents the probability that a fault on gate g is observable to the target T while not observable to the source S.

With the proposed signature representations for each condition, the probability can be calculated by Equation 3.4. This equation represents the ratio of the number of input vectors that meet all three conditions to the number of bits in the signature, and thus, is an estimate of the probability that faults in a specific gate will be masked by a specific redundant wire.

$$P_{protect}(g,i) \approx \frac{\left|cc_u(s)\&(ob_{\{all\,outputs\},\{\}}(g) - ob_{\{all\,outputs\},\{T\}}(g))\&(ob_{\{T\},\{\}}(g) - ob_{\{S\},\{\}}(g))\right|}{K} \tag{3.4}$$

With the above single-gate protective effect estimation equation, the protective effect of an implication for all gates in *Zone A* can be estimated by Equation 3.5.

$$Effect_{ZoneA}(i) \approx \sum_{g \in ZoneA} P_{protect}(g,i) \times f_g \tag{3.5}$$

where f_g is the failure rate for gate g.

Let us take the implication $(G4,1) \rightarrow (G5,0)$ from Figure 3.2 as an example. The *Zone A* of the target *G5* includes three gates: *G1*, *G2* and *G3*. For gate *G3*, the condition *AC1* will be $cc_1(G4) =$ 00000110, and the condition *AC2* will be$ob_{\{O1,O2\},\{\}}(G3) - ob_{\{O1,O2\},\{G5\}}(G3) = 10001010 -$ $00000000 = 10001010$. As for the condition *AC3*, since *G3* is not observable to the source *G4*, the result is simply equal to $ob_{\{G5\},\{\}}(G3) = 10001010$. Finally, the estimated masking probability on *G3*, $P_{protect}(G3,(G4,1) \rightarrow (G5,0)) \approx 1/8$. This result is consistent with the result achieved by the Mandatory Assignment(MA) method proposed in [48] which finds the equivalent conditions ($c = 1$, $e = 1$, $a = 0$) for the proposed conditions.

3.3.2.2 Evaluation of Zone B

As shown in Figure 5, *Zone B* contains the source and its transitive fan-ins. Due to the extra fan-out path introduced by inserting the redundant wire, the susceptibility of the gates in this zone to transient faults will increase. Thus, this side effect needs to be evaluated in order to assess thoroughly the protective effect. Similar to the procedure to evaluate *Zone A*, we will first analyze the necessary conditions and then propose how to approximate using signature representations.

First, the originally unobservable fault must be able to propagate to the source. Only in this way, after inserting the redundant wire originating from the source, will it be possible for this fault to propagate along the new fan-out path. In addition, this fault must be unovservable originally, because we are calculating the extra faults to which a new redundant wire will lead. Furthermore,

suppose this fault arrives at the input side of the target via this new path; the target gate must hold the non-dominant value so as to let this fault pass through. The reason is that when the target has the non-dominant value, all of its inputs, including the newly added redundant wire, will hold to their non-dominant values, respectively. At this state, any incorrect value at the redundant wire will flip the target, i.e.. the fault will pass through the target. Finally, the last condition is that the target must be observable on primary outputs so that the fault can eventually be observed. In summary, the conditions where a fault can lead to a new error after the insertion of a redundant wire are listed below:

BC1: The fault is observable to the source but is not observable to any of the primary outputs;

BC2: The target holds the non-dominant value;

BC3: The target is observable to at least one primary output.

With the three conditions above, the probability that an unobservable fault at a gate $g \in ZoneB$ will create a new error after converting the implication i into a new redundant wire can be generally represented by Equation 3.6.

$$P_{NewError}(g,i) = P(BC1 \cap BC2 \cap BC3) \tag{3.6}$$

where $P(BC1 \cap BC2 \cap BC3)$ is the probability of the intersection of all three conditions. Again, the condition *BC1* can be approximately evaluated by the signature based expression $ob_{\{S\},\{\}}(g) - ob_{\{alloutputs\},\{\}}(g)$, and the condition *BC2* and *BC3* can be evaluated by the expressions $cc_{\bar{v}}(T)$ and $ob_{\{alloutputs\}\{\}}(T)$, respectively, where $\bar{v}$ stands for the non-dominant value of the target . By replacing each condition with its signature format counterpart, the probability $P_{NewError}(g,i)$ can be evaluated by Equation 3.7.

$$P_{NewError}(g,i) \approx \frac{\left|(ob_{\{S\},\{\}}(g) - ob_{\{alloutputs\},\{\}}(g)) \& cc_{\bar{v}}(T) \& ob_{\{alloutputs\}\{\}}(T)\right|}{K} \tag{3.7}$$

With the above single-gate side effect estimation equation, the overall side effect of a redundant

wire on *Zone B* can be approximated by Equation 3.8.

$$Effect_{ZoneB}(i) \approx \sum_{g \in ZoneB} P_{NewError}(g,i) \times f_g \tag{3.8}$$

Taking the example implication $(G4,1) \rightarrow (G5,0)$ in Figure 3.2, we can see that the implication's *Zone B* includes only one gate, *G4*. Signatures satisfying the three conditions *BC1*, *BC2* and *BC3* are $ob_{\{G4\},\{\}}(G4) - ob_{\{all\,outputs\},\{\}}(G4) = 11111111 - 01010101 = 10101010$, $cc_1(G5) = 10101110$ and $ob_{\{all\,outputs\},\{\}}(G5) = 10101110$. So the estimation of the probability of new errors for this implication is $|10101010\&00001000\&10101110|/8 = 1/8$, which means, given a fixed failure rate on gate *G4*, its probability to convert a fault to an observable error will rise 12.5% due to the redundant wire insertion.

3.3.2.3 Evaluation of Zone C

Zone C does not contain any of the original gates in the circuit. Instead, it contains the extra gates that must be inserted for the new implication. As described in Section 3.2.3, if the implicant value is not the dominant value, the addition procedure will add an inverter into this zone, and likewise, if the implicated value is not the dominant value, the procedure will insert a checker gate. It is possible that these new gates will introduce new faults and this is especially significant in smaller circuits. Hence, the objective of evaluating the effects of this zone is to estimate the effects of errors originating from these newly added gates.

Figure 3.5 supposes the driving gate of the target T is a two-input *AND* gate and lists the four different situations in *Zone C* which are classified by whether the implicant value (S) or the implicated value (T) is the dominant value (*Zero*) or not at the corresponding pins. We will analyze each case separately.

Figure 3.5a shows the case where both the implicant ($S = 0$) and implicated ($S = 0$) values are dominant, and thus, the source S can be directly linked to the input of target T. In this case, there are no extra gates inserted and consequently no extra faults will occur in *Zone C*.

In Figure 3.5b, the implicant value ($S = 1$) does not dominate the target, and during the addition, an inverter is needed between the source and the target to flip the non-dominant value from the

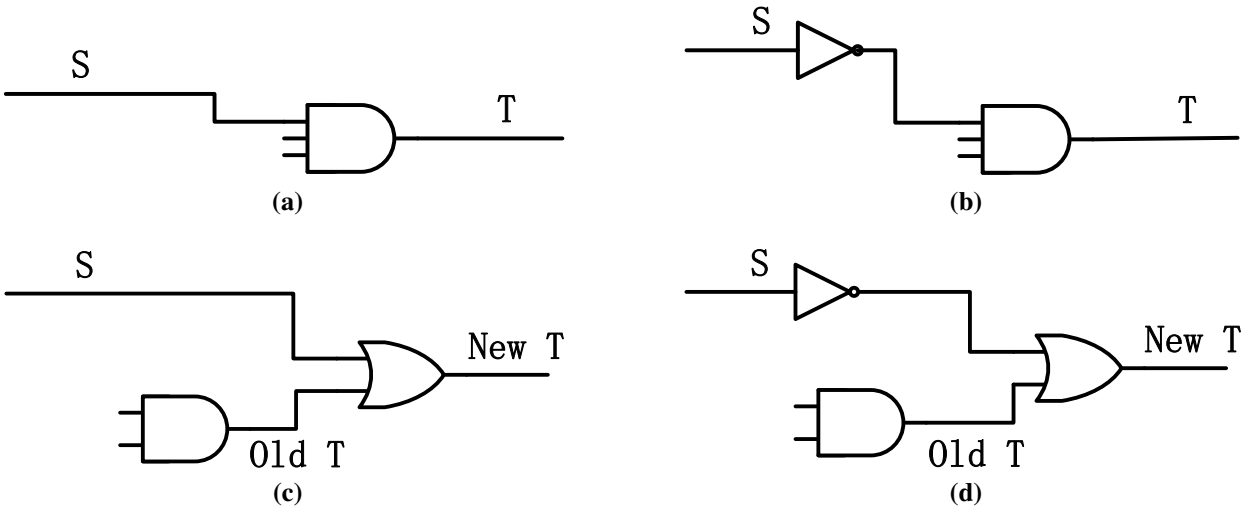


Figure 3.5: Example of new gates in Zone C

source. Since this inverter directly connects to the source, we can treat it as an extension of *Zone B,* and thus, the evaluation methods proposed for *Zone B* are applicable to this inverter as well. Moreover, as it is a newly added gate and is only connected between the source and the target, the first condition *BC1* is always true. Thus, the error probability of this inverter can be represented as shown in Equation 3.9.

$$P_{inverter}(i) = P(BC2 \cap BC3) \tag{3.9}$$

The third type of *Zone C* is when the implicated value ($T = 1$) does not dominate the target. Specifically, in Figure 3.5c, the implicated value does not dominate the target's driven gate (*AND*), and hence, an *OR* gate checker is inserted. In this case, we denote the output of the checker as the new target wire so that if this *Zone C* is treated as a black box, then the part of the circuit outside the box will not notice any changes in terms of wire connection. By comparing the modified circuit with the original one, we find that some of the errors due to the old target (*AND*) will be masked because of the logic masking effect provided by the checker (*OR*), but the newly added checker itself will potentially introduce new errors. Hence, we have to evaluate these two impacts separately.

The calculation of the error probability on the newly added checker is relatively straightforward

since it is equal to the observability of the old target, and thus, can be represented in Equation 3.10.

$$P_{checker}(i) \approx \frac{\left|ob_{\{all\,outputs\},\{\}}(T)\right|}{K} \tag{3.10}$$

On the other hand, the calculation of the masking effects on the original target gate requires us to compare the old and new error probabilities. In the original circuit, the old error probability is equal to the observability of the old target, which can be represented as $ob_{\{all\,outputs\},\{\}}(T)$. As for the new error probability, it is determined by two necessary conditions:

CC1: The source must hold the non-dominant value;

CC2: The target is observable to at least one primary output.

Based on the conditions above, the masking effects on the old target can be evaluated by Equation 3.11.

$$P_{old\,target} \approx \frac{ob_{\{all\,outputs\},\{\}}(T)}{K} - \frac{cc_{\bar{u}}\&ob_{\{all\,outputs\},\{\}}(T)}{K} \tag{3.11}$$

where $\bar{u}$ is the non-dominant value of the source.

