AN INTEGRATED APPROACH FOR SOFT ERROR TOLERANCE OF COMBINATIONAL CIRCUITS

BY

AHMAD TARIQ SHEIKH

A Dissertation Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY
In

COMPUTER SCIENCE & ENGINEERING

APRIL, 2016

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **AHMAD TARIQ SHEIKH** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE & ENGINEERING**.

Dissertation Committee

Dr. Aiman H. El-Maleh (Adviser)

Dr. Sadiq Sait M. (Co-Adviser)

Dr. Muhammad E. S. ElRabaa (Member)

Ashraf

Dr. Ashraf S. Hasan Mahmoud (Member)

Dr. Mahmood K. Niazi (Member)

Dr. Adel Fadhl Ahmed
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

23/5/16
Date

©Ahmad Tariq Sheikh
2016

Dedicated to My

Beloved
Paternal & Maternal Grandparents

ACKNOWLEDGEMENTS

All praise is due to Allah, the most Gracious, the most Compassionate and Merciful.

First of all I would like to thank Almighty Allah for giving me the perseverance to finish this work. My heartfelt gratitude to my mentor and adviser **Dr. Aiman H. El-Maleh** for his perpetual efforts and motivation throughout the research phase. His meticulousness in research has a profound impact on the outcomes of this thesis. At times, I was in an abyss, with no progress in the research. He was always there to drag me out and put me on the right path. If it wasn't for him, I would have lost my way.

I am also grateful to my committee member **Dr. Sadiq M. Sait** for sharing his iotas of wisdom. His three words; *learn*, *justify* and *story* served as a guideline to tackle a research problem of any magnitude. His comments and feedback went a long way in improving the quality of this thesis.

I would also like to thank my committee member **Dr. Muhammad E. S. Elrabaa** for helping me with the technical aspects of the thesis and specially with circuit level simulations. I am also thankful to **Dr. Mahmood K. Niazi** for his feedback from totally different perspective which also improved the quality of this thesis. I am specially thankful to **Dr. Ashraf S. Hasan Mahmoud** for agreeing to serve as a committee member at the eleventh hour. My heartfelt prayers for **Dr. Alaaeldin Amin**, may Allah (SWT) give him fast recovery and good health. Due to his health, he was unable to serve as a committee member.

I acknowledge the generous financial support provided by the KFUPM throughout my stay here. The work in this thesis was fully supported by the grant from Deanship of Scientific Research (DSR) at KFUPM under project # IN131014.

I am deeply thankful to my parents for all their prayers and support throughout my academic career until it culminated at PhD. I can never forget the support and sacrifices of my wife through out my PhD tenure at KFUPM. She was always there through thick and thin. The love and affection of my daughter was the source of happiness and motivation, which kept my morals high.

Lastly, I would like to thank my friends and colleagues for their unwavering support and guidance.

Ahmad

April 2016

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME: Ahmad Tariq Sheikh

TITLE OF STUDY: An Integrated Approach for Soft Error Tolerance of Combinational Circuits

MAJOR FIELD: Computer Science & Engineering

DATE OF DEGREE: April, 2016

With fabrication technology reaching nano-scale, systems are becoming more prone to manufacturing defects with higher susceptibility to soft errors due to the exponential decrease in device feature size. Soft errors, which are caused by radioactive decay and cosmic rays, can flip the output of a gate, resulting in a soft error if it is propagated to the output of a circuit. This work is focused on analyzing, modeling and designing combinational circuits for soft error tolerance with minimum area overhead. The first idea is based on analyzing random pattern testability of faults in a circuit and protecting sensitive transistors, whose soft error detection probability is relatively high, until a desired circuit reliability is achieved or a given area overhead constraint is met. Transistors are protected based on duplicating and sizing a subset of transistors necessary for providing the protection. In the *second approach, the objective is to improve reliability of combinational circuits based on the double modular redundancy scheme. As opposed to TMR, where each module is triplicated followed by a voter, each module in the proposed Double Modular Redundancy (DMR) scheme is duplicated followed by a AND/NAND masking gate. Modules are synthesized by either synthesizing the true or the complement function to maximize soft error masking. The third technique is based on taking advantage of implication relations to maximize the masking probability of a set of critical gates that will maximize the masking of a large number of faults in the circuit. A logic implication denotes an invariant relationship between logic gates in a circuit. Finally, in hybrid scheme, the transistor sizing method is applied to both DMR and implication based technique to further improve the reliability of these methods. Additionally, a novel gate level reliability evaluation technique is proposed that provides similar results to reliability evaluation at the transistor level (using SPICE) with orders of magnitude reduction in CPU time.*

ملخص

الاسم: **أحمد طارق شيخ**

عنوان الدراسة: **نهج متكامل لتحمل الأخطاء الوقتيه في الدوائر التوافقية**

الاختصاص: **علوم وهندسة الحاسب**

تاريخ الشهادة: **ابريل 2016**

مع وصول تقنية التصنيع الى مجال النانومتر أصبحت الأنظمة أكثر عرضة لعيوب في التصنيع مع ارتفاع القابلية للتعرض للأخطاء الوقتية نتيجة للانخفاض الهائل في حجم الأجهزة المصنعة. الأخطاء الوقتية، والتي قد تحدث بسبب الانحلال الإشعاعي والأشعة الكونية، يمكن أن تؤدي الى عكس قيمة البوابة مما يؤدي الى خطأ مؤقت اذا تمكن الى الوصول الى أحد مخرجات الدائرة. يركز هذا العمل على تحليل ونمذجة وتصميم الدوائر التوافقية لتحمل الأخطاء الوقتية مع استخدام الحد الأدنى من المساحة الاضافية في الدائرة. تعتمد الفكرة الأولى على تحليل قابلية عبور الأخطاء العشوائية في الدائرة وحماية الترانزستورات الحساسة، والتي لديها احتمال اكتشاف للأخطاء الوقتية مرتفعة نسبيا، حتى يتم تحقيق موثوقية الدائرة المطلوبة أو الوصول الى حد معين للمساحة الاضافية. تتم حماية الترانزستورات على أساس الازدواج والتحجيم لمجموعة جزئية من الترانزستورات اللازمة لتوفير الحماية. في النهج الثاني، الهدف هو تحسين موثوقية الدوائر التوافقية على أساس تكرار الدائرة الكهربائية واستخدامهما كدائرتين مزدوجتين. وعلى عكس تكرار الوحدات الكهربائية واستخدامهما كثلاث وحدات وربطها مع ناخب (TMR)، يتم تكرار كل وحدة واستخدامهما كوحدتين مزدوجتين متبوعة ببوابة AND/NAND في النهج المقترح (DMR). ويتم تجميع وحدات إما عن طريق تجميع القيمة الحقيقية أو القيمة المكملة لتعظيم حجب الأخطاء الوقتية. وتستند احدى التقنيات على الاستفادة من العلاقات الموجودة بين بوابات الدائرة إلى تحقيق أقصى قدر من تقليل احتمالية عبور الأخطاء عبر مجموعة من البوابات الهامة التي من شأنها تحقيق أقصى قدر من اخفاء عدد كبير من الأعطال في الدائرة. وأخيرا، في مخطط هجين، يتم تطبيق أسلوب تحجيم الترانزستور على كل من تقنية تكرار كل وحدة واستخدامهما كوحدتين مزدوجتين DMR المقترحة وتقنية الاستفادة من العلاقات الموجودة بين بوابات الدائرة من أجل تحسين موثوقية هذه الأساليب. بالإضافة إلى ذلك، تم اقتراح تقنية لتقييم الموثوقية على مستوى البوابة والتي توفر نتائج مماثلة لتقييم الموثوقية على مستوى الترانزستور (باستخدام برنامج سبايس) بالاضافة الى تخفيض الوقت اللازم للتقييم بشكل كبير.

CHAPTER 1

INTRODUCTION

In the year 2000, mysterious system crashes started to infect the internet and telecommunication systems across United States. The crashes mostly occurred in the high-end Sun Microsystems servers. This incident was serious enough that it caught the major headlines in media [12]. More recently in 2010, a small glitch in the voyager 2 space mission forced NASA engineers to suspend its operation [13]. The root cause for both these events was found to be *Soft Errors*.

We are living in an era of smart phones and smart devices that have the computing power of desktop PCs of few years back, yet, they can be occupied in our palms. This shrinking of area is still continuing, posing challenges to the design of efficient synthesis tools. Recent years have witnessed a tremendous development in the field of VLSI. This development is directly correlated with Moores law [14], which states that the number of transistors that can be fabricated in a chip gets doubled every 18 months. As the CMOS technology is continuously improving and shrinking to the nanometer scale, quantum mechanical effects come into the picture barring the additional scaling of CMOS devices. This has opened new avenues of research to investigate new technologies for circuit design. Nanotechnology-based fabrication of devices and circuits promises extra density and performance. However, studies have indicated that high-density chips, upto an order of 10^{12} $devices/cm^2$ are increasingly accompanied by manufacturing defects and susceptible to dynamic faults during the chip operation [15, 16]

Nanoscale devices are limited by several characteristics, most dominant are the devices higher defect rates and the increased susceptibility to soft errors. These limiting characteristics are due to two sources [17]:

1. Inherent randomness in the bottom-up manufacturing process, which results in a large number of defective devices during the fabrication process.

2. Reduced noise tolerance of these devices which is responsible for inducting device malfunctions by external influences like EMI (electromagnetic interference), thermal perturbations and cosmic radiations.

Generally, errors can be categorized as either permanent or transient errors. Permanent (hard) errors may occur during manufacturing process or during the lifetime of a device. Transient (soft) errors can arise due to multiple sources like high-energy particles, coupling, power supply noise, leakage and temporal circuit variations. The transient error can last for one or many clock cycles. Soft error is a phenomena that suddenly changes the data or signal bits of an electronic system for a specific period of time. During this time, the output of the system could remain in an erroneous state. Once the effect of soft error is vanished, the system returns to its normal operation as if nothing had happened. There is no certain pattern regarding the occurrence of soft errors. However, the relative flux at a terrestrial altitude is used to quantify the intensity of sources of soft errors [5]. Both of the aforementioned types of errors i.e., hard and soft errors, affect the reliability of a circuit if they are not tolerated. Reliability of a circuit can be defined as its ability to function properly despite the existence of such errors. In this chapter, we discuss different manifestations that result in soft errors

1.1 An Overview of Soft Errors

Soft errors first started to appear in electronic systems in the $1970's$. It was actually first theorized by Wallmark et al. [18] that the silicon device dimensions would be limited to $10\mu m$ due to cosmic rays in the atmosphere. Soon, reports emerged that highlighted the effect of cosmic-ray induced errors in space electronics [19]. Later on, the proof of cosmic-neutron-induced soft errors was also recorded at the ground level in Cray-1 computers in 1976 [20]. Three years later in 1979, alpha-particle-induced upsets were also recorded in dynamic random memories (DRAM) [21].

Transient faults (SET/SEU) are mainly caused by cosmic-ray neutrons or alpha particles through the materials of ICs. They can either hit in the combinational logic or flip flops of a sequential circuit block. If it happens in the combinational circuit it results in a Single Event Transient (SET) fault. On the other hand, if it happens in the memory cell, it results in a Single Event Upset (SEU) fault. Both SET and SEU pose serious challenges in the reliability of circuits and require due diligence.

To understand the implications of soft errors, consider a scenario when a charged particle strikes a sensitive region in a memory cell, such as a drain of a transistor in the OFF operation mode. In this case, a transient current pulse is generated that can cause a bit flip in the memory cell. A memory cell stores two states i.e. either logic 0 or 1 values. In each state, two transistors are ON and two are OFF. Fig. 1.1 illustrates how an energetic particle can reverse the state of transistors in a circuit, resulting in a bit flip.

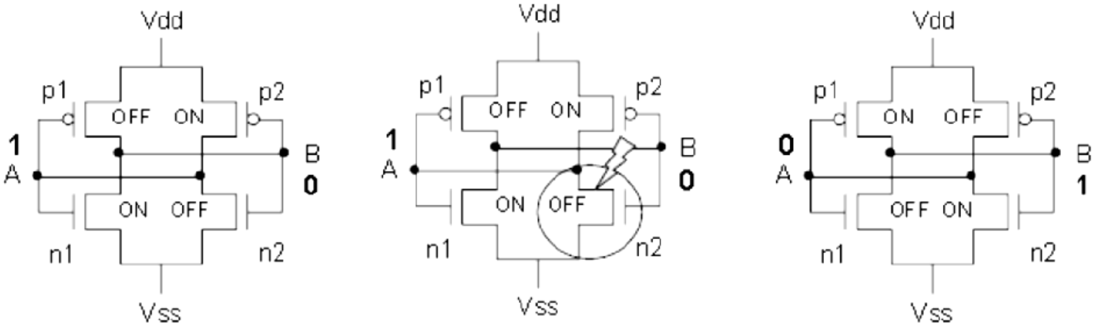


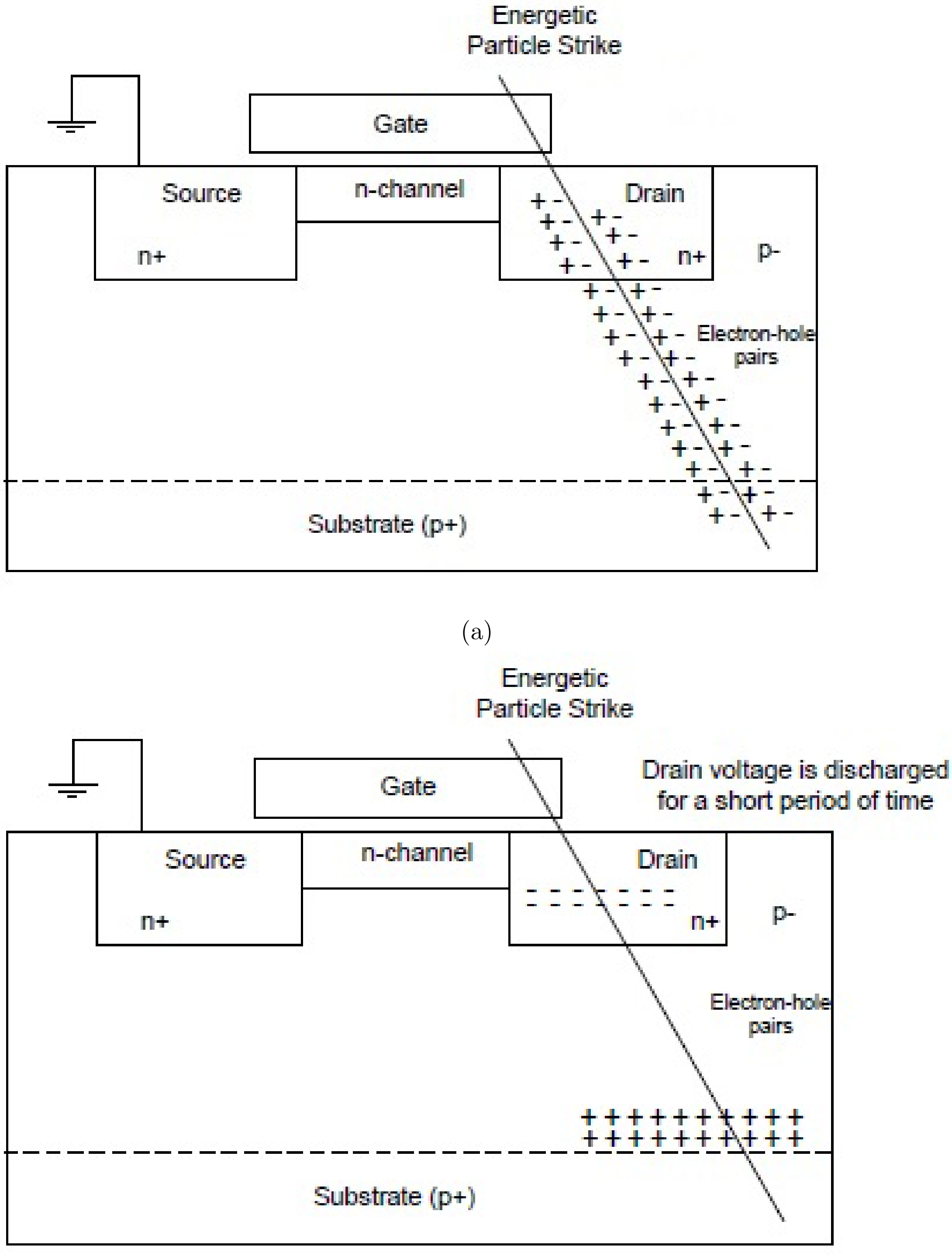
Figure 1.1: Single Event Upset (SEU) effect in an SRAM memory cell.