The last case in *Zone C*, as shown in Figure 3.5d, is a combination of case 3.5b and case 3.5c, and thus, its effect is the sum of these two cases.

To summarize, there are four cases for *Zone C* based on the implication to be inserted, and for each case we have shown how to evaluate the effect of new faults introduced by the extra gate(s).

3.3.2.4 Evaluation of Overlap Between Implications

The previous sections have introduced to the individual implication evaluation. However, during the implication evaluation and selection procedure, when some implications have been selected and their corresponding redundant wires have been inserted, applying the evaluation equations proposed above may lead to a considerable overestimation of the protective effects. This is because the proposed model is derived based on the single redundant wire insertion while the impact of multiple redundant wires is not considered. Specifically, an error might be masked by multiple

implications, and as long as any of them is selected, the others' protection of this error becomes redundant.

In order to overcome this shortcoming, a new K-bit vector called COV is introduced, which corresponds to the K-bit signature. For an arbitrary gate g in the circuit, $COV(g)$ is defined as a K-bit sequence whose i^{th} bit equals 1 if the corresponding bit in the signature has been masked by the selected implications; otherwise, it equals 0. During the algorithm, the algorithm maintains an instance of this vector for each gate in the circuit. At the start, all the COV vectors are initialized to 0s. Each time an implication is selected, its coverage on each gate is recorded into the corresponding COV vectors. When an unselected implication is evaluated, its intrinsic coverage, as well as the COV vectors, will be considered together to give an overall protective effect evaluation for *Zone A*.

The new probability approximation can be represented as shown in Equation 3.12. By replacing Equation 3.4 with Equation 3.12, Equation 3.5 is now able to give an error coverage evaluation which accounts for the overlapping protection from the previous implications. Of course, the negative effect of Zone B and Zone C must still be considered. Finally, the effect of a given implication i can be expressed as shown in Equation 3.13.

$$P_{protect\,with\,COV}(g,i) \approx \frac{\left|cc_u(s)\&(ob_{\{alloutputs\},\{\}}(g)-ob_{\{alloutputs\},\{T\}}(g))\&(ob_{\{T\},\{\}}(g)-ob_{\{S\},\{\}}(g))\&\overline{COV(g))}\right|}{K} \tag{3.12}$$

$$Effect(i) = Effect_{Zone\,A}(i) - Effect_{Zone\,B}(i) - Effect_{Zone\,C}(i) \tag{3.13}$$

3.3.3 Selection

The selection algorithm is used to select valuable implications and to insert the corresponding redundant wires into the given circuit. Finding the truly optimal combination of redundant wires is computationally infeasible. Therefore, we propose a greedy algorithm which can find a near optimal solution in a reasonable time frame, as shown in Algorithm 5.

Algorithm 5 : Implication Selection

```
function SELECTION(C)
    R ← {}
    COV ← 000···000 ∀i ∈ C
    T ← IDENTIFYIMPLICATIONSHIGHEFFORT(C)
    for i ∈ T do
        POSN ← COMPUTEEFFECT(i)
        INSERTAT(R, POSN, i)
    end for
    while EFFECT(R.first) > threshold do
        INSERTWIRE(R.first)
        UPDATECOV(R)
        REMOVE(R.first)
        LAZYUPDATEEFFECT(R)
    end while
end function
```

The system initially creates an implication rank list R and, for each gate, a coverage vector COV. During the selection process, the rank list R keeps all the implication candidates in descending order so that the algorithm can easily retrieve the next best candidate. Next, the algorithm introduced in Section 3.3.1 is applied to find all the implications within the circuit. These implication candidates are then evaluated with the evaluation model described in Section 3.3.2. After the first round of the evaluation process, all implications are ranked and placed into rank list R.

With the preparation work completed, the system starts the selection procedure which repeatedly selects the best implication candidate from the rank list R and inserts the corresponding redundant wire into the circuit. After each selection round, a re-evaluation process is applied to the remaining implications so as to eliminate the overlapping effect introduced in Subsection 3.3.2.4. This re-evaluation process is performed in a lazy fashion. It starts by re-evaluating the best remaining implication, and if it still ranks as the best candidate, the re-evaluation process stops; otherwise, this implication will be inserted at its proper place in the list R and the re-evaluation process moves on to the next best candidate. In this way, only a small number of implications are re-evaluated after each selection. The selection loop terminates when the evaluated protective effect of the best current candidate is smaller than a threshold set in advance. This threshold can be regarded as an error estimation of the evaluation model. When the evaluation result reaches this threshold, the

benefit is considered insignificant. In our simulations, this threshold is set to an empirical value ranging from 0.2 to 1, depending on the target circuit.

3.3.4 Summary

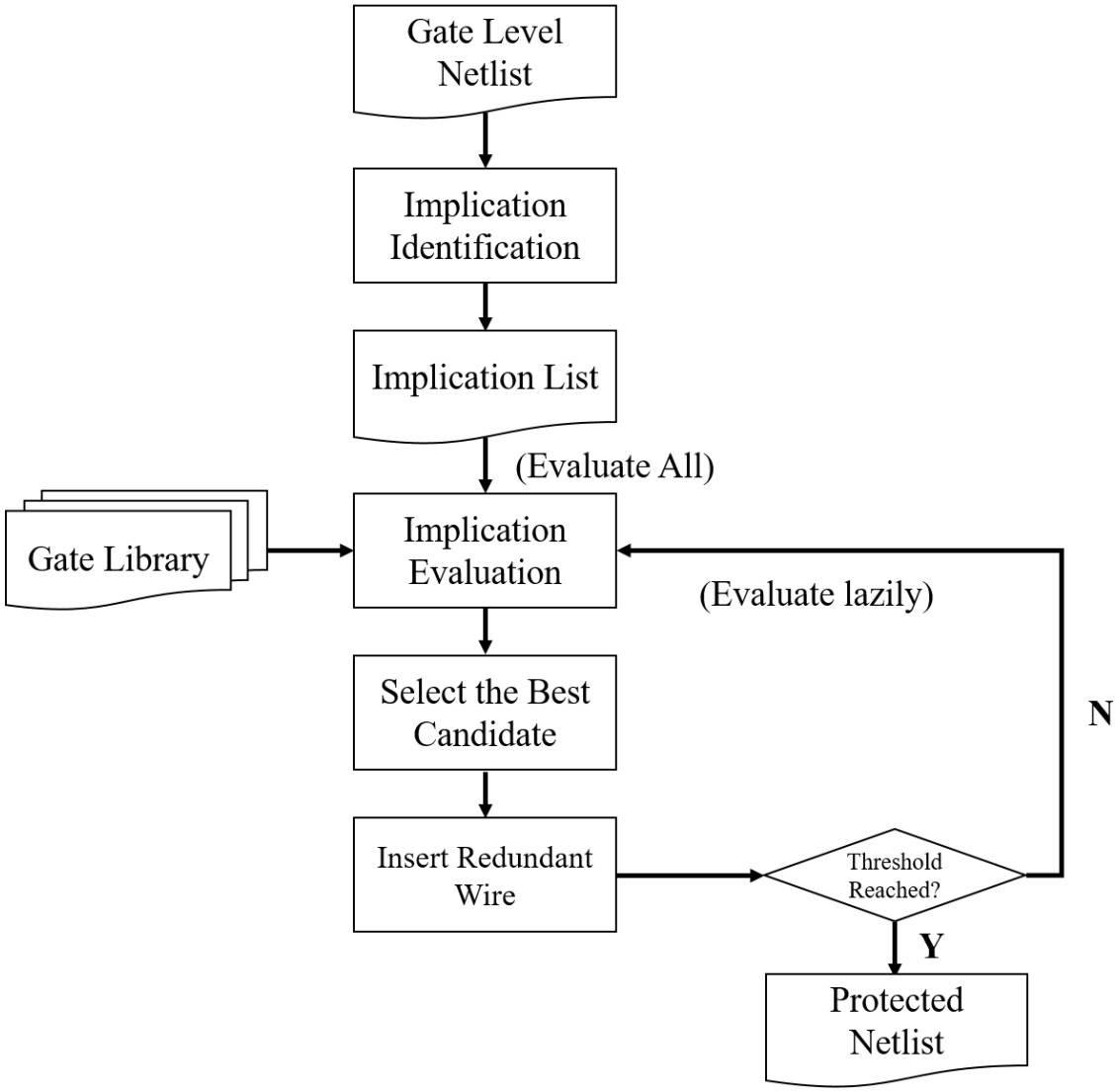


Figure 3.6: High-level Algorithm Flow.

Figure 3.6 shows the overall workflow of the proposed method. The flow starts with a gate level netlist of the target circuit. Then, the implication identification procedure will find and list all the direct and indirect implication candidates by applying the algorithms introduced in Section 3.3.1. With the gate information (such as gate types as well as gate failure rates) from the technol-

ogy library, the implication evaluation procedure estimates the protective effect of each candidate based on the method introduced in Section 3.3.2 and creates a ranking list for the following selection procedure. The selection procedure takes a greedy strategy by always choosing the current best candidate. The chosen candidate will then be inserted into the target netlist as per the redundant wire generation methodology discussed in Section 3.2.3. If the current chosen candidate's estimated effect is higher than a preset empirical threshold, the work flow will loop back and the remaining candidates will be re-evaluated in order to eliminate the influence of present selections. Otherwise, the whole process will terminate and output the protected netlist.

In Section 3.4, the implication identification results are compared with the results from [50], and the evaluation results are compared with the fault injection statistics. Then, the overall protection results are presented, and a concept called Quality of Result (QoR) is introduced in order to illustrate quantitatively the effectiveness of the proposed protection method. Also, the influence of two attributes on the final protective effects are discussed.

3.4 Simulation Results

The algorithm described in Section 3.3 was implemented in C++ and tested on a series of benchmark circuits from the ISCAS'89 and LGSynth93 suites. The selected circuits were synthesized into the 45nm Nangate Library [65] using Synopsys DC®. The failure rate of the original and the protected circuits was evaluated by fault injection simulation with random input vectors. In all cases, one million fault injections were performed, with the faults randomly distributed over all the gates in the circuit, including the newly added gates in the protected circuits. The area, power and delay results were estimated by the Synopsys DC®. All the results are presented based on 512 bits signatures.

3.4.1 Implication Identification

The first step of the protection process is identifying the implications. In Table 3.3, the number of implications identified by both the low-effort (Algorithm 3) and the high-effort (Algorithm 4)

algorithms are listed and compared with the number achieved by the indirect method presented in [50]. The data shows that the proposed methods can identify 157.9% and 163.7% more implications than the indirect method while only taking 2.7% and 3.2% of [50]'s execution time on average, respectively.

By comparing the results of Algorithm 3 and 4, we can see the implications identified by the high-effort algorithm always outnumber the ones identified by the low-effort. This is consistent with the design purpose of the two algorithms. In terms of the time consumption, the high-effort algorithm in most cases takes longer than the low-effort and the greater the difference, the more extra implications the high-effort algorithm can identified. The exceptions may be caused by the instability of the work machine's runtime status, such as CPU temperature.

All the subsequent results were based on the implications generated by the high-effort algorithm.

Table 3.3: Results of Identification Algorithm

Circuit	Indirect [19]		Indirect Low Effort		Indirect High Effort	
	#	Time(s)	#	Time(s)	#	Time(s)
s298	1971	403	2326	3	2378	3
s344	2378	427	3941	9	3965	9
s641	8971	931	11629	121	11666	124
s386	3762	137	7375	11	7897	10
s444	2450	933	3710	10	3859	11
s713	9007	1125	12828	158	12962	226
s526	3203	4223	4694	9	4915	10
s510	9085	4354	19046	19	19488	20
s820	24763	67412	24424	50	25119	69
s832	27856	88849	24250	67	24985	85
s953	32015	21020	68099	114	71458	138
s1196	27445	234116	52199	148	55217	165
s1238	27127	319916	52514	139	55935	158
s1423	12654	42380	18547	421	18755	418
s1488	54922	19804	96112	295	99978	352
s1494	58197	34507	95278	378	99217	379

3.4.2 Protective Effect Evaluation

Besides identifying adequate implication candidates, another key factor of the redundant-wire-based SET mitigation method is to evaluate the protective effect of a set of redundant wires (Section 3.3.2). To validate the accuracy of the evaluation algorithm, we compared the simulation results obtained by fault injection simulation to the estimated reduction of failure rate.

In the simulation, a stuck-at fault model is adopted to evaluate the susceptibility of each circuit. This model assumes that all the erroneous signals caused by radiation effect are long and strong enough to survive both electrical masking and temporal masking. Thus, it allows the simulation to focus on logical masking only. Each node of a circuit has its intrinsic failure rate, and in general the technology adopted as well as the cell type are the determinants of its value. Here, we designate this per-gate failure rate as f_i, and the propagation rate of a fault at a given gate i as $EPP_i(V)$, where V is the given set of input vectors. Given the fault model above, the failure rate of a whole circuit can be represented by Equation 3.14.

$$FailureRate_{cct} = \sum_{i \in Gates} f_i \times EPP_i(V) \tag{3.14}$$

The failure rate reduction estimated by the evaluation algorithm and its corresponding simulation results are summarized in Table 3.4. From left to right, the table lists the name of the circuit, the number of gates, the number of added redundant wires, the estimated failure rate reduction, the simulation results and the percentage error between the two. The simulation data are calculated by Equation 3.14. Here, we assume f_i is a constant equal to 1. However, in a practical application we can assign f_i to a technology specific value. The estimated failure rate reduction is an accumulation of the evaluated protective effect of added redundant wires.

The percentages of evaluating errors are less than 1% in more than half of the benchmarks (13 of 23) and below 5% in almost all the benchmark set. The absolute average percentage of errors is only 1.93%. However, we cannot find a clear trend toward the evaluating errors from the data. A variety of factors such as benchmark scales, circuit structures and numbers of added redundant wires contribute together to the algorithm accuracy.

Table 3.4: Results of Evaluation Algorithm

Circuit	# Gates	Redundant Wires	Estimated Reduction	Simulated Reduction	% Error
s298	119	4	59.54	59.39	0.25
s344	160	10	79.05	79.4	-0.44
s641	379	15	205.89	206.07	-0.09
s386	159	10	37.04	36.75	0.78
s444	181	10	73.61	73.94	-0.45
s713	393	15	206.35	211.1	-2.25
s526	193	11	81.07	80.31	0.95
s510	211	23	83.44	81.95	1.82
s820	289	17	55.55	56.21	-1.17
s832	287	15	56.46	56.48	-0.04
s953	395	51	135.61	120.02	12.99
s1196	529	37	124.72	126.57	-1.46
s1238	508	29	126.03	125.68	0.28
s1423	657	53	273.95	274.52	-0.21
s1488	653	41	131.1	137.51	-4.66
s1494	647	69	122.25	126.8	-3.59
alu2	244	29	106.8	112.07	-4.7
ex5p	239	20	115.5	116.17	-0.58
f5lm	102	13	43.45	44.81	-3.04
inc	88	20	39.38	37.98	3.69
sao2	109	20	16.86	16.93	-0.41
alu4	653	40	44.31	44.55	-0.54
cmb	14	1	4.15	4.15	0
				ABS Average:	1.93

3.4.3 Protection Results

The selecting algorithm incrementally picks the implication candidate and inserts it into the target netlist. This selecting procedure will generate a series of netlists, each of which receives an increasing fault-masking level, as well as additional area and power overheads. By tracking the status of these intermediate netlists, we can have a different view of the algorithm. Figures 3.7 and 3.8 illustrate the changes of overheads versus the percentages of failure rate reduction in two benchmarks. In S713, the proposed SET mitigation method achieves a 17.8% failure rate reduction with a 2.4% area overhead and a 3.2% power overhead. In S1488, the method achieves a 21.3% failure rate reduction at the cost of an extra 3.2% area and 3% power overhead.

The rising trend lines in both figures indicate the circuit overheads increase with the addition of redundant wires. In addition, with more redundant wires added, the slope of the trend lines becomes steeper, which means that in order to achieve the same reduction percentage, more overheads are incurred toward the end than at the start. This observation proves that the method adds redundant wires in descending order by their protective effects. We can also see that the power overhead always increases with the growth of the area overhead, while the increase of timing overhead does not share the same pattern. The addition of redundant wires may not affect the total circuit delay as long as the number of circuit layers is not increased after addition. Thus, the resulting timing overhead is highly dependent on the structure of the target circuit.

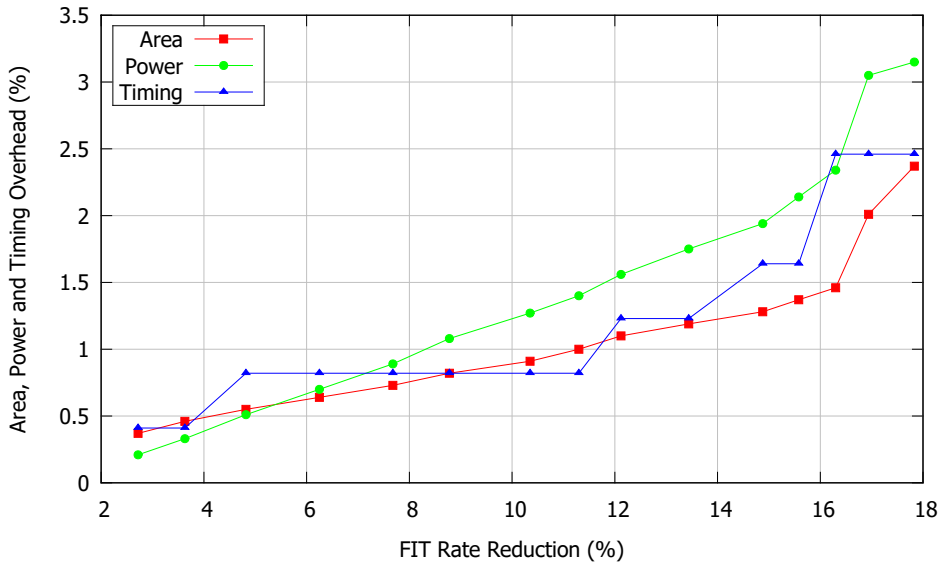


Figure 3.7: Failure rate reduction for benchmark S713.

The results when applying the proposed SET mitigation method are summarized in Table 3.5. We selected 16 circuits from the ISCAS'89 benchmarks and compare at the results to those published by [50]. The first five columns display the basic information about each benchmark. The sixth column is the percentage of error rate reduction achieved when the selecting algorithm terminates. This value is the ratio of the reduced error rate to the original. Columns 7-9 list the corresponding overheads in area, power and delay, respectively, and the tenth column is the number of added redundant wires. The eleventh column introduces a new concept called the Quality of Result (QoR). This metric is the ratio of the error rate reduction to the area overhead, which gives

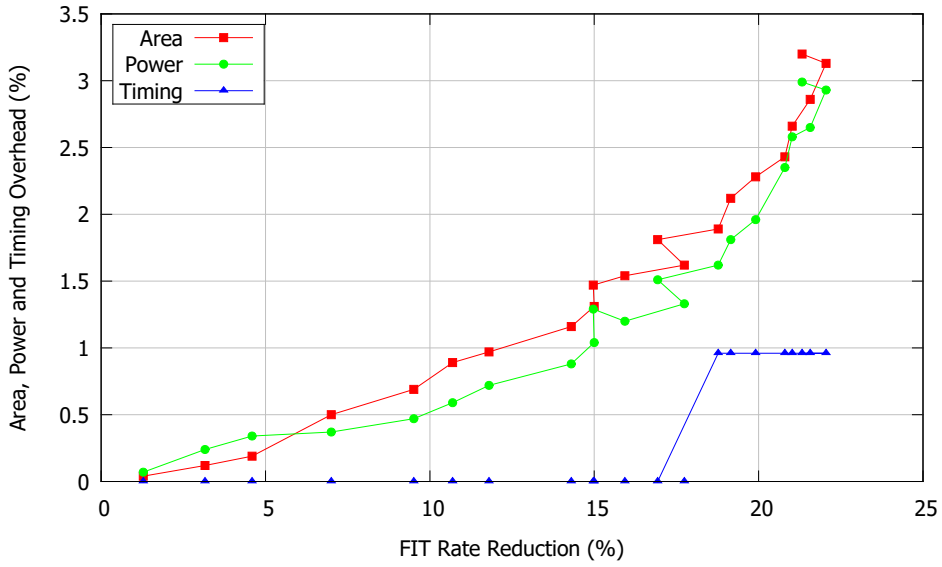


Figure 3.8: Failure rate reduction for benchmark S1488.

us a simple way to measure the effectiveness of a mitigation method at a unit area cost. The three right columns are the corresponding results published by or calculated based on [50].

From the data in Table 3.5, we see the reduced percentages of error rate range from 4.5% to as high as 49.7%. The average percentage is 18.4%. The average area, power and delay overheads are 4.3%, 4.4% and 5.4%, respectively. For each circuit, its power overhead is close to its area overhead, while its delay overhead varies dramatically, which is consistent with the aforementioned analysis. The average QoR is 4.9, i.e., 4.9% error rate reduction per 1% area overhead. By comparing to [50]'s results, we can see that the proposed method can achieve a better protective effect while causing less than half of the overhead. The average QoR is 2.9 times higher than [50]'s result.