A single event transient (SET) occurs when a charged particle hits the combinational logic, resulting in a transient current pulse. This can change the logic level of a gate. If this transient has enough width and magnitude, it can result in an erroneous value to be latched. Once it is latched, a single event transient becomes a single event upset (SEU). It is worth to mention that a single SET can produce multiple transient current pulses at the output. This is due to the logic fan-out in the circuit. Hence, SETs can produce multiple SEUs in the memory elements.

For more details on how a transient soft error can change the state of transistor, consider the NMOS transistor shown in Figure 1.2a. The transistor is assumed to be in the OFF state. During normal operation, a current will flow from the drain to the source that makes the transistor ON. If an alpha particle strikes the drain of the NMOS transistor, it loses its energy as it travels along the path inside the semiconductor material. In this period, the particle ionizes the material around it, which results in the generation of electron-hole pairs. Consequently, the holes are collected by the (p+) substrate and the electrons are collected by the drain as shown in Fig. 1.2b. This results in a prompt component of current at the drain in shape of negative pulse. If this prompt current has a high enough charge, this will lead to discharging the voltage at the drain for a very short period of time lasting in the order of 100 to 200 picoseconds [22]. Hence, the state of transistor is changed to ON state in that period of time.

In the previous generations of CMOS technologies, the sizes of CMOS transistors were large enough to neglect the effect of the resulting prompted current. However, with device dimensions shrinking to nanometer scale, SET and SEU faults are no longer considered a small attenuation. Instead, they will be considered as normal circuit signals. Therefore, tolerance of soft and transient errors is no longer limited to specific applications like aerospace applications, and they can no longer be ignored.

In the following sections we will discuss different mechanisms responsible for



(a)

(b)

Figure 1.2: NMOS transistor hit by ion particle.

the occurrence of soft errors in digital systems.

1.1.1 Single Event Transients

When an energetic particle strikes a semiconductor, usually the sensitive region is the reverse-biased pn junction [3,4]. The charge generated as a result of particle strike is due to either *direct ionization* or *indirect ionization.* In direct ionization, when a charged energetic particle strikes the semiconductor it frees electron-hole pairs along its path. In indirect ionization, usually a light particle interacts with the silicon nuclei to generate various heavy ions and charged particles, which in turn can produce charge through direct ionization.

The distance travelled by the energetic particle in semiconductor is measured in terms of *Linear Energy Transfer* (LET). The LET defines the amount of energy loss per unit length of a particle as it passes through the material [3]. LET is expressed in the units of MeV/cm or $MeV{-}cm^2/mg$ if normalized by the material density [23]. In silicon, energy of 3.6eV is required to generate one electron-hole pair and the LET of 97 $MeV - cm^2/mg$ results in the charge deposition of approximately 1pC/μm. A curve of particular interest is the LET of a particle versus the depth it travels through the material. Fig. 1.3 shows one such example. The peak in the charge deposition is referred to as the *Bragg Peak* and it occurs when a particle reaches an energy near $\approx 20MeV - cm^2/mg$.

The energy of a particle is highly correlated to the amount of distance it travels through the device. Fig. 1.4 shows the effect of particle strike on a semiconductor

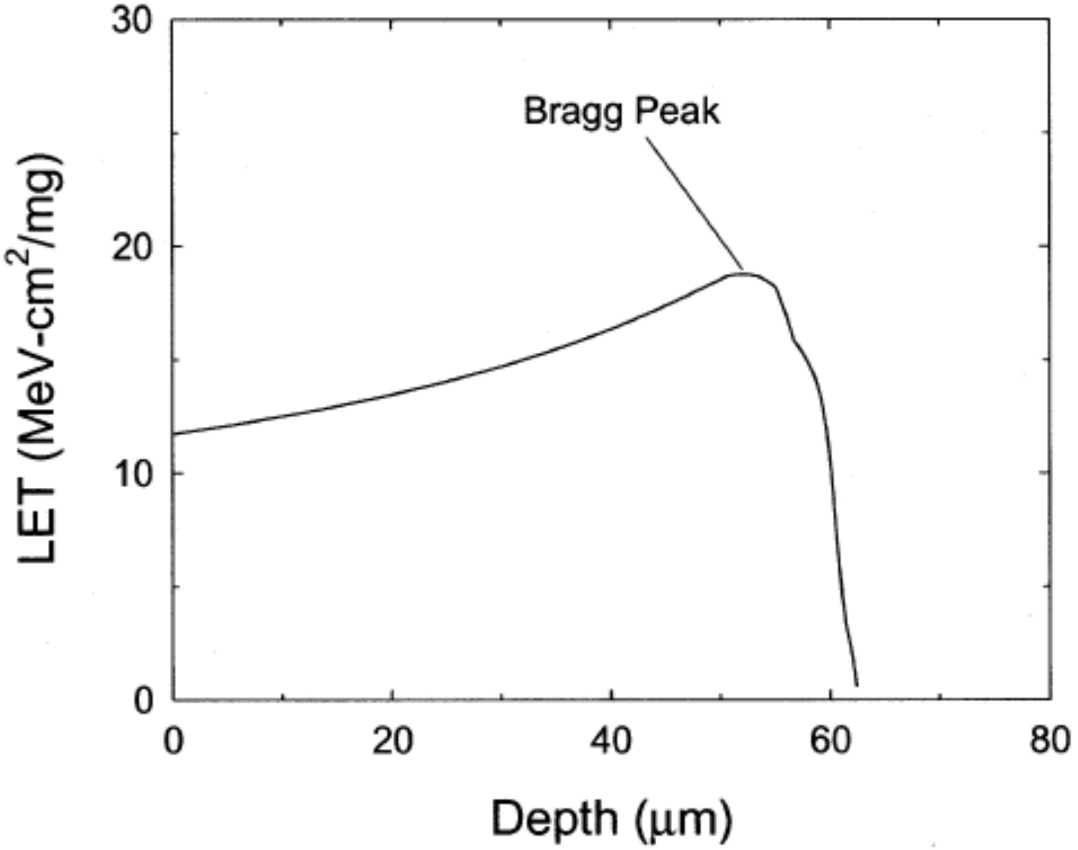


Figure 1.3: Linear energy transfer (LET) versus depth curve for 210-MeV chlorine ions in silicon [3].

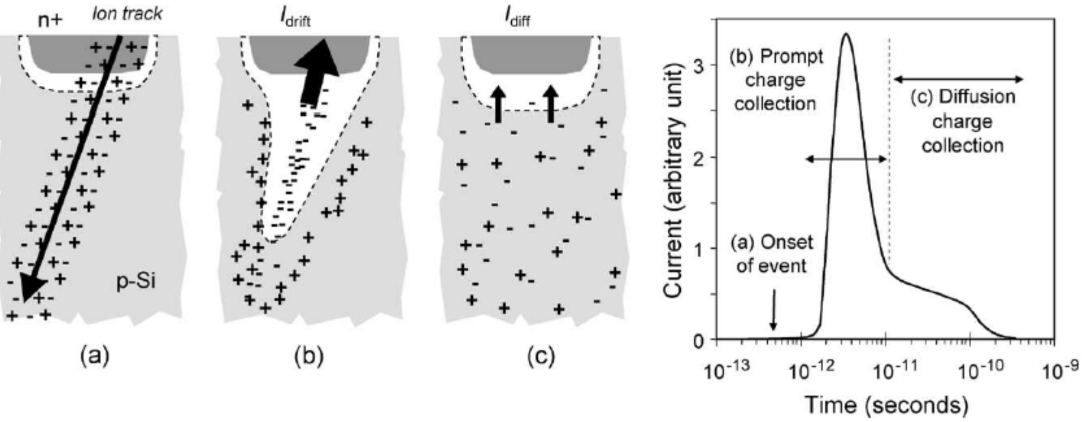


Figure 1.4: Charge generation and collection [4].

material. It can be observed that, once the initial strike happens, numerous electron-hole pairs are generated as the particle travels through the material. The rapid collection of charge happens near the depletion region and a high current value is observed due to $drift$ action. The $drift$ denotes the speed and magnitude of current observed as a result of particle strike. Once the particle loses its energy, the slower charge collection happens due to $diffusion$ action. The overall current pulse due to $drift$ and $diffusion$ is shown in Fig. 1.4d. Baumann [4] and Dodd et al. [3] discussed in detail about the charge collection process in semiconductor materials.

The effect of particle strike can be either transient (soft error) or permanent (hard error). We will now briefly discuss each one of these error types.

Soft Errors

A single event transient (SET) occurs when a charged particle hits the combinational logic, resulting in a transient current pulse. This can change the logic level of a gate. If this transient has enough width and magnitude, it can result in

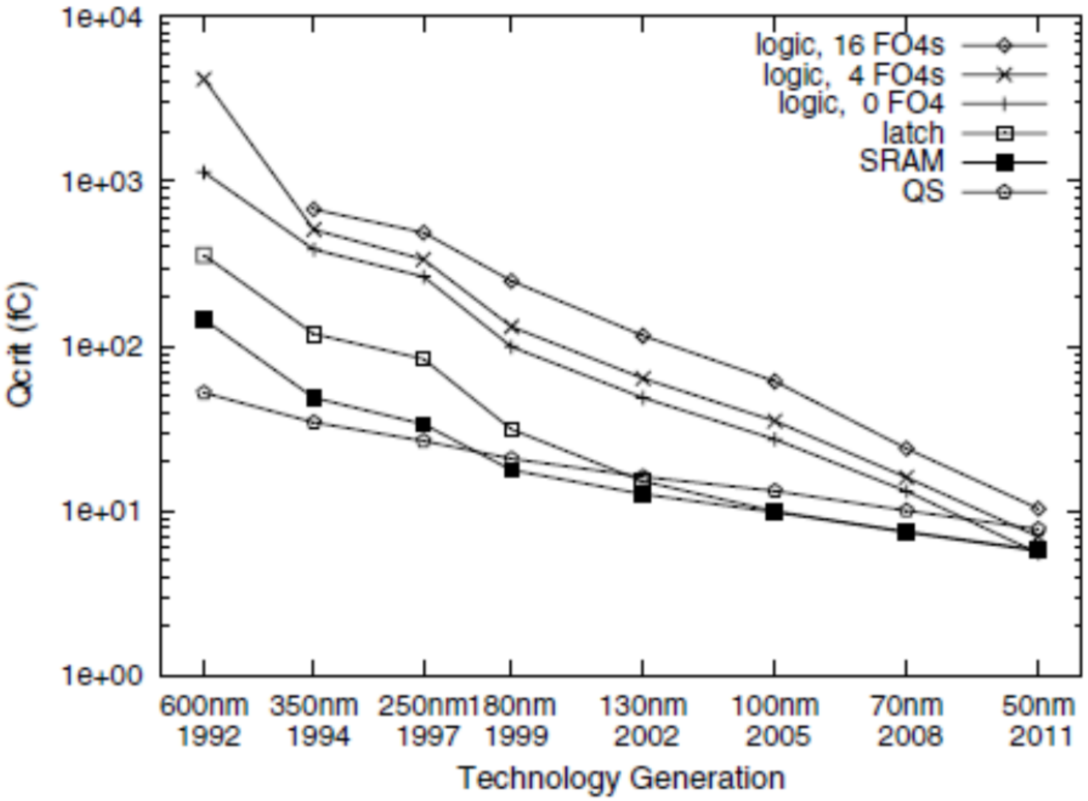


Figure 1.5: Technology generation effect on Q_{crit}.

an erroneous value to be latched. The minimum amount of charge that can cause a SET is called a *Critical Charge* or Q_{crit} of the device. Shivakumar [24] modeled the effects of soft errors on memory devices and logic devices and demonstrated that with increasing technology generation, soft errors will increase by orders of magnitude in logic devices. He also showed that Q_{crit} is also going to be reduced with technology improvement and with the advent of low-power devices, as evident from Fig. 1.5.

Due to aggressive nodes and voltage scaling, the effect of transient fault is no more constrained to a node where the incident particle strikes. This could result in the possibility of deposited charge being simultaneously shared by multiple circuit nodes in the circuit [25–27] leading to the Single Event Multiple Upsets or SEMUs, also referred to as Multiple- Bit Upsets or MBUs [3]. Therefore, soft errors are going to play a critical role in the reliability of modern low-power devices.

Soft Error Rate (SER) is used to quantify the amount of soft errors encountered by a device. The SER of a device also depends on its geographical location. Ziegler et al. [5] presented intensive experimental study over the period of 15 years to evaluate the radiation-induced soft fails in Large Scale Integrated (LSI) electronics at different terrestrial altitudes. They observed that soft failure rates scale directly with the cosmic ray intensity and the energy and density of cosmic rays increases with increase in the altitude. The particle density measured at New York is approximately 100,000/cm^2/yr [5]. SER can also be expressed as *Failure in Time* (FIT) or as *Mean Time Between Failure* (MTBF). 1 FIT is equivalent to 1 failure in 10^9 hours of continuous device operation. MTBF denotes the number of hours between individual failures (1 MTBF = 114,077 FIT). Be noted that, "failure" here is not equivalent to a "soft error", since a "soft error" might not always result in a system failure.