3.4.4 Impact of Synthesis Constraints

It is known that timing constraints have a significant impact on the structure of a synthesized netlist. Usually, a tighter constraint on timing will lead to less gate and logic sharing, and less circuit depth. In this section, we explore the influence of timing constraints on the proposed method by applying the method on two netlists synthesized with different constraints. In Table 3.6, six LGSynth93 benchmarks are selected, and each benchmark is synthesized separately with both

Table 3.5: Protection Results

Name	# PI	# PO	# Gates	# Inv	% Error Reduction	Area %	Power %	Delay %	# Implications Added	QoR	% Error Reduc. [50]	Area % [50]	QoR [50]
s298	17	20	75	44	4.5	1	0.8	0	4	4.6	5.3	2.1	2.6
s344	24	26	101	59	6.5	2.7	2.9	10.9	10	2.4	2.2	2.2	1
s641	54	43	107	272	19.6	1.5	2.6	2.7	15	13.3	6.1	2.8	2.2
s386	13	13	118	41	36.1	5	1.9	5.9	10	7.3	5.5	2.9	1.9
s444	24	27	119	62	14.1	3.4	4.9	0	10	4.2	13	10.8	1.2
s713	54	42	139	254	17.8	2.4	3.2	2.5	15	7.5	8.5	4.7	1.8
s526	24	27	141	52	8.6	3.2	4.4	0	11	2.7	5	3.4	1.5
s510	25	13	179	32	23.7	8	10.1	22.7	23	2.9	23.8	17.9	1.3
s820	23	24	256	33	11.4	2.9	2.3	0	17	4	17.4	7.3	2.4
s832	23	24	262	25	10.4	1.8	1.5	0	15	5.7	17.1	7.2	2.4
s953	45	52	311	84	49.7	11.4	8.9	20.6	51	4.4	42	17.5	2.4
s1196	32	32	388	141	20	6.4	7.7	9.2	37	3.1	23.7	15.1	1.6
s1238	32	32	428	80	15.9	7.1	9.3	8	31	2.3	38	21.4	1.8
s1423	91	79	490	167	6.4	2.9	1.5	0.7	53	2.2	15	16.1	0.9
s1488	14	25	550	103	21.3	3.2	3	1	41	6.7	9.8	7	1.4
s1494	14	25	558	89	28.3	5.5	5.2	1.9	69	5.1	18.5	17.5	1.1
			Average		18.4	4.3	4.4	5.4	25.8	4.9	15.7	9.7	1.7

a lower and a higher timing constraints. The error reduction percentage and the corresponding overheads were collected after applying the proposed method on both generated netlists.

The results indicate that the proposed method can achieve higher error reduction rates while causing less area and power overheads on the tighter constrained netlists. A possible explanation of this result is that the synthesis with a tighter constraint will generate a netlist with less gate and logic sharing, which means the gates in the generated netlist will have fewer fan-outs. Based on the evaluation analysis in Section 3.3.2, a gate with fewer fan-outs will be better protected by redundant wires. However, the delay overheads increase extremely rapidly due to the tightly constrained netlists' high sensitivity towards timing.

3.4.5 Impact of Signature Length

The signature technique provides a simple but effective way to evaluate the protective effect a set of redundant wires can provide, and the evaluating results directly guide the selecting process of the proposed method. In order to explore the influence of signature lengths on performance, we selected 16 benchmarks and re-executed the redundant wire method on these benchmarks but with different signature lengths (128-bit and 32 bit). The collected error reduction and area overhead results were compared to the benchmark results (512-bit signatures) and are represented as a per-

Table 3.6: Impact of Synthesis Constraints

Circuit	# Gates	Extra Delay (ns)	% Error Reduction	% Area	% Power	% Delay	# Wires
alu2	244	0.86	11.50	9.50	8.80	8.10	40
	415	0.51	16.10	4.40	4.80	23.50	40
ex5p	239	0.56	4.70	4.00	4.00	3.60	20
	403	0.32	9.60	2.20	2.50	15.60	20
f51m	102	0.5	6.70	5.30	7.40	4.00	13
	167	0.35	10.60	4.20	5.50	17.10	15
inc	88	0.45	19.50	11.40	14.30	55.50	20
	144	0.3	23.70	4.90	5.20	39.00	18
sao2	109	0.55	19.10	13.30	11.70	9.10	20
	170	0.34	51.80	3.40	4.30	32.40	13
alu4	653	1.12	11.00	3.10	3.50	0.90	29
	1220	0.52	27.10	2.40	2.50	65.40	41

centage of the benchmark results. Thus, a number less than 100% on the error reduction means the protective result is not as good as the benchmark result, and a number more than 100% on the area overhead means the resulting overhead is higher than the benchmark result.

Table 3.7 lists the 128-bit and 32-bit results as a percentage of the benchmark results (512-bit). From the data, we can see that in most cases, the results obtained from shorter signatures are worse than the ones from longer signatures. On average, the 128-bit can achieve a 94% protective effect of the benchmark results and the 32-bit can achieve 88% by causing equal area overhead. However, since the selecting algorithm adopts a greedy strategy, there are cases where a sub-optimal choice in the earlier selecting decisions will lead to a better final solution. Therefore, in a few cases, the shorter signatures can produce a better result.

3.5 Conclusions

We presented a redundant-wire-based SET mitigation approach. The approach utilizes the signature-based evaluating equations to guide the selection. Simulation results indicate this new selection method is more efficient than those of previous works. The simulation results also show that the proposed approach can achieve high coverage with a low cost, which is very suitable to

Table 3.7: Impact of Vector Size on Quality of Results

Circuit	128-bit		32-bit	
	% Error Reduction	% Area Overhead	% Error Reduction	% Area Overhead
s298	92.60	100.00	99.16	100.00
s344	87.73	146.15	85.51	76.92
s641	97.18	100.00	100.12	120.00
s386	91.41	90.32	76.52	100.00
s444	91.35	100.00	99.93	100.00
s713	100.70	88.46	99.70	146.15
s526	88.10	100.00	56.83	57.14
s510	100.29	103.57	81.68	105.36
s820	92.56	88.24	88.84	97.06
s832	90.84	131.82	81.03	100.00
s953	99.52	104.32	98.99	112.95
s1196	100.59	95.54	87.28	79.46
s1238	100.10	94.79	96.30	98.96
s1423	88.06	100.00	85.87	106.35
s1488	95.60	79.52	83.51	90.36
s1494	87.88	90.91	87.05	102.10
Average	94.03	100.85	88.02	99.55

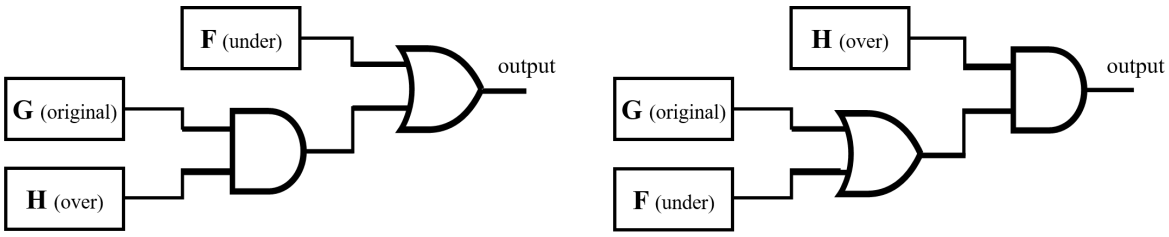
ground-level applications.

CHAPTER 4

NEW APPROXIMATE LOGIC BASED APPROACHES FOR SYNTHESIS OF REDUNDANT COMBINATORIAL LOGIC FOR SELECTIVE FAULT TOLERANCE

Chapter 3 presented a highly efficient SET error mitigation method based on redundant wire technique. However, the number of good redundant wire candidates is limited in some circuits, which constrains the maximum coverage the technique can achieve (e.g. average 18.4%). In this Chapter, a new approximate logic based technique is presented. This technique will provide designers with greater flexibility and its best effort case, which is also known as TMR, can reach a 100% coverage.

The proposed algorithm builds on concepts that were identified in [49, 29]. The basic idea is that any combinatorial logic function, G, can be augmented with a simpler approximation of the min-terms (F) and the max-terms (H), as shown in Figure 4.1. For any input vector for which F is true, G is also true. As a result, if there is a fault causing G to incorrectly be false, then it may be corrected by F. In a similar way, H may fix faults where G is incorrectly true.



(a) Form 1 **(b)** Form 2

Figure 4.1: Approximate Logic Functions

Potentially F or H can be empty (e.g., $H = 1$, $F = 0$). The maximum added delay is two gates, and if either F or H is empty, the delay is a single gate.

4.1 Fault Model and Evaluation Method

The failure rates of gates vary significantly. Clearly, an inverter consisting of two transistors is less prone to SET faults than a complex gate with 10 transistors, like an XOR, as seen in Figure 4.2. The sensitivity to radiation-induced SETs can vary by nearly an order of magnitude, as seen from the simulation results for six cells taken from a 28nm commercial cell library [66] shown in Figure 4.3. For these reasons, we need to consider the actual technology failure rates when synthesizing redundant logic functions. We assume that the per-gate FIT rate, f_i, has been characterized in advance.

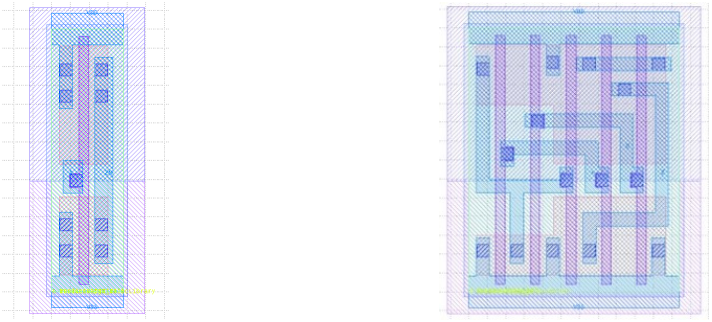


Figure 4.2: Layout of INV and XOR Gate

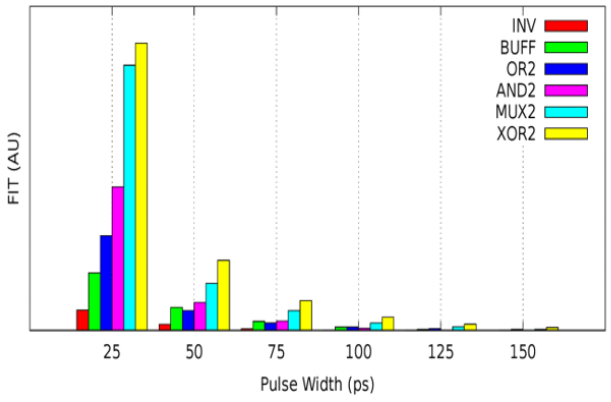


Figure 4.3: Transient Sensitivity of 28nm Gates.

Due to the logical masking effect, a fault in a given gate has a probability of propagation to the

outputs that we designate $EPP_i(V)$, where V is a set of selected input vectors, and f_i is the FIT rate for the given gate. The FIT rate of the full circuit is given by Equation 4.1:

$$FIT_{cct} = \sum_{i \in Gates} f_i \times EPP_i(V) \tag{4.1}$$

The goal is to reduce the circuit-level failure rate using redundant logic, while minimizing the area and power overhead. We evaluate FIT_{cct}, using Equation 4.1, both for the original circuit and then for the protected circuit with the redundant logic. When evaluating the protected circuit, the failure rate of the gates in the redundant logic is included. Thus, to yield a net improvement in reliability, the fault masking of the redundant gates must more than offset the increased intrinsic failure rate due to a higher gate count.

4.2 Proposed Algorithm

Many problems in logic synthesis are NP-hard [67], and thus, heuristics must be used to find solutions in a reasonable time. The starting point for the proposed algorithm is a minimal two-level representation of the original logic function, as produced by the Espresso logic minimizer [68]. Consider an example logic function with two outputs, $G0$ and $G1$, whose Karnaugh map is shown in Figure 4.4. After logic minimization, the list of cubes is shown in Table 4.1.

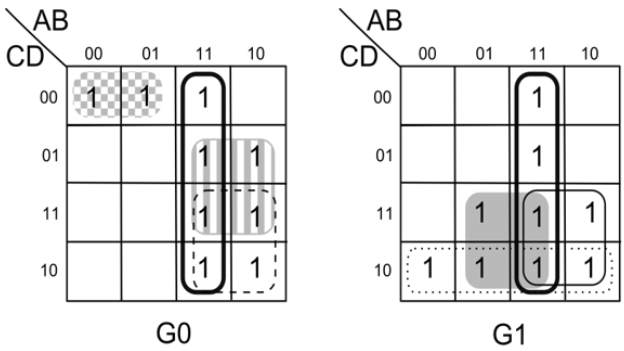


Figure 4.4: Karnaugh Map Examples

The algorithm selects cubes from this list to add to the F or H approximate functions, as shown in Figure 4.5. A fault reduction (FR) metric is computed for each cube using Equation 4.2. The

Table 4.1: Logic Cubes for Example Circuit

TERM	TYPE	COVERAGE
G0		
0 - - 1	max	4
0 - 1 -	max	4
1 0 0 0	max	1
1 1 - -	min	4
1 - - 1	min	4
1 - 1 -	min	4
0 - 0 0	min	2
G1		
0 - 0 -	max	4
- 0 0 -	max	4
0 0 - 1	max	2
- 1 1 -	min	4
- - 1 0	min	4
1 1 - -	min	4
1 - 1 -	min	4

cover of the cube is trivial to obtain from the two-level representation and measures the number of input vectors that will be protected by the cube, e.g., the number of covered vectors of 0--1 is 4 (0001, 0011, 0101 and 0111). The EPP (error propagation probability) is the likelihood that errors in the gates actually affect one of the outputs, for the vectors covered by the cube. This probability is estimated through fault-injection simulations.

$$FR(cube) = (\#min/max-terms) \times \widehat{EPP}_l \tag{4.2}$$

The cubes are ranked and the best cube is selected for addition to the approximate function (F or H). After selecting one cube, the FR metric for the remaining cubes is recalculated in a lazy fashion described below. The cubes are then re-sorted to identify the next best candidate.

The error propagation probability (EPP) is calculated by statistical fault injection on the gate-level circuit using a digital simulator. Random input vectors, within the scope of the cube, are generated based on the distribution of vectors in V (a set of vectors taken from an input trace). For each fault injection, one gate is randomly chosen, weighted by the per-gate FIT rates, f_i. A SA (0 or 1) or SET is injected on the selected gate and it is determined if the fault propagates to a

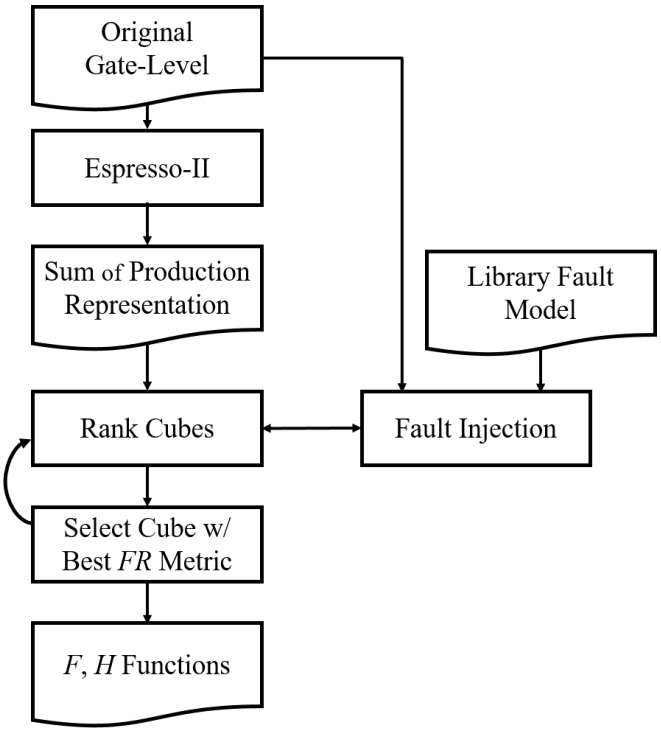


Figure 4.5: High-Level Algorithm Flow.

primary output. This biased fault injection approach was chosen in order to take into account the actual expected failure rates of the gates in the mapped circuit. This ensures that the redundant logic functions are optimized to cover the faults that are most likely to occur.

The objective is to perform the minimum number of fault injections necessary in order to identify the next best term to select. To avoid running more simulations than necessary, the simulations are launched in small batches of size N_{batch}. After the first batch of simulations, the value of EPP is estimated ($\widehat{EPP}$) based on the fraction of simulations where the fault propagates to an output. If we assume that the distribution of $\widehat{EPP}$ is normal, then the confidence interval is given by Equation 4.3 [24, p130], which consists of the EPP estimate and the error bar. For 95%, 90% and 80% confidence intervals, $z_{\alpha/2}$ is 1.96, 1.65 and 1.28, respectively.

$$\widehat{EPP}_n \pm z_{\alpha/2}\sqrt{\frac{\widehat{EPP}_n(1-\widehat{EPP}_n)}{n}} \tag{4.3}$$

Using the number of cubes and the estimated EPP, the fault reduction (FR) is computed for each cube and the cubes are ranked. Due to the error bars, at the top of the list, there may be

multiple overlapping best candidates. In this case, additional fault injections are performed in groups of N_{step} in order to further reduce the error bars. Additional simulations are only run for those cubes that are potentially the best candidate. A maximum of N_{max} simulations is performed, and if there are still multiple candidates, the term with the best FR is simply selected, disregarding the other candidates whose FR estimate may overlap. In these cases, there is little difference between the best candidates.

Table 4.2: Results after First Round

INPUTS A B C D	OUTPUT	TYPE	COVERAGE	# FI	FR	ERR BAR
0 - - 1	**G0**	**Max**	**4**	**20**	**2.6**	**0.546**
- 1 1 -	**G1**	**Min**	**4**	**20**	**1.8**	**0.570**
- - 1 0	**G1**	**Min**	**4**	**20**	**1.8**	**0.570**
1 1 - -	**G0**	**Min**	**4**	**20**	**1.8**	**0.570**
1 - - 1	G0	Min	4	20	1.4	0.546
0 - 1 -	G0	Min	4	20	1.4	0.546
1 1 - -	G1	Min	4	20	1.2	0.526
1 - 1 -	G0	Min	4	20	1.2	0.526
1 - 1 -	G1	Min	4	20	1.0	0.496
0 - 0 0	G0	Min	2	20	0.8	0.280
0 0 - 1	G1	Max	2	20	0.7	0.273
1 0 0 0	G0	Max	1	20	0.7	0.131
- 0 0 -	G1	Max	4	20	0.6	0.409
0 - 0 -	G1	Max	4	20	0.4	0.344

Consider the example logic function shown in Figure 4.4. After the first round of the algorithm, with 20 fault injections, the results are shown in Table 4.2. The term 0--1 has the highest FR metric, however, there are three other terms (-11-, --10, 11--) with FR metrics that could overlap due to the error bars. The remaining ten terms (not bold) can be eliminated as their FR metric is too low, even considering the error bars, which are calculated using Equation 4.3.

After additional fault injections, the term 0--1 is selected as the first term to add to F. Before proceeding to the next round, the table of min/max-terms is updated to reflect the overlap between 0--1 and other terms. For example, the uncovered terms for 0-1- is reduced to 2 (0011 and 0111).

For the next round, new fault injections must be performed to reflect the coverage provided by the redundant logic that has been added, but this is only necessary for terms that overlap with the

newly added, redundant term. Furthermore, the new FR can only be lower than the FR metric from the previous round, and thus the previous FR estimate serves as a valid upper bound. Additional fault injections are only performed for those terms which are potentially the next best-candidate in the following round. In this way, a sequence of F, H functions are generated with each one providing increasingly better coverage of G, taking into account the actual failure rates of the library gates over a set of actual input vectors.

4.3 Simulation Results

Selected LGsynth93 and ISCAS benchmark circuits were synthesized into the 45nm Nangate Library [65] using Synopsys DC™. For both the original and the protected circuits, the FIT rate was evaluated using Equation 4.1. For the purpose of evaluating the circuit-level FIT rate, 10,000 fault injections were performed, to ensure small error bars. For the SA fault model, an arbitrary FIT rate (f_i), was assigned to each gate, proportional to the number of transistors in the gate. For the SET fault model, actual SET FIT rates and pulse widths for the NanGate library were used [69].

We present the results as the FIT rate reduction which is the ratio of the protected FIT rate to that of the original circuit. Note that this metric reflects the fact that failures can occur in the redundant logic, and thus, represents a net FIT rate reduction. If the original FIT rate was 10 and the reduced FIT rate is 5, the reduction is 2x.

The algorithm described in Section 4.2 was implemented using a Ruby script. The parameter N_{batch} was set between 20 and 100 based on the size of the circuit. N_{max} was set to 400 and N_{step} was set to 10. An 80% confidence interval was used ($z_{\alpha/2} = 1.28$) when evaluating the error bars. The cubes were generated with Espresso and the fault injection simulations were performed with Mentor's Modelsim™. The redundant logic functions (F, H) produced by the script were synthesized with Synopsys DC™.

4.3.1 Simulation Results with SA Fault Model

Figures 4.6a, 4.6b and 4.6c show the area and power overhead required to achieve increasing levels of SA fault masking for three circuits. Each point was produced after five additional cubes were added to the redundant logic. The connecting lines show the order of cube selection. In some cases, adding additional cubes reduces the area and power, presumably because the extra cubes enable new logic simplification.

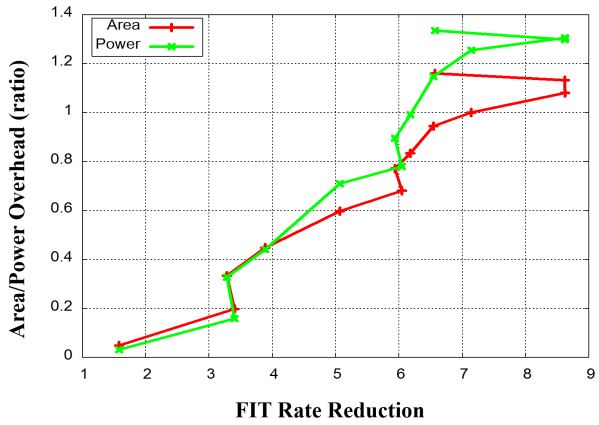
In Figure 4.6a, we see that initially a small overhead is required to achieve a significant reduction in the FIT rate. In Figures 4.6b and 4.6c, comparative data points from previous works [47, 70] are shown. Note that the referenced works assumed a uniform, SA fault model.