It has been observed that the SER can easily exceed 50,000 FIT/chip if not mitigated [4]. An SER of 50,000 FIT/chip is equivalent to one soft failure every 2 years. For commodity applications, this SER is perfectly tolerable, and even if it occurs, it goes unnoticed. However, if we consider the enterprise, defense, telecommunication etc. systems where there are numerous number of chips per system. Due to the nature of their functionality, the reliability requirements of such systems are very high. Since all the chips are used in parallel, the situation exacerbates due to the fact that the SER effect is multiplicative i.e., SER of one chip is multiplied by the total number of chips to get the SER of a system. For example, in a system with 100 chips, the SER of one soft fail every 2 years will be reduced to one failure per week for that system [4]. Such systems must be provided protection at any cost as their failure will cost grave financial losses to the companies. Fig. 1.6 shows the monthly SER as a function of the number of chips in the system and the amount of memory integrated in each chip. Chapter 2 discusses in detail about soft error tolerance techniques.

Hard Errors

If the striking particle is large enough and has high energy, it can not only result in soft error, but it can also permanently cause defect in the system known as *Single-Event Gate Rupture* (SEGR). In SEGR, the electric field across the transistor gate oxide surpasses the the critical breakdown field allowed [28]. This results in the permanent breakdown of oxide and causes the short circuit through the oxide. For modern CMOS processes at ground level, SEGR is no more of a concern. *Single-Event Latchup* (SEL) occurs when a particle strikes in the vicinity of two neighboring `nmos pmos` transistors. It then activates the parasitic PNPN structure formed by the `nmos-pmos` pair, thereby creating a short impedance path between the power and the ground of the circuit, which could potentially destroy the circuit. Once SEL occurs, it can only be removed by restarting the power supply [28]. SEL can be removed by properly insulating each individual

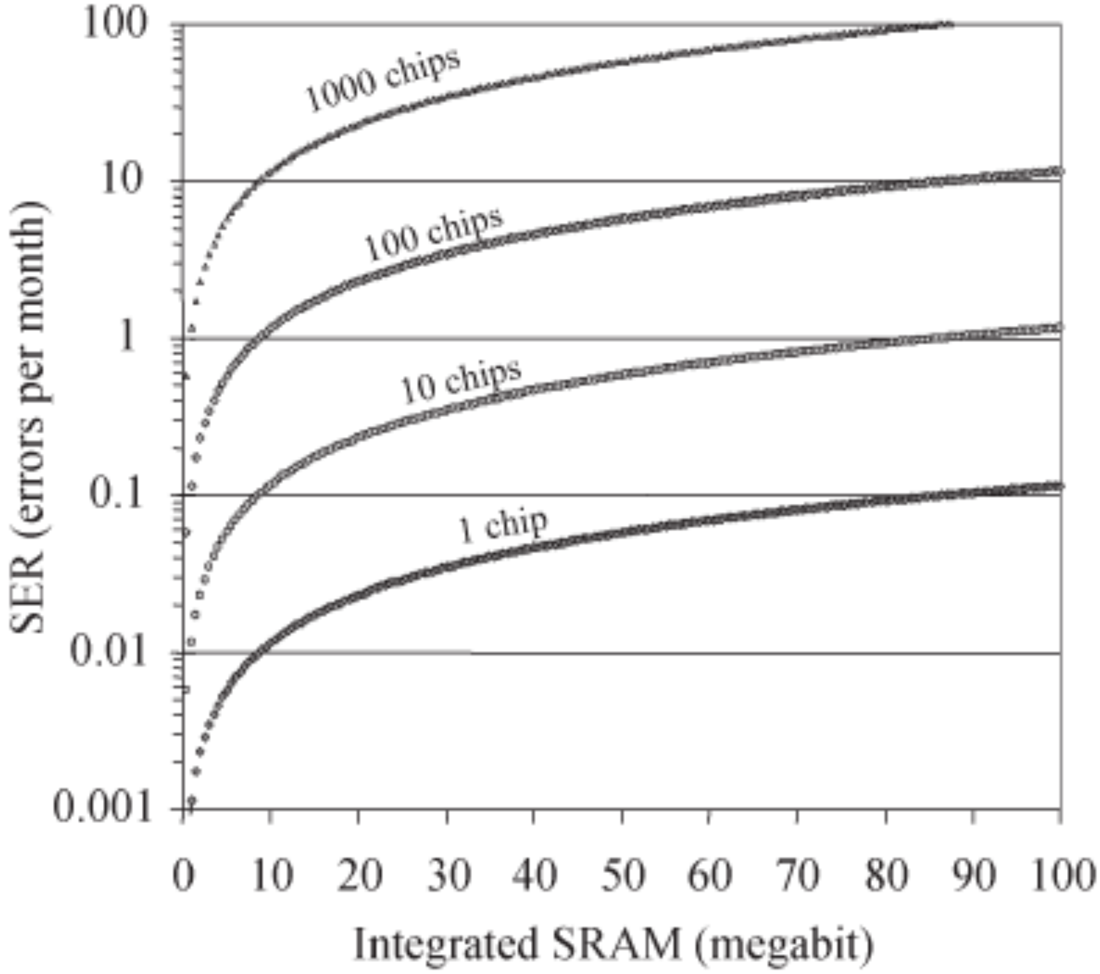


Figure 1.6: Monthly system SER as a function of the number of chips in the system and the amount of embedded SRAM per chip [4].

transistor. *Silicon-on-Insulator* (SOI) silicon processes provide inherent tolerance against SEL. Sexton [29] discusses in detail about various permanent single-event effects in semiconductor devices. The focus in this work is to deal with soft errors only.

1.2 Sources of Radiation

In this section, we will discuss about the different sources of radiation responsible for soft errors in devices. The three dominant sources of soft errors are [4]:

1. Alpha particles,
2. High-energy cosmic rays,
3. Low-energy cosmic rays.

We will now briefly discuss each dominant source one by one.

1.2.1 Alpha Particles

Alpha particles are emitted from the traces of radioactive impurities found in the packaging material of semiconductor devices. The most common radioactive impurities are uranium ^{238}U, ^{235}U and thorium ^{232}Th. These impurities emit alpha particles in the range of 4 to 9 MeV. An alpha particle with 10-MeV of energy travels a distance of $\approx 100\mu m$. Therefore, alpha particles emitted by the packaging and device materials must be considered. Fig. 1.7 shows the energy spectrum of alpha particles emitted from the surface of ^{232}Th. The broad energy

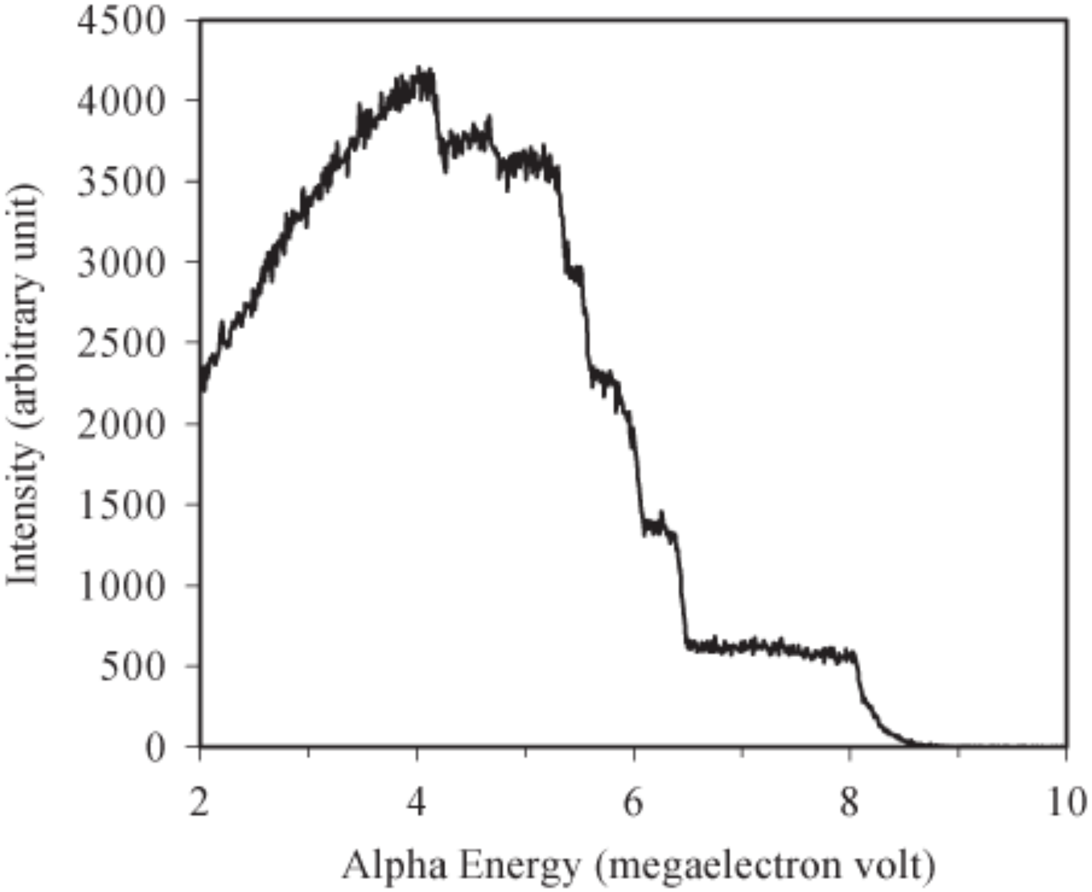
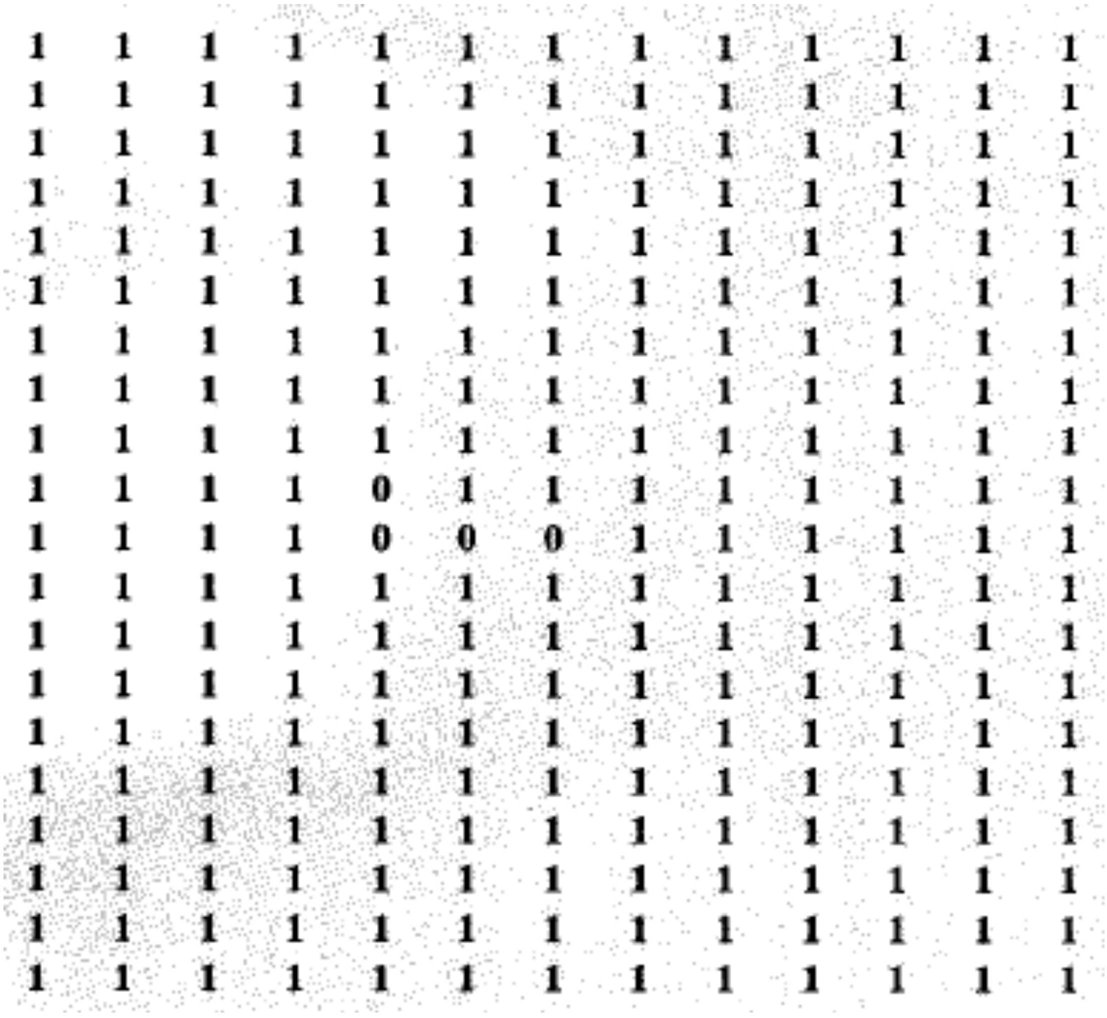


Figure 1.7: Alpha energy spectrum emitted from a thick foil of Th-232 [4].

spectrum depicts the energy loss while travelling different random distances before reaching the surface and being detected. The peak LET of alpha particle is 16 fC/μm. Fig. 1.8 shows the effect of a single radioactive atom decay on computer memory. The figure is a snapshot of the readout portion of 64Kb DRAM memory chip. Initially, the memory was filled with all ones and then a radioactive element was brought closer to it. It was found and can also be observed that a single alpha-particle was able to change the contents of four memory locations from logic "1" to logic "0" [5].

There are two ways to mitigate the effect of alpha particle: 1) purification of



1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1.8: Effect of a single radioactive atom decay on a computer memory [5].

all production materials in close proximity to the device, 2) methods to reduce the probability of alpha particles reaching the sensitive parts of the device [4]. Fabrication companies constantly scrutinize their manufacturing processes and raw materials to meet the requirements of ultra low alpha i.e., alpha emission rate from materials must be $< 0.001\alpha/h\ cm^2$. To achieve that level, ^{238}U and ^{232}Th impurity level must be around 1 part per 10 billion. If manufacturing process and packaging materials of the CMOS devices could be purified so that together they achieve the alpha emission rate of $< 0.001\alpha/h\ cm^2$, then this corresponds to the SER reduction of approximately 20%. Further reduction in emission rate would be extremely expensive beyond this point as cosmic rays are still the dominant source of soft errors in digital circuits.