4.3.2 Simulation Results with SET Fault Model

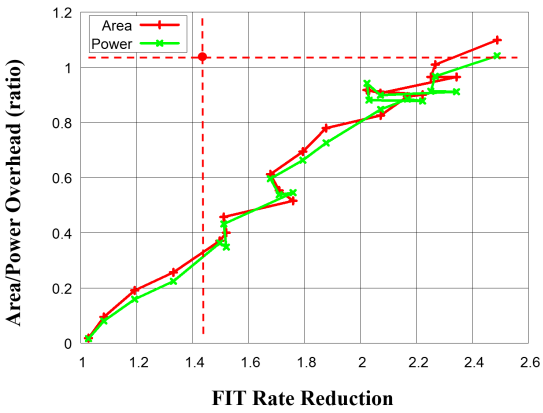
Figures 4.7a and 4.7b show the area and power overhead required to achieve increasing levels of SET fault masking for s1488 and sao2. Since the absolute SET error rates are much lower compared to SA fault rates, the relative uncertainties in the FR metric from the fault injection are larger. More simulations are required to correctly rank the cubes, and the N_{max} limit is often reached. As a result, the cube selection is not always optimal and the resulting curves are not consistently monotonic. However, the algorithm does generate a sequence of redundant logic functions which significantly reduce the SET sensitivity. Table 4.3 summarizes the area, power and timing overhead for nine circuits, showing the first generated result with at least a 2x reduction in FIT rate, both for the SA and SET fault models.

4.3.3 Comparing Results between Fault Models

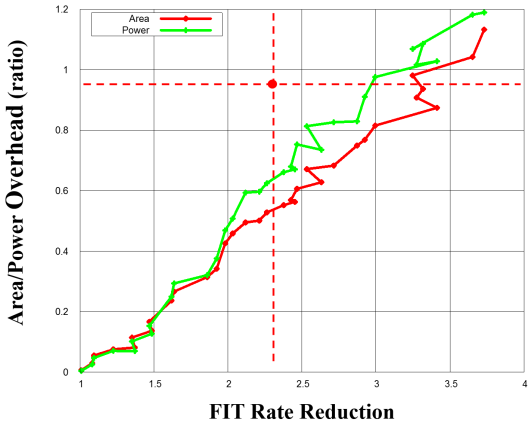
Because the absolute error rates are smaller with the SET fault mode, the cube ranking is more difficult and more fault injections are required to identify the best candidate. We thus evaluated the ability of the approximate logic generated using the SA fault model to mask SETs. Figure 4.8 shows the SET masking ability of redundant logic functions created by the algorithm using SA fault model and an evaluation with SET faults for two circuits.



(a) sao2



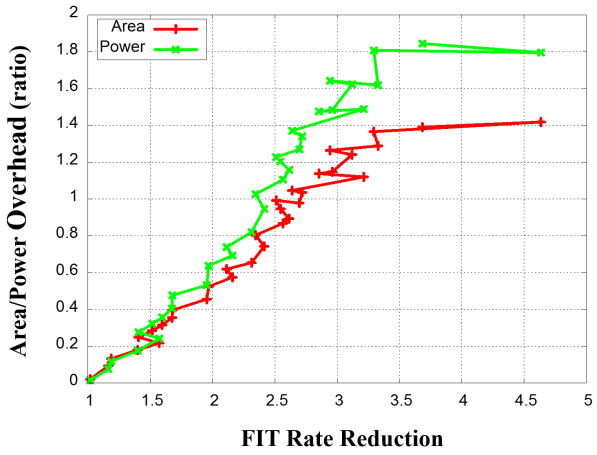
(b) alu2 and Comparison with [47]



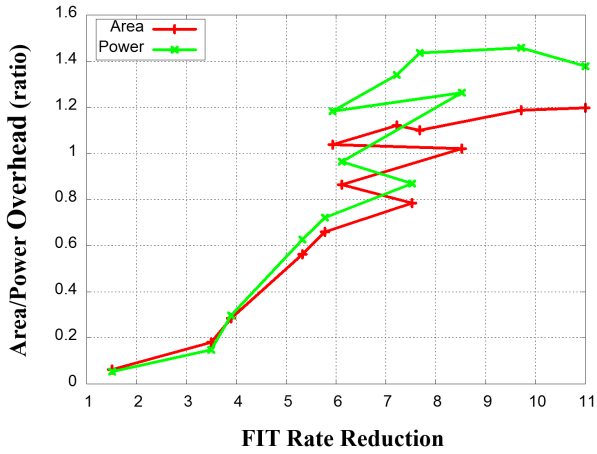
(c) alu4 and Comparison with [70]

Figure 4.6: Area, Power Overheads vs FIT Reduction Results with SA Fault Model (SA)

The generated functions are actually very effective at masking SETs and this trend was observed for other circuits. In order to better understand why this result was observed, for the s386 and alu4 circuit, the sensitivity of each gate was assessed using an SA and an SET fault models. The gates were then ranked from the least to most sensitive. Figure 4.9 shows a scatter plot where the horizontal axis is the gate's sensitivity rank under the SET model, and the vertical axis, the gate's sensitivity rank under the SA model. The correlation is very strong. Based on these results, it appears that redundant logic functions generated using an SA fault model also perform well at masking SETs and they are easier to synthesize.



(a) s1488



(b) sao2

Figure 4.7: Area, Power Overheads vs FIT Reduction Results with SET Fault Model.

4.3.4 Effect of Logic Sharing on Fault Tolerance

During synthesis, we allowed the synthesis tool to share logic between the F and H functions across multiple outputs, but it was ensured there was no logic sharing with the original function G. We note that this approach is different from [47] where sharing between F and H was explicitly prevented during synthesis. When logic sharing is prevented, faults in one of the approximate functions are fully blocked because the original function and the other approximation are correct (see Figure 4.1). However, this benefit comes with the downside that the area of the shared logic

Table 4.3: Summary of SA and SET Results for Benchmark Circuits ($N_{max} = 400$)

Cct	PI/PO	# Gates	# Terms	FIT Reduction	SA			SET		
					Area (%)	Power (%)	Delay (%)	Area (%)	Power (%)	Delay (%)
sao2	10/4	70	167	2x	7	6	20	7	6	20
alu4	14/8	469	2066	2x	32	35	40	37	40	40
5xp1	7/10	65	153	2x	107	116	70	104	99	70
ex5p	8/63	188	652	2x	55	26	30	63	42	40
f51m	8/8	73	155	2x	101	89	50	84	77	50
inc	7/9	69	142	2x	61	71	50	63	75	40
s386	13/13	64	163	2x	24	22	30	16	11	30
s1488	14/25	297	1639	2x	45	59	30	40	47	30
s1494	14/25	298	1636	2x	52	66	40	42	48	40

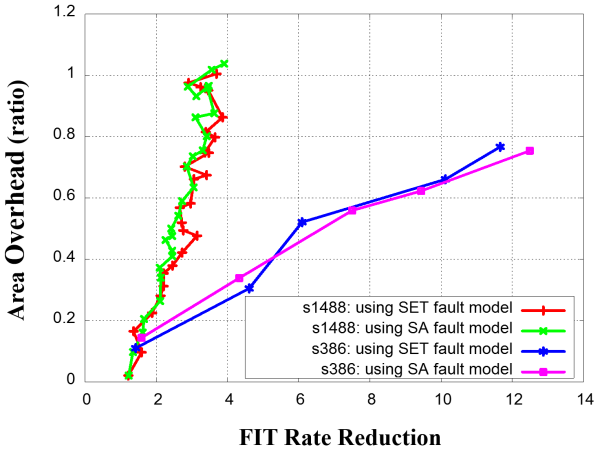


Figure 4.8: SET Masking Effect of Redundant Logic Generated Using SA Fault Model (s386 and s1488).

is increased.

To investigate which approach provides better overall results, the redundant logic functions were synthesized for two circuits (sao2, inc) with and without logic sharing. The resulting circuits were evaluated with an SA fault model and the results are shown in Figure 4.10.

When a small level of fault masking is required, both approaches have very similar performance. For higher levels of fault masking, however, it appears better to prevent logic sharing between F and H.

When the number of cubes is small, the potential for logic sharing is limited. However, as the

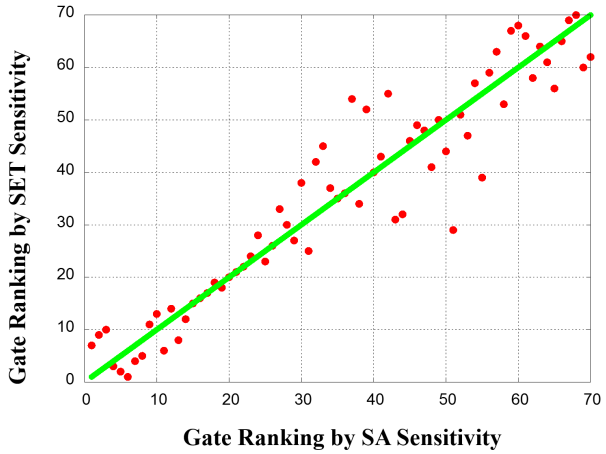


Figure 4.9: Per Gate - Fault Model Correlation (sao2).

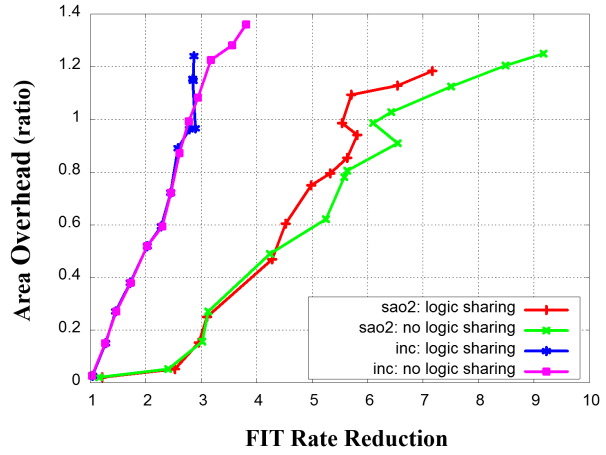


Figure 4.10: Effect of Logic Sharing (F, H) for sao2 and inc.

number of cubes increases, there is a greater chance that a single fault impacts a shared term in both approximate functions. It thus appears better to prevent logic sharing between F and H.

4.3.5 Results Prediction

When we observe the performance of the algorithm on different circuits, it is apparent that it is very effective in some cases (e.g., sao2) and less effective for others (e.g., inc). To identify for which circuits the proposed algorithm is able to best generate approximate logic, we analyzed the distribution of logic cubes of different size, as shown in Figure 4.11. Not surprisingly, the algorithm performs well for circuits like sao2 and s1488 with large logic cubes which can provide

high coverage at lower implementation cost.

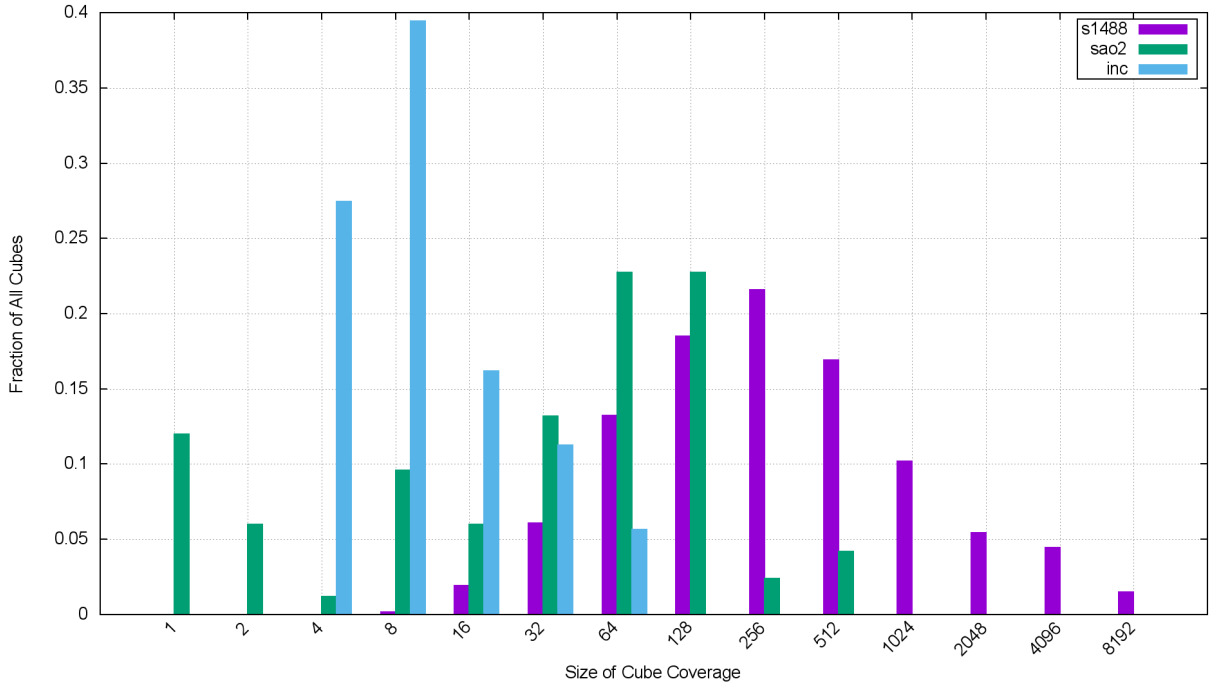


Figure 4.11: Cube Size Distribution for sao2, inc and s1488.

4.4 Conclusions

We presented an approximate logic based SET fault tolerant approach. The approach takes the probability of faults in different types of gates into account and utilizes the error bar estimation to limit computations. The SA and SET fault-injection simulations show promising results for certain types of circuits which are efficiently represented in Sum of Products (SOP) form.

CHAPTER 5

SUMMARY, CONCLUSION AND FUTURE WORK

5.1 Summary

This work proposed two new SET mitigation techniques for ground-level applications. The utilized redundant wire and approximate logic techniques are both generic and can be applied to any combinatorial circuits. The way in which the proposed algorithms incrementally add redundancy provides users with the flexibility to balance the cost and reliability requirements. The work also explored and proposed various techniques (like signature-based evaluation, table-based implication identification and error bar) to accelerate effectively the generation process. Finally, the simulation results substantiated that the proposed mitigation techniques are more efficient than those in previous works.

The separated summary of the techniques are listed below.

Low Cost Mitigation of Logic Faults with Redundant Wires

The work has presented new algorithms which are able to identify quickly all the implications in a logic circuit. These algorithms are able to find more implications and are significantly faster than previous algorithms.

The work then developed a systematic approach for assessing the protective benefit when a redundant wire is inserted into a circuit. This approach takes into account the beneficial masking effects of the redundant wire as well as the fact that the redundant wire will increase the propagation of other faults. It also evaluates the new faults that can occur in the added gates. The assessment is performed using signature techniques which make it fast. The simulation data shows that the

estimations are highly accurate. Using the list of implications and the evaluation technique, we implemented a greedy algorithm to add iteratively redundant wires in order to protect logic circuits from faults.

In the simulation section, it shows that the algorithm for identifying implications is significantly faster than previous techniques. The quality of the protection solutions found using our final algorithm is significantly better (288%) than those presented in [50].

The work studied the role that synthesis constraints on the original circuit play when adding redundant wires. It was found that netlists generated with aggressive timing constraints are better candidates for adding redundancy. The work also performed a study of the sensitivity of the results to the length of the signature and found that even with 32-bit signatures, the reduction in the failure rate protection is only 12%. Thus, the algorithms could be made even faster through the use of shorter signatures, with only a modest reduction in the quality of the results.

Finally, the algorithms presented are more efficient than previous work for generating redundant wires and have demonstrated that this technique can generate low cost (typically less than 10%) solutions for masking faults in large circuits.

New Approximate Logic based Approaches for Synthesis of Redundant Combinatorial Logic for Selective Fault Tolerance

The proposed algorithm generates circuits which achieve a modest (<3x) FIT rate reduction for a lower cost than TMR. Protecting random, combinatorial logic from faults is a challenging problem. The proposed approach, based on an SOP representation and using realistic fault injections shows promising results for certain types of circuits which are efficiently represented in SOP form.

The overheads are significant compared to those for protecting memories or other regular structures, but, the soft FIT rates due to combinatorial logic are no longer trivial and in an industrial context, generic solutions are required which can be applied directly to a gate-level netlist.

The proposed technique can be applied automatically to any combinatorial circuit and the algorithm is scalable due to the use of a two-level logic representation and the reduced number of fault-injection simulations. Furthermore, the algorithm takes into account the relative probability

of faults in different types of gates, something that is not considered in previous work.

The work studied the effect of logic sharing on the protective performance. The simulation results indicated that when a small level of redundancy is required, the performance of both is similar, but for a high level of redundancy, separating F and H logic is recommended. The work also explored the relationship between the distribution of logic cubes and the effectiveness of the proposed technique. We only touched the surface of this topic, but did show that the technique performs better on circuits with large logic cubes.

5.2 Conclusion

In this paper, two novel SET mitigation methods based on redundant wire and approximate logic techniques are proposed. Both methods are generic and provide great flexibility in the trade-off between SET coverage and overhead. The simulation results showed low computational requirement and efficient SET mitigation ability. The proposed mitigation techniques provide designers more choices in developing reliable combinatorial logic in ground-level applications.

5.3 Future Work

The current results are collected from fault-injection simulations. In order to validate the proposed techniques in a real environment, future work will need to implement the techniques on real chips for radiation tests.

REFERENCES

[1] S. Buchner, M. Baze, D. Brown, D. McMorrow, and J. Melinger, "Comparison of Error Rates in Combinational and Sequential Logic," *IEEE Transactions on Nuclear Science*, vol. 44, no. 6, pp. 2209–2216, 1997.

[2] J. R. Schwank, M. R. Shaneyfelt, and P. E. Dodd, "Radiation Hardness Assurance Testing of Microelectronic Devices and Integrated Circuits: Radiation Environments, Physical Mechanisms, and Foundations for Hardness Assurance," *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 2074–2100, 2013.

[3] D. Binder, E. C. Smith, and A. B. Holman, "Satellite Anomalies from Galactic Cosmic Rays," *IEEE Transactions on Nuclear Science*, vol. 22, no. 6, pp. 2675–2680, 1975.

[4] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–315, 2005.

[5] E. Normand and T. J. Baker, "Altitude and latitude variations in avionics SEU and atmospheric neutron flux," *IEEE Transactions on Nuclear Science*, vol. 40, no. 6 pt 1, pp. 1484–1490, 1993.

[6] A. H. Taber and E. Normand, "Investigation and Characterization of SEU Effects and Hardening Strategies in Avionics," in *Technical Report of Defense Nuclear Agency*, 1995.

[7] T. C. May and M. H. Woods, "A New Physical Mechanism for Soft Errors in Dynamic Memories," in *16th International Reliability Physics Symposium*. IEEE, 1978.

[8] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset

in digital microelectronics," *IEEE Transactions on Nuclear Science*, vol. 50 III, no. 3, pp. 583–602, 2003.

[9] H. T. Nguyen, Y. Yagil, N. Seifert, and M. Reitsma, "Chip-level Soft Error Estimation Method," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 365–380, 2005.

[10] A. Evans, S. Wen, and M. Nicolaidis, "Case Study of SEU Effects in a Network Processor," in *IEEE Workshop on Silicon Errors in Logic - System Effects (SELSE)*, Urbana-Champagne, United States, Mar. 2012.

[11] M. Baze and S. Buchner, "Attenuation of Single Event induced Pulses in CMOS Combinational Logic," *IEEE Transactions on Nuclear Science*, vol. 44, no. 6, pp. 2217–2223, 1997.

[12] S. Buchner, K. Kang, D. Krening, G. Lannan, and R. Schneiderwind, "Dependence of the SEU window of vulnerability of a logic circuit on magnitude of deposited charge," *IEEE Transactions on Nuclear Science*, vol. 40, no. 6, pp. 1853–1857, 1993.

[13] N. Mahatme and I. Chatterjee, "Analysis of soft error rates in combinational and sequential logic and implications of hardening for advanced technologies," in *IEEE International Reliability Physics Symposium, IRPS*, 2010, pp. 1031–1035.

[14] N. Seifert, D. Moyer, R. Mueller, R. Hokinson, N. Leland, M. Shade, and L. Massengill, "Frequency dependence of soft error rates for sub-micron CMOS technologies," *International Electron Devices Meeting*, pp. 14.4.1–14.4.4, 2001.

[15] (2017, Mar.) Moores law. [Online]. Available: https://en.wikipedia.org/wiki/Mooreslaw

[16] L. W. Massengill, B. L. Bhuva, W. T. Holman, M. L. Alles, and T. D. Loveless, "Technology scaling and soft error reliability," in *IEEE International Reliability Physics Symposium Proceedings*, 2012.

[17] S. E. Diehl, J. E. Vinson, B. D. Shafer, and T. M. Mnich, "Considerations for Single Event Immune VLSI Logic," *IEEE Transactions on Electron Devices*, vol. NS-30, no. 6, pp. 4501–4507, 1983.

[18] R. Koga and W. A. Kolasinski, "Effects of Heavy Ions on Microcircuits in Space: Recently Investigated Upset Mechanisms," *IEEE Transactions on Nuclear Science*, vol. 34, no. 3, pp. 46–51, 1987.