1.2.2 High-Energy Cosmic Rays

High-energy cosmic rays are the most dominant source of soft errors in digital systems. The origins of the primary cosmic rays are galactic. They react with the Earth's outer atmosphere, which results in the generation of cascade of secondary particles. At sea level, less than 1% of primary flux reaches which is mostly composed of muons, protons, neutron and pions [5]. Fig. 1.9 shows the distribution of cosmic rays at the ground level. It can be observed that neutrons have the higher flux component and are most likely to cause a soft error at terrestrial altitude. The neutrons only generate energetic particles through indirect ionization when they interact with silicon or other dopants present in the chip [4]. The neutron

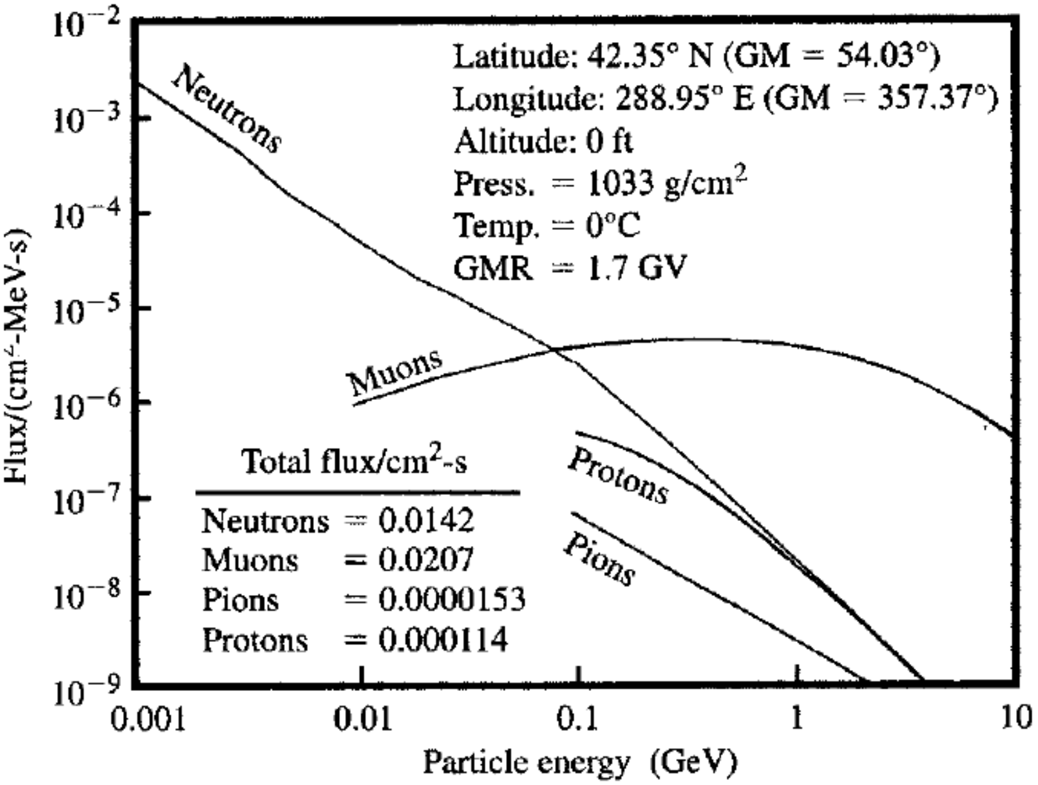


Figure 1.9: Theoretical sea-level cosmic rays [5].

flux is highly dependent on the altitude. For example, at 10,000 ft. from sea level, the cosmic ray flux increases 10×. However, the trend is not linear and starts to saturate at 50,000 ft. Therefore, while designing systems to tolerate soft errors, the altitude factor must also be taken into account as this could have a significant impact on the perceived and real SER.

Unlike alpha particles, the effect of neutron flux cannot be reduced at the chip level using conventional methods like shielding or high-purity materials: one foot of concrete reduces the neutron flux by merely 1.4× [30]. Also, multibit upsets (MBU) and single-event latchups (SEL) mainly occur due to high-energy neutrons as the LET threshold of these events is above 16 fC/μm. Therefore, SER due to cosmic rays must be mitigated by reducing the device sensitivity which could be achieved by employing either design or process modifications.

1.2.3 Low-Energy Cosmic Rays

The third most dominant source of ionizing particles in electronic systems is the low energy neutrons or thermal neutron generated as a result of interaction of low-energy cosmic ray neutrons ($\ll 1$ MeV) and boron. Boron consists of two isotopes: ^{11}B (80.1% abundance) and ^{10}B (19.9% abundance). The ^{10}B isotope is unstable when interacted with the neutrons and has the thermal neutron capture cross section higher than other isotopes by three to seven orders of magnitude. When a ^{10}B nucleus interacts with thermal neutron, it breaks into and Lithium-7 (^{7}Li) recoil nucleus and an alpha particle that can induce a soft error. Boron is used as p-type dopant and also as an implant in the phosphosilicate glass (BPSG) dielectric layers (three orders of magnitude higher than silicon). Therefore, in conventional BPSG-bases processes, BPSG is the main source of soft errors due to boron reactions. The SER due to ^{10}B interaction can be reduced by either completely eliminating the BPSG from silicon processes or by enriching BPSG ^{11}B isotope [31]. The reaction cross-section of ^{11}B is million times smaller than ^{10}B and its reaction with neutron produces gamma rays which lack energy cause any damage.

1.3 Motivation

In this chapter, we discussed about different manifestation responsible for the occurrence of faults in digital systems. It is clear that soft errors are the most challenging types of faults to encounter. With process technologies reaching nano-levels and near threshold operating ranges, the SER due to soft errors is becoming a dominating factor more than ever. Hard errors or permanent faults have been dealt with improvements in the process manufacturing and to a certain extent SER due to soft errors caused by alpha particles and low-energy cosmic rays can also be mitigated with advanced manufacturing processes but at the expense of hefty cost. It is shown that the major source of soft errors is the high-energy cosmic rays which are challenging to deal with due to the variations in their energy levels at different terrestrial altitudes.

The focus in this work is to tolearte the effect of soft errors due to high-energy cosmic rays present in the atmosphere at sea-level. Many techniques have been proposed in the literature to combat soft errors in digital circuits and are discussed in detail in Chapter 2. The motivation here is to propose a set of techniques that could reduce the effect of soft errors in combinational circuits by selectively and smartly adding redundancy. The three proposed techniques lie in different design space of digital circuit design. The first technique applies protection against soft errors at the circuit/transistor-level. The second technique applies redundancy at the gate-level. Both techniques exploit logical masking in combinational circuits to reduce the effect of soft errors. The third technique benefits from the inherent implication relations present between different gates of a circuit for soft error tolerance. Then, integrated approach is proposed that combines the first technique with the second and third technique further improve the reliability of combinational circuits. Finally, a gate-level reliability evaluation method is also proposed that achieves reliability measures very similar to the transistor-level simulations (using SPICE) with order of magnitude less time.

1.4 Problem Statement

Given a combinational logic circuit, the objective is to increase the reliability of this circuit against soft errors while keeping the area overhead as minimum as possible.

1.5 Thesis Contributions

The development of an integrated soft error tolerance framework is proposed to tolerate the soft errors in combinational circuits. The proposed framework is developed based on the evaluation of three proposed techniques where the three techniques are utilized to provide the required soft error tolerance at the minimum possible area overhead. For that matter, the following objectives for this thesis are laid out:

Selective Transistor-Redundancy (STR) Based Fault Tolerance Technique

The focus in this method is to protect only the critical transistors of a circuit. Asymmetric transistor sizing is applied to the most sensitive transistors of the circuit by considering that particles can hit any transistor of a logic gate. The algorithm protects the sensitive transistors or the ones with high probability of failure of the circuit. The proposed algorithm can be utilized in two capacities; 1) apply protection until the probability of failure (POF) of circuit reaches a certain threshold, 2) apply protection until certain area overhead constraint is achieved. The research objective is to quantify the reliability for different protection thresholds and area overhead constraints.

Double Modular Redundancy (DMR) Based Fault Tolerance Technique

In this method, we propose a double modular redundancy (DMR) technique that aims to achieve high reliability with area overhead of nearly double the original area. This method offers significant improvement to the Triple Modular Redundancy (TMR) technique as double instead of triple modular redundancy is used. In this method, redundancy is applied at the gate-level and selectively at the transistor-level. The technique is based on identifying the probability of occurrence of logic “0” or “1” at each output of a circuit and then synthesizing the circuit based on these probabilities.

Implication Based Redundancy Insertion Fault Tolerance Technique

The purpose of this method is to benefit from the inherent implication relations present between different gates of a circuit for soft error tolerance. An implication from a source gate to a target gate indicates that a value assignment at the source gate forces a consistent value assignment at the target gate. For a given target gate, source gates with high probability of "1" or "0" in the logic cone of the target gate are identified. Then, implication relations between the identified source gates and the target gate are explored. If such implication relations are found, then the probability of masking of the target gate can be increased by inserting appropriate extra connections reflecting the identified implication. This approach is an attractive approach as it has the least area overhead over other approaches. However, its effectiveness relies on the ability to identify effective implication relations.

An Integrated Soft Error Tolerance Framework Finally, an integrated framework is developed that combines the STR technique with the DMR and the Implication based fault tolerance technique. It is observed that the DMR and the Implication based fault tolerance techniques are unable to improve the reliability of circuits beyond a certain point. Therefore, the hybrid approaches are proposed to further improve the reliability of circuits. Additionally, the combined application of STR technique with existing techniques results in significant improvement in reliability.

Gate level Reliability Evaluation Technique A novel method to compute the reliability of a circuit at the gate level is proposed. The circuit level simulations performed using SPICE are accurate but become very slow as the circuit size increases. However, there isn't much of an impact of circuit size in gate level simulations. The proposed technique bridges the gap between the circuit level simulations and the gate-level simulations. The proposed gate level reliability evaluation method achieves similar results in comparison to circuit level simulations (using SPICE) with orders of magnitude reduction in CPU time.

1.6 Thesis Organization

The rest of the thesis is organized into the following chapters. The literature review is presented in Chapter 2. Chapter 3 discusses the first proposed technique i.e., Selective transistor redundancy based fault tolerance technique. Double modular redundancy technique is presented in Chapter 4. Implications based fault tolerance method is proposed in Chapter 5. The results and discussion for each proposed method are contained in their respective chapters. A novel reliability evaluation technique is proposed in Chapter 6. Finally, the thesis is concluded in Chapter 7 along with the possible future directions of the current work.

CHAPTER 2

LITERATURE REVIEW

In this chapter, a detailed review is presented about fault tolerance techniques. First, a discussion about inherent error masking types in circuits is presented. Secondly, it wil be shown that fault-tolerant techniques can be classified into three major categories: *hardware redundancy*, *synthesis-based*, and *physical characteristics based* techniques. Literature review of each of the aforementioned category will then be discussed in detail.

2.1 Error Masking Types

Although the incident alpha particles cause voltage transients, these transients must propagate through a certain path to get latched and result in soft errors. The following are three types of masking that shield the SEUs from propagating.

2.1.1 Logical Masking

Logical masking prevents the SET from propagation from fault location to primary outputs of a circuit because of path gate inputs that stop logical transition of the gates output. As shown in Fig. 2.1, there is a particle strike at the output of the A1 gate which results in a wrong logic value of “1” instead of logic value “0”. This wrong value is one of the inputs of the A2 gate.

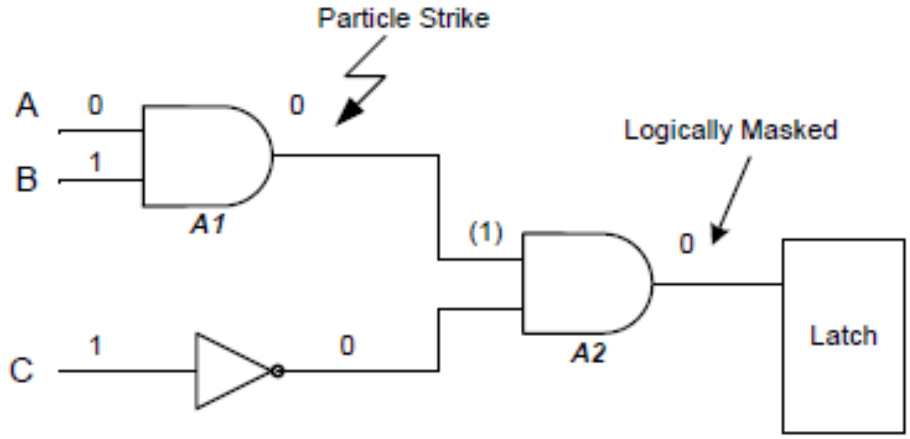


Figure 2.1: Logical Masking.

When one of the inputs of the A2 gate is tied to logic “0”, the output of A2 gate is always logic “0” irrespective of the other input. Therefore, this input of A2 gate is called controlling input. The transient caused by the alpha particle strike is logically masked. Hence a correct value is latched by the following memory element.

2.1.2 Electrical Masking

Electrical masking attenuates or completely masks the SET signal due to electrical properties of gates. The voltage transient caused by the particle strike is attenuated as it propagates through a series of gates. The transient error gets attenuated to an extent where it is ignored by the following memory element.

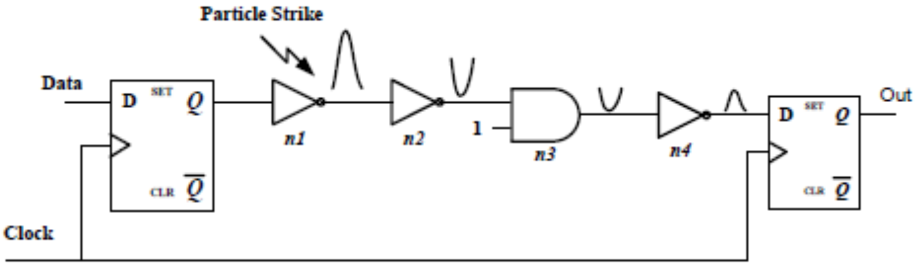


Figure 2.2: Electrical Masking.

As shown in Fig. 2.2 , the voltage pulse generated at the output of the gate n1 attenuates as it passes through gates n2, n3 and n4. The attenuation is due to the parasitic capacitances of succeeding gates. A pulse with duration more than the gate delay attenuates as it propagates [6].

2.1.3 Latching Window Masking

In latching window masking, if a SET doesn't arrive on time, then it will be masked; this depends on the hold and setup times of the target memory element. This is a timing-related masking technique. For a voltage transient to get latched by a memory element, the pulse should be available exactly at the latching window. The transient is masked if it arrives before or after the latching window. As shown in Fig. 2.3, the value of “out” changes only when the glitch is available at the latching window. In all the other cases, the output is error free.

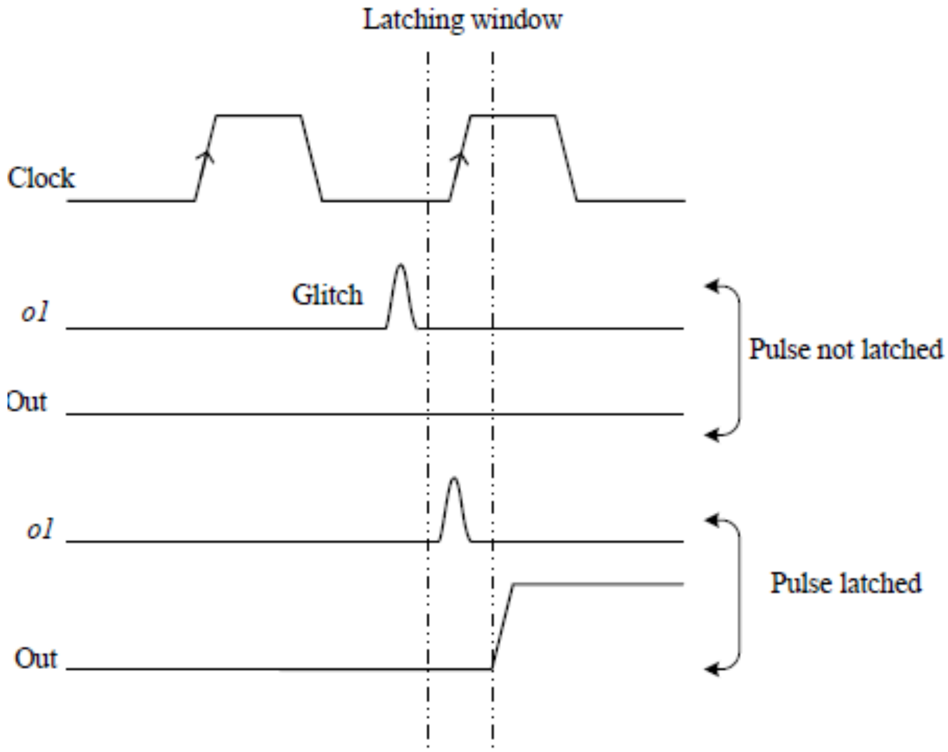


Figure 2.3: Latching window masking [6].

2.2 Fault Tolerance Mechanisms

There are two dominant mechanisms in order to reduce soft error failure rate.

2.2.1 Fault Avoidance

The traditional approach to achieve reliability in systems is mostly based on fault avoidance mechanisms [32]. The purpose of fault avoidance mechanism is to ensure that a part, subsystem or a system doesn’t fail. In this mechanism, defective modules are identified and replaced by other modules. Two widely used methods to perform fault avoidance through configuration are [33]: 1):*Hardware-oriented methods*, where the faulty components are replaced by a spare using additional wires, switches and controllers, 2): *Reconfiguration-oriented*, where fault avoidance is applied through reconfiguration and partial mapping modification. There are several situations in which the fault avoidance approach clearly does not suffice. These include situations where the frequency and duration of repair time are unacceptable, or where the system may be inaccessible to manual maintenance and repair activities. Fault avoidance is often used for handling defects in digital circuits. An alternative approach to fault avoidance is that of fault tolerance.

2.2.2 Fault Tolerance

This approach involves the use of protective redundancy. A system can be designed to be fault tolerant by incorporating additional components and special algorithms, which attempt to ensure that occurrences of erroneous states or erroneous output do not result in later system failures. The degree of fault tolerance depends on the success with which erroneous states, which corresponds to faults are identified and detected, and the success with which such states are repaired or replaced [34]. The fault tolerance mechanism is aimed at either to mask, or to recover from faults once they have been detected [35]. This mechanism attempts to maximize the probabilities of the three masking mechanisms i.e., logical, electrical and latching window masking. Therefore, fault tolerant designs are required for reliable systems that will operate correctly in spite of transient dynamic faults. All fault tolerance approaches rely on some sort of redundancy; otherwise, there is no way to guarantee that a device will not fail if a fault occurs.

2.3 Soft Error Tolerance

Fault tolerance techniques work on circuit level or higher levels of abstractions to achieve soft error rate (SER) improvement. Fault tolerance techniques for combinational circuits can be classified into three major categories: *hardware redundancy*, *synthesis-based* and *physical characteristics based* techniques.

In this section, a survey of the current fault-tolerant methods to tolerate SEU/SET in combinational circuits are discussed.

2.3.1 Hardware Redundancy Techniques

Hardware redundancy methods are based on adding redundant hardware. Multiple modules are used to represent the same function in order to maximize masking of errors. Multiple copies of either the entire circuit or part of the circuit are used as redundant hardware. Redundancy can be added at the module-level, gate level, transistor-level [36] or even at the software level. At the software level, certain software transformations are applied to reduce the vulnerability of critical instructions of a program [37].

Von Neumanns Multiplexing

John von Neumann in the 1950s [38] first initiated the idea of reliable systems using unreliable components. His idea is based on replacing the processing unit by multiplexed units. A unit consists of an *executive stage* and the *restorative stage*. The basic functions of the unit are performed by the *executive stage*, while the error correction is performed by the *restorative stage* due to the errors caused by the *executive stage*. In the *executive stage*, a unit is replaced by N multiplexed units having N copies of every input and output of the unit. The inputs are randomly paired together to feed the N units. For example, consider the case when the processing unit is a single 2-input NAND gate, with N=*4*, Von Neumann multiplexing is shown in Fig. 2.4. The unit U represents a random permutation of the input signals. The two inputs of each NAND gate are selected randomly from the first and second inputs X and Y respectively. The restorative stage is constructed the same way as the executive stage. However, the outputs of the executive stage are duplicated and used as inputs for the restorative stage. Note that, this approach will invert the result if its used only once, thus, two steps are required. By defining some critical level Δ such that $0 < \Delta < 1/2$, if the number of lines carrying a positive state (logic 1) is larger than $(1 - \Delta) \times N$, it considers this as a positive state of the bundle, if it was less than Δ, it interprets this as negative state (logic 0). In cases where the number of positive state lines does not meet either of these criteria, then the output is not decided, and so a fault will occur.

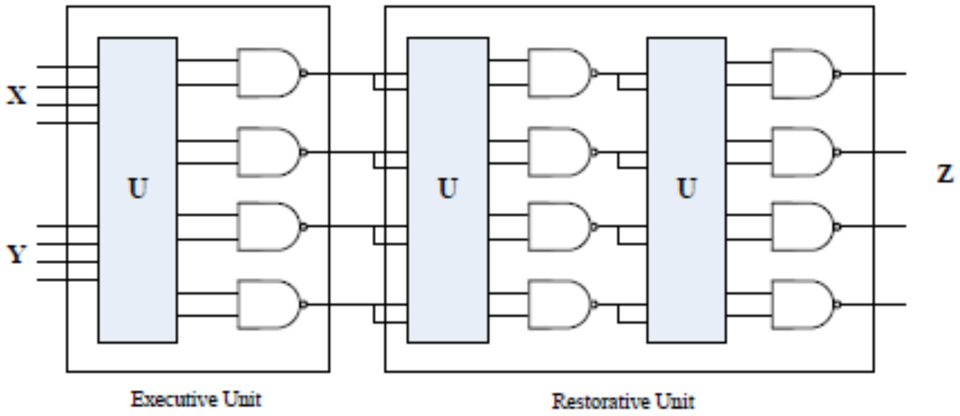


Figure 2.4: Von Neumanns logic for 2-input NAND gate with N = 4.

Giving a probability of failure ϵ for each gate, Von Neumanns structure requires a large amount of redundancy and a low error rate for individual gates. For deep logic with a gate failure probability $\epsilon = 0.01$ and $N = 100$, it is shown in [39] that a circuit failure probability in the order of 10^{-6} can be obtained. This required amount of redundancy is huge and is considered impractical. In order to reduce this large amount of redundancy, the works in [40,41] combine NAND multiplexing with reconfiguration.

Triple Modular Redundancy (TMR)

Triple Modular Redundancy is one of the most well-known techniques to tolerate soft/hard errors in combinational circuits [42, 43]. Its a special case of the NMR system. An NMR system (also known as M-of-N system) is a system that consists of N modules and needs at least M of them for proper operation. TMR is a system where M=2 and N=3, which consists of three functionally identical copies of the original circuit that feed a 2- out-of-3 majority voter as shown in Fig.

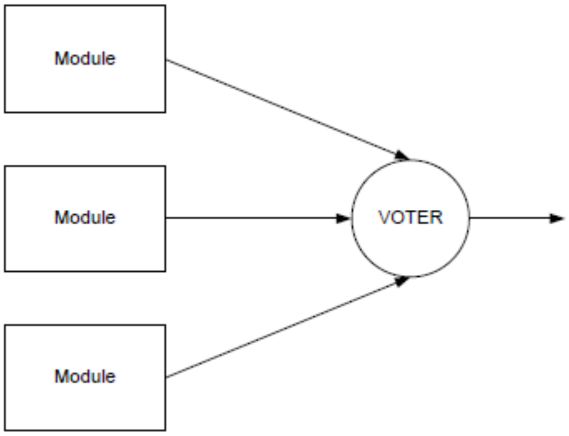


Figure 2.5: A Triple Modular Redundant (TMR) structure.

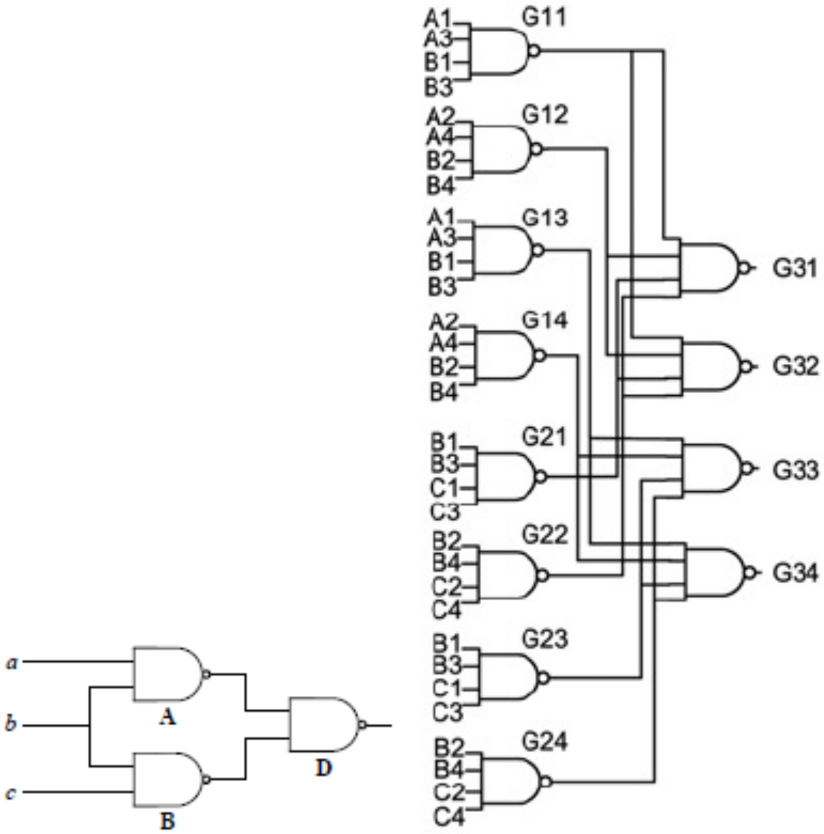
2.5. If 2 modules out of 3 produce expected correct results, then the majority of the modules produces correct results, and so the error in the third module will be masked. However, TMR suffers from high overhead in terms of area and power (more than 200%).

In a structure where M=2 and N=3, the voter selects the majority vote. If a single voter is used, that voter becomes a critical point of failure and the reliability of the TMR structure is limited by that of the final arbitration unit (i.e., voter), which makes the approach difficult in the context of highly integrated nano-systems [17]. Despite this limitation, TMR is heavily used in practice, especially when single faults are needed to be protected. Even in the case of multiple faults, some of these faults could be masked due to electrical and logical masking in each module.

Interwoven Redundant Logic and Quadded Logic

Pierce [44] suggested another scheme called interwoven redundant logic. This scheme considers two types of faults 0 → 1 and 1 → 0 faults. The error correction mechanism in interwoven redundant logic depends on asymmetries in the effects of these two types of binary errors. The effect of a fault depends on the value of the input and the type of gate. Consider a NAND gate, for an instance, if the value of one of the inputs is 0 while it should be 1, the output of NAND gate will be 1 regardless of the values of other inputs. In this case the output will be stuck at 1. On the other hand, if an input value is 1 while it should be 0, the output will depend on other inputs and the output will not be stuck. The type of faults that cause the output to be stuck is considered as critical; the other type is subcritical in the sense that its occurrence alone does not cause an output error. Hence, alternating layers of NAND (or NOR) gates can correct errors by switching them from critical to subcritical.

Quadded logic [45] is an ad hoc configuration of the interwoven redundant logic. It requires four times as many circuits, interconnected in a systematic way, and it corrects errors and performs the desired computation at the same time. A quadded circuit implementation based on NAND gates replaces each NAND gate with a group of four NAND gates, each of which has twice as many inputs as the one it replaces. The four outputs of each group are divided into two sets of outputs, each providing inputs to two gates in a succeeding stage. The interconnections in a quadded circuit are eight times as many as those used in the non-redundant



(a) Original circuit. (b) Quadded logic circuit.

Figure 2.6: Quadded logic example.

form. In a quadded circuit, a single critical error (1 → 0) is correctable after passing through two stages of logic and a single sub-critical error (0 → 1) will be corrected after passing a single stage. In quadded logic, it must be guaranteed that the interconnect pattern at the output of a stage differ from the interconnect patterns of any of its input variables. While quadded logic guarantees tolerance of most single errors, errors occurring at the last two stages of logic may not be corrected. Fig. 2.6 shows an example of a quadded logic circuit.