[19] D. G. Mavis and P. H. Eaton, "SEU and SET Mitigation Techniques for FPGA Circuit and Configuration Bit Storage Design," in *The Military and Aerospace Application of Programmable Devices and Technologies Conference*, 2000, pp. 26–28.

[20] ——, "Soft Error Rate Mitigation Techniques for Modern Microcircuits," in *IEEE International Reliability Physics Symposium*, 2002, pp. 216–225.

[21] V. Ferlet-Cavrois, P. Paillet, M. Gaillardin, D. Lambert, J. Baggio, J. R. Schwank, G. Vizkelethy, M. R. Shaneyfelt, K. Hirose, E. W. Blackmore, O. Faynot, C. Jahan, and L. Tosti, "Statistical Analysis of the Charge Collected in SOI and Bulk Devices under Heavy Ion and Proton Irradiation-Implications for Digital SETs," *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3242–3252, 2006.

[22] P. E. Dodd, M. R. Shaneyfelt, J. A. Felix, and J. R. Schwank, "Production and Propagation of Single-Event Transients in High-Speed Digital Logic ICs," *IEEE Transactions on Nuclear Science*, vol. 51, no. 6 II, pp. 3278–3284, 2004.

[23] V. Sridharan and D. Liberty, "A Field Study of DRAM Errors," in *Workshop on Silicon Errors in Logic - System Effects*, 2012.

[24] SIA, "The International Technology Roadmap for Semiconductors," Tech. Rep., 2001.

[25] V. Ferlet-Cavrois, L. W. Massengill, and P. Gouker, "Single event transients in digital CMOS - A review," *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 1767–1790, 2013.

[26] R. E. Lyons and W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer Reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962.

[27] K. P. Rodbell, D. F. Heidel, J. A. Pellish, P. W. Marshall, H. H. K. Tang, C. E. Murray, K. A. Label, M. S. Gordon, K. G. Stawiasz, J. R. Schwank, M. D. Berg, H. S. Kim, M. R. Friendlich, A. M. Phan, and C. M. Seidleck, "32 and 45 nm Radiation-Hardened-by-Design (RHDB) SOI Latches," *IEEE Transactions on Nuclear Science*, vol. 58, no. December, pp. 2702–2710, 2011.

[28] N. N. Mahatme, S. Jagannathan, T. D. Loveless, L. W. Massengill, B. L. Bhuva, S. J. Wen, and R. Wong, "Comparison of combinational and sequential error rates for a deep submicron process," *IEEE Transactions on Nuclear Science*, vol. 58, no. 6 PART 1, pp. 2719–2725, 2011.

[29] K. Mohanram and N. Touba, "Partial error masking to reduce soft error failure rate in logic circuits," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*. Institute of Electrical and Electronics Engineers Inc., 2003, pp. 433–440.

[30] Y. Dotan, N. Levison, and D. Lilja, "Fault tolerance for nanotechnology devices at the bit and module levels with history index of correct computation," *IET Computers & Digital Techniques*, vol. 5, no. 4, pp. 221–230, 2011.

[31] A. H. El-maleh and F. Chikh, "A generalized modular redundancy scheme for enhancing fault tolerance of combinational circuits," *Microelectronics Reliability*, vol. 54, no. 1, pp. 316–326, 2014.

[32] Q. Zhou, M. R. Choudhury, and K. Mohanram, "Tunable Transient Filters for Soft Error Rate Reduction in Combinational Circuits," in *13th European Test Symposium*, 2008, pp. 179–184.

[33] V. Srinivasan, a.L. Sternberg, a.R. Duncan, W. Robinson, B. Bhuva, and L. Massengill, "Single-event mitigation in combinational logic using targeted data path hardening," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2516–2523, 2005.

[34] D. Limbrick, N. Mahatme, W. Robinson, and B. Bhuva, "Reliability-aware synthesis of combinational logic with minimal performance penalty," *IEEE Transactions on Nuclear Science*, vol. 60, no. 4, pp. 2776–2781, 2013.

[35] D. B. Limbrick, D. A. Black, K. Dick, N. M. Atkinson, N. J. Gaspard, J. D. Black, W. H. Robinson, and A. F. Witulski, "Impact of logic synthesis on soft error vulnerability using a 90-nm bulk CMOS digital cell library," in *Southeastcon, 2011 Proceedings of IEEE*, 2011, pp. 430–434.

[36] 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," *IEEE Transactions on Nuclear Science*, vol. 56, no. 6, pp. 3050–3056, 2009.

[37] D. G. Mavis and P. H. Eaton, "SEU and set modeling and mitigation in deep submicron technologies," in *IEEE International Reliability Physics Symposium*, 2007, pp. 293–305.

[38] S. Tarnick, "Bounding Error Masking in Linear Output Space Compression Schemes," in *Proc. of Asian Test Symposium*, 1994, pp. 27–32.

[39] S. Almukhaizim, P. Drineas, and Y. Makris, "Concurrent error detection for combinational and sequential logic via output compaction," in *Proceedings - 5th International Symposium on Quality Electronic Design, ISQUED*, 2004, pp. 459–464.

[40] K. Mohanram, E. S. Sogomonyan, N. A. Touba, and L.-c. Ced, "Synthesis of Low-Cost Parity-Based Partially Self-Checking Circuits," in *On-Line Testing Symposium, 9th IEEE*, 2003, pp. 35–40.

[41] A. Morozov, V. Saposhnikov, and V. Saposhnikov, "New Self-Checking Circuits by Use of Berger-Codes," in *6th IEEE International On-Line Testing Workshop*, 2000, pp. 141–146.

[42] J.-c. Lo, "Single fault masking logic designs with error correcting codes," in *Proceedings of International Workshop on Defect and Fault Tolerance in VLSI*, 1995, pp. 296–304.

[43] A. Dutta and A. Jas, "Combinational Logic Circuit Protection Using Customized Error Detecting and Correcting Codes," *Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on*, pp. 68–73, 2008.

[44] B. D. Sierawski, B. L. Bhuva, and L. W. Massengill, "Reducing soft error rate in logic circuits through approximate logic functions," *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3417–3421, 2006.

[45] M. R. Choudhury and K. Mohanram, "Approximate logic circuits for low overhead, non-intrusive concurrent error detection," in *Proceedings of the conference on Design, automation and test in Europe, DATE*. New York, New York, USA: ACM Press, 2008, pp. 903–908.

[46] ——, "Low Cost Concurrent Error Masking Using Approximate Logic Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 8, pp. 1163–1176, aug 2013.

[47] A. Sanchez-Clemente, L. Entrena, M. Garcia-Valderas, and C. Lopez-Ongil, "Logic masking for SET Mitigation Using Approximate Logic Circuits," in *IEEE 18th International On-Line Testing Symposium, IOLTS*. Ieee, jun 2012, pp. 176–181.

[48] A. Sanchez-Clemente, "Error masking with approximate logic circuits using dynamic probability estimations," in *IEEE 20th International On-Line Testing Symposium, IOLTS*, 2014, pp. 134–139.

[49] F. Yuan and Q. Xu, "InTimeFix: A low-cost and scalable technique for in-situ timing error masking in logic circuits," in *Proceedings of the 50th Annual Design Automation Conference, DAC*, 2013.

[50] S. Almukhaizim and Y. Makris, "Soft Error Mitigation Through Selective Addition of Functionally Redundant Wires," *IEEE Transactions on Reliability*, vol. 57, no. 1, pp. 23–31, mar 2008.

[51] S. Krishnaswamy, S. M. Plaza, I. L. Markov, and J. P. Hayes, "Enhancing design robustness with reliability-aware resynthesis and logic simulation," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, 2007, pp. 149–154.

[52] ——, "Signature-based SER analysis and design of logic circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 1, pp. 74–86, 2009.

[53] K. C. Wu and D. Marculescu, "A low-cost, systematic methodology for soft error robustness of logic circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 2, pp. 367–379, 2013.

[54] L. Goldstein, "Controllability/observability analysis of digital circuits," *IEEE Transactions on Circuits and Systems*, vol. 26, no. 9, pp. 685–693, 1979. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1084687

[55] L. Goldstein and E. Thigpen, "SCOAP: Sandia controllability/observability analysis program," *Design Automation, 1980. 17th*, 1980. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1585245

[56] K. Butler and R. Kapur, "The roles of controllability and observability in design for test," in *IEEE VLSI Test Symposium*, 1992, pp. 211 – 216. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=232754

[57] S. Chang, W. Jone, and S. Chang, "TAIR: Testability analysis by implication reasoning," *Computer-Aided Design of ...*, vol. 19, no. 1, pp. 152–160, 2000. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=822627

[58] V. Dhare and U. Mehta, "Object Oriented Implementation of Combinational Controllability and Observability Algorithms," *International Journal of Electronics Engineering*, vol. 2, no. 1, pp. 93–97, 2010.

[59] H. Y.-P. Chang, S. G. Chappell, C. H. Elmendorf, and L. D. Schmidt, "Comparison of parallel

and deductive fault simulation methods," *IEEE Transactions on Computers*, no. 11, pp. 1132–1138, 1974.

[60] S. C. Chang, L. P. P. P. Van Ginneken, and M. Marek-Sadowska, "Circuit optimization by rewiring," *IEEE Transactions on Computers*, vol. 48, no. 9, pp. 962–970, 1999.

[61] Y.-C. Chen and C.-Y. Wang, "Fast detection of node mergers using logic implications," in *IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers, ICCAD*, 2009, pp. 785–788.

[62] R. Arora and M. S. Hsiao, "Enhancing sat-based equivalence checking with static logic implications," in *18th IEEE International High-Level Design Validation and Test Workshop*, 2003, pp. 63–68.

[63] W. Kunz, "Recursive Learning : A New Implication Technique for Efficient Solutions to CAD," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 9, pp. 1143–1158, 1994.

[64] N. Alves, S. Member, A. Buben, J. Dworak, and R. I. Bahar, "A Cost Effective Approach for Online Error Detection Using Invariant Relationships," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 788–801, 2010.

[65] (2017, Mar.) Nangate 45nm predictive library. [Online]. Available: http://www.nangate.com

[66] A. Evans, "Techniques for Estimating the Effect of Combinatorial SER at the Chip Level," in *CPMT SER Workshop*, 2013.

[67] R. L. Rudell and A. Sangiovanni-Vincentelli, "Multiple-Valued Minimization for PLA Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 5, pp. 727–750, 1987.

[68] (2017, Mar.) Espresso. [Online]. Available: https://embedded.eecs.berkeley.edu/pubs/downloads/espresso

[69] A. Evans, D. Alexandrescu, E. Costenaro, and L. Chen, “Hierarchical RTL-based combinatorial ser estimation,” in *IEEE 19th International On-Line Testing Symposium, IOLTS*, 2013, pp. 139–144.

[70] A. Adeleke, “Logic repair and soft error rate reduction,” Master’s thesis, Vanderbilt University, 2012.