Partial Error Masking Scheme Based on TMR

In [7], a partial error masking scheme is proposed based on TMR shown in Fig. 2.7. It targets the nodes with the highest soft error susceptibility. Two reduction heuristics are used to reduce soft error failure rate, namely, cluster sharing reduction and dominant value reduction. Instead of triplicating the whole logic as in TMR, only the nodes with highest soft error susceptibility are triplicated, the rest of the nodes are clustered and are shared among the triplicated logic. The dominant value reduction heuristic exploits the fact that the logic value “0” and logic value “1” soft error susceptibility of certain primary outputs is highly skewed. Such outputs are identified and the triplication is replaced by duplication. The 2-out-of-3 majority is replaced by AND (OR) logic. The Generalized Modular Redundancy (GMR) [46] scheme takes into account the probability of occurrence of each combination at the output of a circuit. The redundancy is then added to only those combinations that have high probability of occurrence, while the remaining combinations can be left un-protected to save area.

Fault Tolerance Based on History Index of Correct Computation

A more recent technique based on TMR maintains a history index of correct computation (HICC) module to select the correct result [8]. Instead of using merely majority voting to transmit results, HICC module uses the history indices of redundant units to transmit the correct computation. It represents a measure of a hardware units reliability. The most reliable unit is the unit with the highest

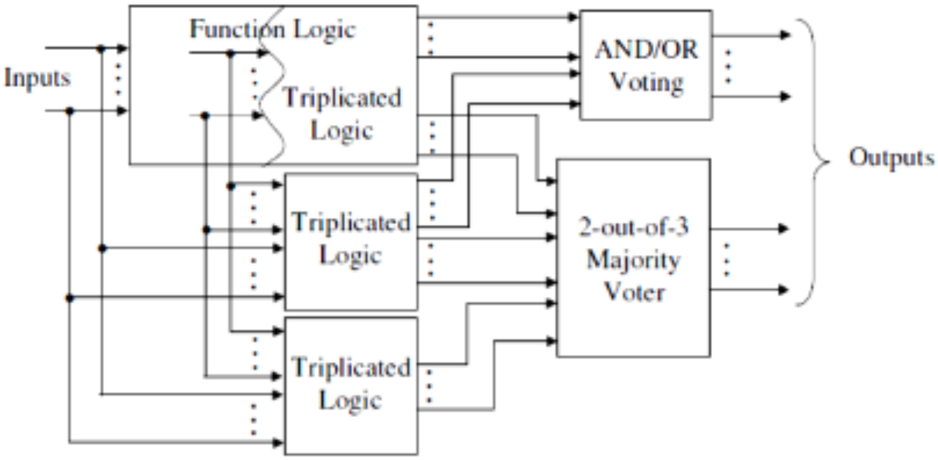


Figure 2.7: Partial error masking scheme [7].

history index. The computations of other redundant units that implement the same function are ignored.

Fig. 2.8 shows an example that demonstrates the concept of the HICC module. In the figure, an ALU module is triplicated as units A, B, and C. The result selector decides the unit with the correct result based on stored history index of each unit. The unit with the highest index is considered to be the most reliable unit, and its result is transmitted. When all units have the same history index value, a bitwise majority voting is used to decide the result. After that, the history index of each unit is incremented by 1 if its result is identical to the result of majority; otherwise it is decremented by 1. The HICC logic is distributed within the modules themselves. Hence, unreliable modules are identified simultaneously in real time and are ignored.

Maximizing the reliability of a system based on nano-devices may require a combination of techniques [47]. Previous results that used error correcting codes (ECCs) and TMR at the bit and module levels demonstrated that recursive TMR

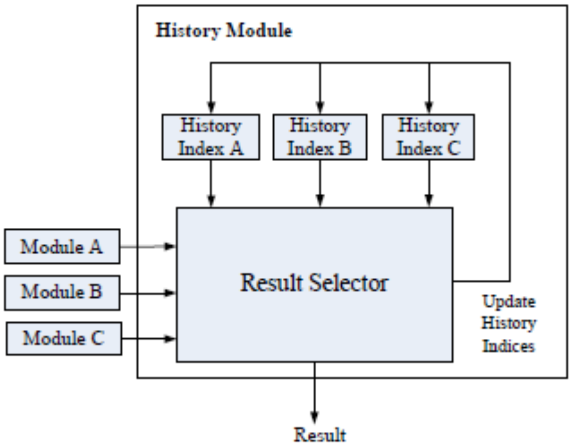


Figure 2.8: HICC module [8].

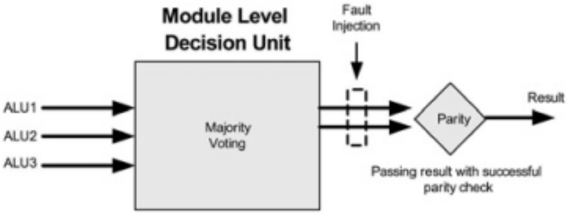


Figure 2.9: Enhanced majority voting with parity checking at the module level [8].

at both levels has the best resilience to noise [48]. Thus, they combined redundancy and reconfiguration to make the system more tolerant of faults. To increase the fault tolerance of the error-prone decision units at the module level, a second copy of the result is stored with an additional parity bit, as illustrated in Fig. 2.9 and Fig. 2.10. The parity checker transmits the even parity result. History indices also have an additional parity bit. The index is updated if even parity is detected. They have stated that without such extra redundancy at the module level, HICC performance is deteriorated.

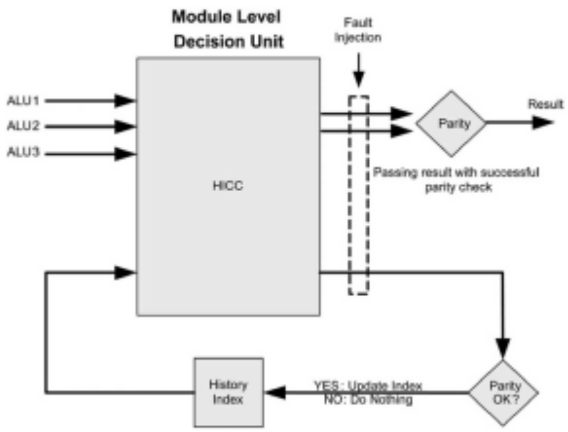


Figure 2.10: Enhanced HICC unit with parity checking at the module level [8].

Double Modular Redundancy (DMR)

Teifel [9] proposed a Double/Dual Modular Redundancy (DMR) scheme that utilizes voting and self-voting circuits to mitigate the effects of SETs in digital integrated circuits. A *Self-voter* shown in Fig. 2.11 is a 3-input majority voter configured to vote on two external inputs and with the state of its current output. The output of a self-voter goes high when both its inputs are high and becomes low when both inputs are low. The output remains unchanged when inputs to the self-voter differ.

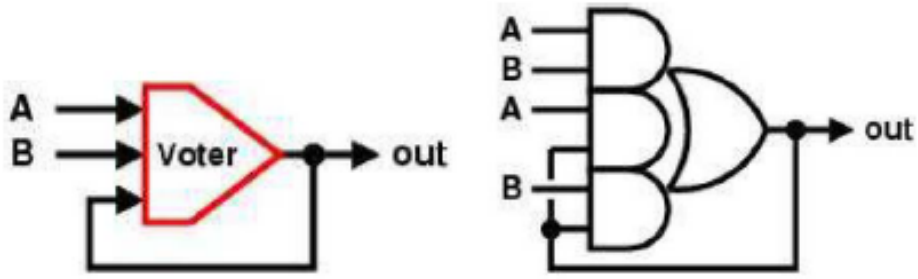


Figure 2.11: Self-voting majority circuit: schematic and standard-cell circuit [9].

The Bayesian detection technique from the communications theory has been applied to the voter in NMR, called soft NMR [10]. In comparison to NMR, voter in Soft NMR is composed of a detector as shown in Fig. 2.12. The underlying assumption in Soft NMR is that each processing element (PE) or redundant module is a noisy channel, and the detector acts as a slicer. In most cases, it is able to identify the correct output even when all duplicated modules are in error, but at the expense of very high area overhead cost of the soft voter.

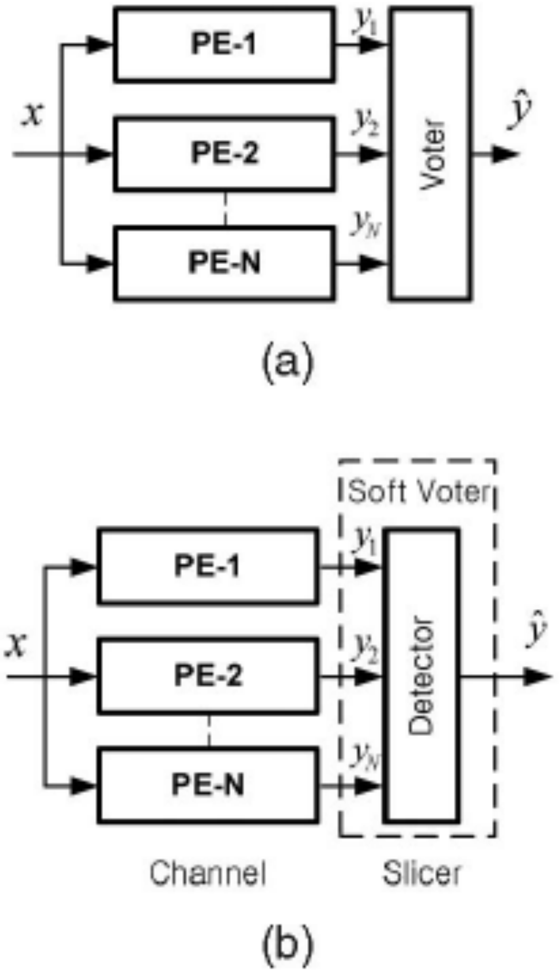


Figure 2.12: Block diagram of: (a) NMR and (b) Soft NMR [10].

Design diversity is a mechanism in which the redundancy is applied by implementing functionally equivalent but structurally different designs to potentially protect a circuit from multiple faults. Mitra et al. [49] proposed the use of design diversity mechanism in modular redundancy to detect common mode failures (CMFs). To quantify the diversity between redundant logic, various methods have been proposed. To understand the mechanism of design diversity, consider an example logic function shown in Fig. 2.13 which implements the logic function $Z = AB + AC$ in two different ways. The faults are mentioned as $f_1 = w$ Stuck-at-0 (written as w/0) in Fig. 2.13a and $f_2 = y$ Stuck-at-0 (written as y/0) in Fig. 2.13b. The input combinations $ABC = 111, 101$ and 110 will produce an error at Z_1. However, the only input combination that causes an error at Z_2 is $ABC = 101$. So, in a duplex system consisting of design from Fig. 2.13a and Fig. 2.13b, only an input pattern $ABC = 101$ will result in the system failure or results in the failure of both systems. Therefore, due to this input pattern, the fault goes undetected. If we consider that all input patterns are equally likely then the diversity value of duplex systems consisting of (f_1, f_2) will be $1 - \frac{1}{8} = \frac{7}{8}$. The applications of diversity design in DSP and communication systems is discussed by Reviriego et al. in [50].

Smith [51] proposed a DMR technique based on the use of a single-event transient (SET) suppressor circuit to each primary output of the circuit. The SET suppressor consists of two gates *(AND, OR)* and a simple two input multiplexer with its output connected to its own select line to select between $AND(OR)$ gate output when the combinational circuit primary output is logic 0(1). The

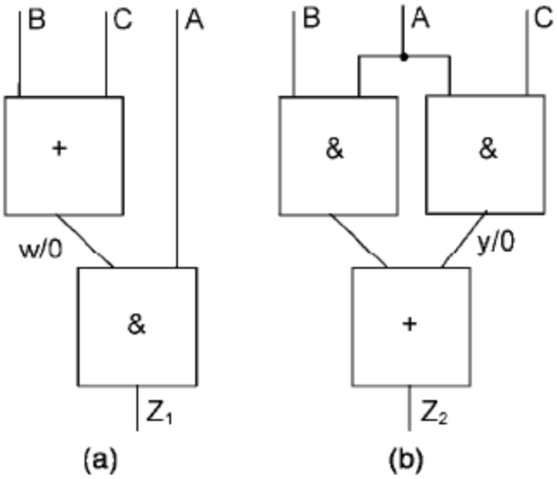


Figure 2.13: Example of design diversity.

$AND(OR)$ gates are fed by the outputs of the two functionally equivalent modules and are used to suppress SET from propagating to the primary output when the primary output is producing logic value 0(1).

A similar scheme has been proposed by Rezgui [52] to protect combinational and sequential circuit from SETs by utilizing C-Element [53]. The C-Element [54] is an asynchronous logic component which preserves the previous state of the logic circuit when the two modules produce different values shown in Fig. 4.5. The application of C-Element in modular redundancy is shown in Fig. 2.15.

Soft Errors in Sequential Circuits

El-Maleh et al. [55] proposed increasing sequential circuit reliability by introducing redundant equivalent states to states with high probability of occurrence in sequential circuit behavioral machine. To maintain the same operation of the unprotected FSM, the newly added redundant states have the same input, next state, and output as the original states. Other states with low probability are

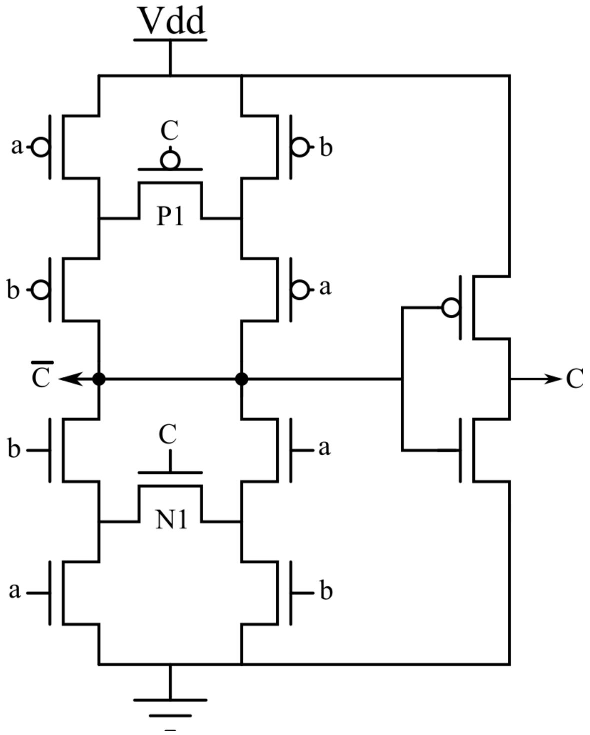


Figure 2.14: C-Element.

kept without protection to minimize the area overhead. The author divided the original states of the state machine into protected states with high probability of occurrence, and normal states with low probability of occurrence. For each protected state, equivalent redundant states are added to guarantee single soft fault tolerance. The author developed an algorithm to determine the number of bits needed to encode protected, redundant and normal states. The algorithm will also provide states codes. It was found that failure rate of sequential circuits which involves protecting states with high probability of occurrence is less than the ones involving protecting random or lower state probability.

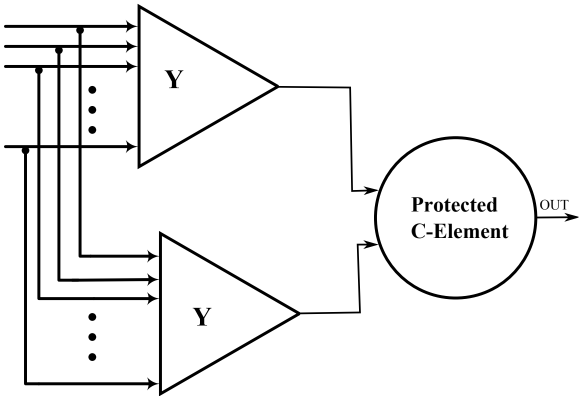


Figure 2.15: DMR with C-Element.

Defect-Tolerant Transistor Structures

A defect tolerant technique that adds redundancy at the transistor level of the circuit is proposed in [56]. The N^2 structure is a generalization of the quadded-transistor structure. In the quadded-transistor structure, each transistor, A, is replaced by a structure that implements either the logic function (AA) + (AA) or the logic function (A+A)(A+A). In such structure, any single transistor defect is tolerated. However, in the N^2 structure, N blocks are connected in series such that each block contains N parallel transistors. If number of defects is less than or equal to (N-1), N^2 structure guarantees the tolerance of all those defects. It was shown that this technique achieves higher defect tolerance compared to gate level based techniques such like quadded logic and TMR.

Soft error protection of combinational logic can be achieved by adding redundancy at the transistor-level. Nicolaidis [57] proposed a scheme where a circuit is duplicated containing all but last stage gate where the last stage gate is implemented as a code word preserving gate. This last stage gate is either a NOT, NAND or NOR gate with each transistor duplicated and connected in series to preserve the fault-free state that the output had before the transient fault occurred. More recently,

Implication Based Redundancy Addition for Fault Tolerance

Recently, there has been a growing interest to investigate the gate-level invariant relationships to solve multitude of CAD problems. Error detection and correction is one of the challenging problems. For memory elements it is quite simple as all that is required to check is the comparison of data being read and the data that is already stored. For memories it is usually employed using error correction codes (ECC). For logic circuits it is quite complicated to check for the errors as the correct answer is not known a priori. To circumvent this issue in logic circuits, Alves et al. [58–60] used logic implications for online error detection. If any invariant relation is violated the checker hardware will detect and signal an error. This resulted in very high fault coverage with very less area overhead. Use of logic implications for power optimization is proposed by Bahar et al. [61] in which high-power dissipating nodes are eliminated to reduce the switching activity. This results in the reduction of power dissipation of the entire circuit.

There has also been an interest to use logic implications for soft error tolerance. The mechanism is simple: if a SET occurs and distorts a signal, then the added functionally redundant wire will attenuate this effect from propagating to the primary output. Mukhaizim et at. [62] proposed a gate-level soft error mitigation technique by adding functionally redundant wires (FRWs). The proposed method first finds the implication relation between the sensitive wires-the wires that have high probability of fault detection-and all the other wires. The implication is added to the target gate if it reduces the soft error rate (SER) of the entire circuit. In order to find the implication that reduces/minimizes the sensitivity of the target gate, fault simulation has to be repeated for each found implication between the sources and a target. This results in significantly high computation time. Zhou et al. [63] proposed an improvement which considered electrical and timing window masking effect besides the logic masking. To eliminate the necessary fault simulation in order to quantify the value of an implication in [62], the authors in [63] employed a SER relation to do so. Additionally, the priority is given to the non-invert implication paths as the invert implication paths costs more hardware due to an extra inverter which introduces a new site for potential SET.

2.3.2 Synthesis-Based Fault Tolerance Techniques

In the synthesis-based techniques, the combinational circuit is restructured in order to maximize masking properties of the circuit. Logical masking is the main masking property to be maximized

Localized Circuit Restructuring Tolerant Technique

In [64], logic masking of errors is increased by taking the advantage of conditions already present in the circuit, such as observability don't cares. Two techniques are used to improve reliability: dont care-based re-synthesis and local rewriting. In the first method, high-impact nodes are identified. A node has high impact if many observable faults flow through it. High-impact nodes are used to select areas of the circuit for restructuring, in which a vulnerable node is replicated by adding a single gate. Local rewriting is also used to optimize small sub-circuits to obtain overall area improvements.

Reliability-Driven Dont Care Assignment Method

In [65], two algorithms are proposed to improve input error resilience. They focus on input error due to propagated failures from previous blocks. Both algorithms determine 0/1 assignments for the most critical Don't Care (DC) terms. Consider the correct input vector for a circuit is 0100, if a fault happens that fails the third input, the 0110 vector will be applied to the logic circuit. If the implementation is identical for these two vectors, then the error will be masked. If 0110 is a don't care, then the assignment of this minterm to either 0 or 1 will determine the masking of an error on the third input of the 0100 vector. Given a circuit with a set of don't care minterms, the output after applying proposed algorithms is the circuit with new on-set minterms, new off-set minterms and new don't cares set.

Redundancy Addition and Removal Technique

In [66], a framework is proposed based on redundancy addition and removal for soft error rate (SER) reduction. It performs a series of wire addition and removal by searching for redundant wires in the circuit. It will go through an iterative process trying to keep wires/gates with higher masking impact and to remove wires/gates with higher error impact; this will be guided using some metrics. The masking impact takes into account the three masking mechanisms.

Sub-circuit Extraction & Synthesis

El-Maleh et al. [2] proposed a scheme in which small sub-circuits are extracted from the original circuit, re-synthesized and merged back in the original circuit in order to maximize the probability of logical masking when a soft error occurs. Once each sub-circuit is extracted, the probabilities of its input vectors to occur are computed. Based on this input occurrence probability a new two-level circuit is produced which is then merged with the original circuit.

In [67], a circuit simplification method is proposed for error tolerant applications. In some applications such as images, video, audio and graphics many faulty versions of a chip can be used. In this work, the original combinational circuit is given with a defined error threshold, and the minimum area simplified circuit version is derived such that the error it produces is within the given threshold.

2.3.3 Physical Characteristics Based Fault Tolerance Techniques

The physical characteristics based techniques attempt to reduce SER based on the physical characteristics to maximize the electrical masking.

Many methods found in literature attempt to reduce SER based on the physical characteristics to maximize the electrical masking and latching-window masking. Gate resizing strategy [68] reduces SER by modifying the W/L ratios of transistors in gates. To achieve significant improvement in SER, potentially large overheads in area, delay, and power are introduced. In [69], a related method is introduced, which uses optimal assignments of gate sizes, threshold voltages, supply voltages, and output capacitive loads to get better results while keeping overheads smaller. Nevertheless, the design complexity is increased in this method in addition to the possibility of making circuit hard to optimize at physical design. Another scheme [70] focuses on the selection of flip-flop from a given set. It increases the probability of preventing faulty transients from being registered by selectively lengthening latching-windows associated with flip-flops, but it doesn't consider logical masking and electrical masking. A hybrid approach [71] combines flip-flop selection with gate resizing to achieve SER improvement.

Lazzari [1] proposed an asymmetric transistor sizing technique i.e., `nmos` and `pmos` transistors are sized independently of each other of the most sensitive gates of the circuit, but they considered that incident particles strike only the drain of transistors connected to the output of a gate. Sizing parallel transistors according to the sensitivity of their corresponding series transistors can significantly improve the fault tolerance of combinational circuits [72, 73]. Variable sizing among all transistors in a gate is a viable option if particle strikes of varying charge are considered. To further improve the fault tolerance, more up sizing quota is given to the most sensitive gates [74].

A more recent technique is presented in [75] that attempts to increase electrical masking. In this method the impact of using reliability-aware logic synthesis to reduce both the pulse width and the drain area of a circuit is analyzed. The idea here is to replace highly vulnerable cells with alternative cells or logical functions to reduce overall vulnerability of a circuit. The pulse width and drain area are used in this study as the reliability metrics to rank cells. The strategy is as follows: circuits are synthesized to a given cell library, then, the frequently used and highly vulnerable cells are identified; those identified cells are removed from library and are replaced with alternative implementations. Thus, an improved cell library is created.

To protect memories and latches from soft-errors, *cell hardening* techniques [76–78] have been used. An example of this approach is a DICE memory cell [76] that uses twice the number of original transistors (i.e., 12 transistors as compared to 6 transistors). The limitation of these approaches is that they are designed to tolerate soft errors in memory elements only and not in the combinational logic.

CHAPTER 3

SELECTIVE TRANSISTOR-REDUNDANCY BASED FAULT TOLERANCE TECHNIQUE FOR COMBINATIONAL CIRCUITS

This chapter is focused on designing combinational circuits for soft error tolerance with minimal area overhead. The idea is based on analyzing random pattern testability of faults in a circuit and protecting sensitive transistors, whose soft error detection probability is relatively high, until a desired circuit reliability is achieved or a given area overhead constraint is met. Transistors are protected based on duplicating and sizing a subset of transistors necessary for providing the protection. LGSynth'91 benchmark circuits are used to evaluate the proposed algorithm. Simulation results show that the proposed algorithm achieves better reliability than other transistor sizing based techniques and the Triple Modular Redundancy (TMR) technique with significantly lower area overhead for 130nm process technology at ground level.

It is observed that modular redundancy algorithms suffer from high area overhead as they either duplicate or triplicate the whole system followed by a voter. Even if the duplication or triplication is applied selectively, the area overhead is still high. On the other hand, symmetric transistor sizing approaches incur too much area cost, while asymmetric sizing approaches did't consider the transient hit at all possible locations. Asymmetric transistor sizing is applied to the most sensitive gates of the circuit by considering that particles can hit the drain of any transistor of a logic gate. A selective-transistor scaling method is proposed that protects individual sensitive transistors of a circuit. A sensitive transistor is a transistor whose soft error detection probability is relatively high. This is in contrast to previous approaches where all transistors, series transistors or those transistors connected to the output of a sensitive gate, whose soft error detection probability is relatively high, are protected. Transistor duplication and asymmetric transistor sizing is applied to protect the most sensitive transistors of the circuit. In asymmetric sizing, `nmos` and `pmos` transistors are sized independently. Reliability is evaluated for different protection thresholds and area overhead constraints.

The rest of the chapter is organized as follows: Section 3.1 highlights the motivation and rationale behind the proposed approach, Section 3.2 presents the proposed selective transistor-redundancy algorithm, an illustrative example is discussed in Section , simulation results are elaborated in Section 5.4, and finally the paper is concluded in Section 7.1.

3.1 Effect of Energetic Particle Strike

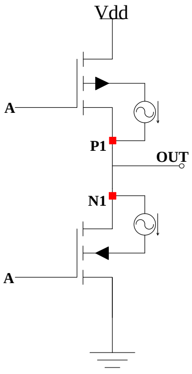
When an energetic particle strikes a semiconductor, it ionizes the region around it, resulting in the generation of electron-hole pairs. The charge due to the particle strike is then transported by drift and diffusion resulting in the establishment of transient electric field i.e., *SET*. The change in voltage observed at the output due to *SET* depends on the energy and angle of incidence of energetic particle. Source and drain regions are the most sensitive nodes to such events due to the large field around the junction regions which sweeps in the generated electron-holes and result in large currents. If the energy of a striking particle is high enough, it will flip the output of a gate resulting in a single event transient (SET) [3], [4].

To explain the selective transistor-redundancy principle, let us first consider the effect of an energetic particle striking a CMOS inverter. When the inverter input is LOW and the energetic particle strikes the drain of an `nmos` transistor, the output voltage is temporarily lowered. Whereas, when the inverter input is HIGH and the energetic particle strikes the drain of a `pmos` transistor, the output voltage is temporarily raised. In both cases, the output logic value of the inverter can be changed to a wrong value if enough charge is collected. This is shown in Fig. 3.1, using 130nm Predictive Technology Model (PTM) [79]. Fig. 3.1a illustrates the fault injection mechanism employed in this work. The output load is assumed to be equal to an inverter load. The soft error is modeled by injecting a current I of charge Q at the drain of a transistor. The direction of injected current is from *drain-to-body (bulk)* in the `nmos` transistor and from *body (bulk)-to-drain* in the `pmos` transistor. The double exponential current pulse I is used to model the charge deposited due to a particle strike at the drain of `nmos` or `pmos` transistor [80], [81] and is depicted as:

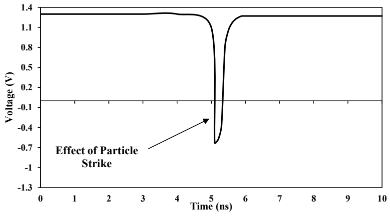
$$I(t) = \frac{Q}{(\tau_f - \tau_r)} \left(e^{-\frac{t}{\tau_f}} - e^{-\frac{t}{\tau_r}} \right) \tag{3.1}$$

Where Q is the charge deposited by a particle strike, τ_f denotes the falling time of the pulse, τ_r denotes the rising time of injected current pulse and vary for each process technology. The value of τ_f is greater than τ_r. The supply rail *VDD* is connected to 1.3V. We will be taking 130nm technology as a case study in this work, however, the technique is general and applicable to any process technology.

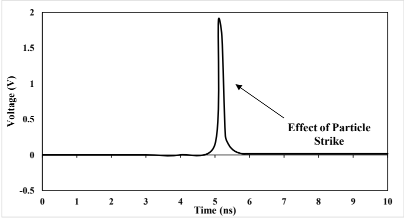
Fig. 3.1b illustrates the effect of a particle strike on the drain of an `nmos` transistor when the true output of an inverter is HIGH. The particle strike at N1 will cause a sudden drop in the output voltage (≈-0.7V) of inverter. This type of soft error will be modeled as a *stuck-at-0* (sa0) fault at the output of the gate. To protect from this fault, the `pmos` transistors of an inverter must be scaled enough so that the output voltage becomes > *VDD/2*. Fig. 3.1c illustrates the effect of a particle strike on the drain of a `pmos` transistor when the true output of an inverter



(a) Particle strike model.



(b) Effect of particle strike at `nmos` drain.



(c) Effect of particle strike at `pmos` drain.

Figure 3.1: Effect of energetic particle strike on CMOS inverter at $t = 5ns$.

is LOW. The particle strike at P1 will cause a sudden rise in the output voltage (≈1.9V) of inverter. This type of soft error will be modeled as a *stuck-at-1* (sa1) fault at the output of the gate. To protect from this fault, the `nmos` transistor of an inverter must be scaled enough so that the output voltage becomes < *VDD/2*.

Now, consider the transistors arrangement shown in Fig. 3.2a where duplicate `pmos` transistors are connected in parallel. The width of the redundant transistors must also be increased to allow dissipation (sinking) of the deposited charge as quickly as it is deposited so that the transient doesn't achieve sufficient magnitude and duration to propagate to the output. If the output is currently high and an energetic particle hits the drain N1 of the `nmos` transistor (with the same current source used in the simulations shown in Fig. 3.1), this should result in a lowered voltage observed at the output. But due to the employed transistor configuration, the net negative voltage effect will be compensated, as evident from Fig. 3.2b, resulting in a spike that has lesser magnitude as compared to the one shown in Fig. 3.1b. The spike magnitude is reduced due to increased output capacitance and reduced resistance between VDD and the output.

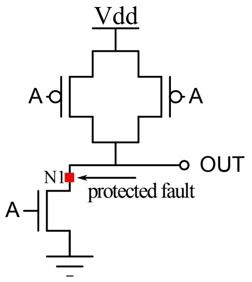
Consider another arrangement of transistors in Fig. 3.2c where redundant `nmos` transistors are connected in parallel. If the output is low and the incident energetic particle strikes the drain P1 of `pmos` transistor, then the raised voltage effect at the output shown in Fig. 3.1c will be reduced as observed from Fig. 3.2d. This reduction in the spike magnitude is due to the same reasons mentioned for the `nmos` transistor.

Similarly, to protect from both sa0 and sa1 faults, the transistor structures in Fig. 3.2a and Fig. 3.2c can be combined to fully protect a NOT gate. A fully protected NOT gate offers best hardening by design, but at the cost of higher area overhead and power. It must be noted that the optimal size of the transistor for SEU immunity depends on the charge Q of the incident energetic particle.

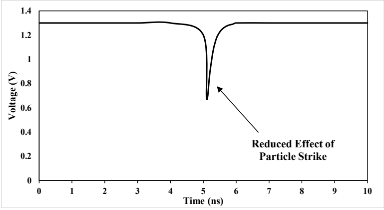
Due to aggressive nodes and voltage scaling, the effect of transient fault is no more constrained to a node where the incident particle strikes. This could result in the possibility of deposited charge being simultaneously shared by multiple circuit nodes in the circuit [25–27] leading to the *Single Event Multiple Upsets* or *SEMUs*, also referred to as *Multiple-Bit Upsets* or *MBUs* [3]. Consider the inverter example in Fig. 3.1, if two particles strike at the drain of `nmos` and `pmos` transistors simultaneously, then the charge collection at the `nmos` and `pmos` transistors will offset each other, resulting in an insignificant change in voltage at the output. Therefore, by duplication of transistors, it is intended to increase the probability of multiple fault hits at the same gate, so that the victim transistors could cancel the effect of each other. For that matter, LEAP [82] placement technique can be utilized. This scheme places the drain contact nodes of `nmos` and `pmos` transistors in an interleaved fashion so that multiple drain contact nodes can act together to fully or partially suppress the SETs. Another advantage of using parallel duplicate transistors is the defect tolerance of transistor stuck-open faults for protected transistors.

3.2 Proposed Algorithm

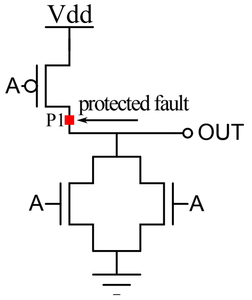
In this section, the proposed selective transistor-redundancy (STR) algorithm is presented. The algorithm protects sensitive transistors whose probability of failure (POF) is relatively high. The proposed algorithm can be utilized in two capacities: 1) apply protection until the POF of circuit reaches a certain threshold, 2) apply protection until certain area overhead constraint is met. We will first discuss different relations that realize the circuit POF. These relations are then used in the proposed algorithm.



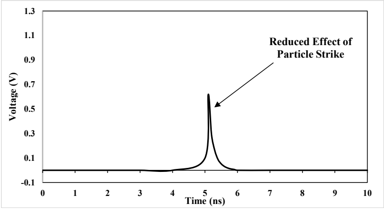
(a) Particle hit at `nmos` drain, OUT=HIGH.



(b) Reduced effect of particle strike at `nmos` drain.



(c) Particle hit at `pmos` drain, OUT=LOW.



(d) Reduced effect of particle strike at `pmos` drain.

Figure 3.2: Proposed protection schemes and their effect.

3.2.1 Circuit Probability of Failure

Let us first define the probability of failure of a transistor. In all discussions, subscripts i and j refer to gate i and transistor j, respectively. The POF_{ij} of j^{th} transistor of gate i is defined as the probability of circuit failure due to a fault hitting the transistor. It is computed using the following relation:

$$POF_{ij} = P_{DET_{ij}} \times P_{HIT_{ij}} \tag{3.2}$$

Where $P_{DET_{ij}}$ is the probability of detecting a fault hitting transistor j of gate i at a primary output, and $P_{HIT_{ij}}$ is the probability that transistor j of gate i is hit by a fault. The greater the transistor width/area is, the greater its hit probability is.

$P_{HIT_{ij}}$ is computed separately for `nmos` and `pmos` transistors as they have different drain widths. Let NW_{ij} and PW_{ij} be the width of `nmos` and `pmos` transistors, respectively, $Area$ be the total circuit area, then the probability of a transistor j of gate i to be hit by a fault, $P_{HIT_{ij}}$, is computed using the following relation:

$$P_{HIT_{ij}} = \frac{W_{ij}}{Area} \quad W_{ij} \in \{NW_{ij}, PW_{ij}\} \tag{3.3}$$

$P_{DET_{ij}}$, as defined before, depends on two factors; 1) probability of input patterns for which a fault that hits the transistor is propagated to the output of a gate i.e., controllability conditions to excite the fault, 2) stuck-at fault observability probability of the gate at one of the primary outputs of a circuit i.e., observability probability. $P_{DET_{ij}}$ is computed using following relation:

$$P_{DET_{ij}} = P_{Excitation_{ij}} \times P_{Propagation_{ij}} \tag{3.4}$$

Where $P_{Excitation_{ij}}$ denotes the probability that the fault is excited at gate i

output due to a fault hit at transistor j. $P_{Propagation_{ij}}$ denotes the probability that an error that is excited at the gate's output is observable at one of the primary outputs. Let $\mathbf{S}$ be a set of patterns for which an error that strikes transistor j is propagated to the output of gate i, then $P_{Excitation_{ij}}$ is computed as:

$$P_{Excitation_{ij}} = \sum_{k=1}^{|\mathbf{S}|} Prob.\ \mathbf{S}_k \tag{3.5}$$

Where $Prob.\ \mathbf{S}_k$ denotes the probability of occurrence of k^{th} input pattern. SPICE tool is used to find the input patterns for which a transistor fault is excited and observed at the gate output.

Similarly, $P_{Propagation_{ij}}$ can be computed using the following relation:

$$P_{Propagation_{ij}} = \frac{stuck - at - detection - prob_i}{PC_i} \tag{3.6}$$

Where $stuck - at - detection - prob_i$ defines stuck-at fault detection probability of gate i and PC_i is the controllability probability to produce logic value opposite to the fault effect at the gate output. The fault simulator tool HOPE [83] is used to compute the *stuck-at* fault detection probability and PC_i of a gate i.

Finally, the circuit probability of failure $\mathbf{POF_C}$ *for a single fault* is simply the summation of POFs of all transistors n over all gates m of a circuit.

$$\mathbf{POF_C} = \sum_{i=1}^{m} \sum_{j=1}^{n} POF_{ij} \tag{3.7}$$

Table 3.1: Parameters considered in the study.

Technology (T)	$130m = 0.13\mu$
nMOS width (NW_{ij})	$2 \times T = 0.26\mu$
pMOS width (PW_{ij})	$4 \times T = 0.52\mu$
Charge (Q)	0.3pC

3.2.2 Example: NAND Gate

A thorough case study for the $130nm$ process technology is performed to elaborate POF_{ij}. Here, we will consider the case of a 2-input NAND gate. The basic process related parameters used in this study employ minimum feature size and are shown in Table 3.1. It must be noted that in practical designs the minimum widths of the transistors are adjusted to cater for the specifications of desired application. The value of charge Q considered here is $0.3pC$ which is the worst case charge value deposited by the 130nm process technology [68].

For NAND gates, the sa0 fault excitation probability is computed for input patterns where the output is logic “1”. For a 2-input NAND gate, there is a maximum of four input combinations, $\{00, 01, 10, 11\}$. Therefore, the sa0 excitation probability of the j^{th} `nmos` transistor of gate i is computed based on the input combinations producing logic value “1” at the output i.e., $\{00, 01, 10\}$.

Fig. 3.3 shows the CMOS structure of a 2-input NAND gate. A transient with charge (Q) of $0.3pC$ injected at drain “N1” will always be propagated to the output of the gate for input patterns $\{00, 01, 10\}$. Therefore, the fault excitation probability for `nmos` transistor connected to input “A” in Fig. 3.3, computed using Eqn. 3.5, is $(\frac{3}{4})$ or 0.75. So, when a fault hits the `nmos` transistor “N1” of a 2-input NAND gate, it will fail with a probability of $(\frac{3}{4})$. This is because for 3 out of the

4 possible input combinations, the gate will fail.

Logically, the transient hit at “N2” should only be excited to the output if and only if the value of input “A” is logic “1” i.e., for input pattern 10, implying fault excitation probability of $(\frac{1}{4})$ or 0.25 . However, since the fault excitation is highly dependent on the transient charge value, it is not necessarily true that the fault injected at “N2” will not be excited for the input patterns $\{00, 01\}$. To overcome the uncertainty regarding the fault excitation under all possible input combinations, transistor level simulation using SPICE is performed. Based on SPICE simulation, the transient fault injected at drain “N2” with charge (Q) value of $0.3pC$ is observed at the output for input pattern $\{10\}$, only. Therefore, the fault excitation probability for `nmos` transistor “N2” connected to input “B” is $(\frac{1}{4})$ or 0.25. Here, $stuck-at-detection-prob_i = PC_i$, because the NAND gate is a stand alone gate and is not connected to any other gate. Therefore, any fault excited due to a fault hit at the transistor will make the gate/circuit fail. So, $P_{DET_{N1}} = 0.75$ and $P_{DET_{N2}} = 0.25$.

Fig. 3.4 illustrates the sa1 scenario when a fault hits any of the `pmos` transistors P1 or P2 of a 2-input NAND gate. In this case, the fault will be observed at the output for input pattern $\{11\}$ only. Again, $stuck-at-detection-prob_i = PC_i$. Thus, the fault excitation probability due to a fault hit at `pmos` transistors “P1” or “P2” is $(\frac{1}{4})$ or 0.25. This is because the gate will fail for 1 out of 4 possible input combinations. So, $P_{DET_{P1}} = P_{DET_{P2}} = 0.25$.

The gate failure probability of a 2-input NAND gate can be computed using

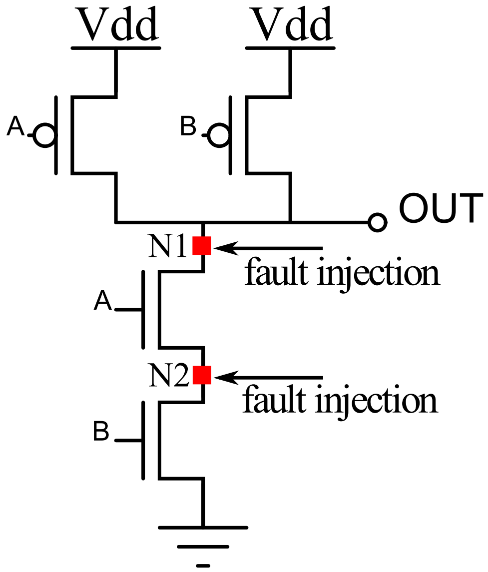


Figure 3.3: Stuck-at-0 case of 2-input NAND gate.

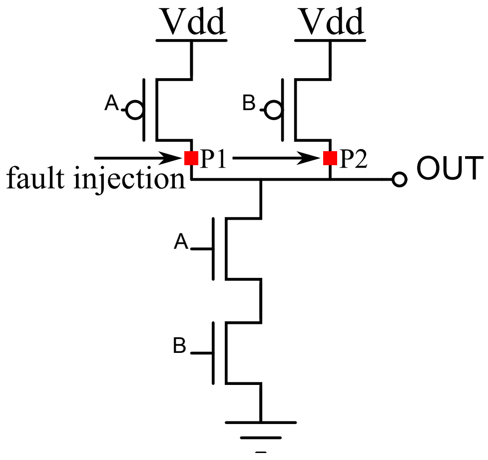


Figure 3.4: Stuck-at-1 case of 2-input NAND gate.

Eqn. (3.7) as follows:

$$\begin{aligned}
Gate\ Failure\ Prob. &= \sum_{j=1}^{2} P_{DET_{Nij}} \times \frac{NW_{ij}}{Area} \\
&\quad + \sum_{j=1}^{2} P_{DET_{Pij}} \times \frac{PW_{ij}}{Area} \\
&= \overbrace{\left(\frac{0.75 \times 0.26 + 0.25 \times 0.26}{1.560}\right)}^{N1,N2} \\
&\quad + \overbrace{\left(2 \times 0.25 \times \frac{0.52}{1.560}\right)}^{P1,P2} \\
&= 0.167 + 0.167 \\
&= 0.334
\end{aligned}$$

3.2.3 Selective Transistor-Redundancy (STR) Based Design

The selective redundancy technique is applied to protect transistors of a circuit that have relatively high POF_{ij}. Sensitive transistors that have relatively high probability of failure are identified based on fault simulation of random input patterns. Different arrangements of `nmos` and `pmos` transistors are proposed for each gate for various transistor protection scenarios.

Algorithm 1 highlights the steps of the proposed method. Initially, POF of circuit under test is computed using Eqn. (3.7) by first computing the POF of each transistor using Eqn. (3.2). The proposed algorithm applies transistor protection until the circuit POF reaches a pre-defined protection threshold, or a certain area overhead constraint is met. Each time, the algorithm selects a transistor with the highest POF. The effect of a transient fault on the selected `nmos(pmos)` transistor is suppressed or reduced by duplicating and scaling the widths of a subset of transistors necessary for providing the protection. For example, in a 2-input NAND gate, protecting an `nmos` transistor requires duplicating and scaling its corresponding `pmos` transistor connected to the same input. However, protecting a `pmos` transistor requires duplicating and scaling both of the `nmos` transistors in the gate. Once a transistor is protected, the POF of all transistors in the circuit are updated. Protecting a transistor in a gate g_i affects the selection/hit probability of all transistors in the circuit. Therefore, after protecting a transistor in a gate, the POF of the selected transistor is reduced significantly, while the POF of the remaining transistors may increase or reduce slightly. The circuit area, POF of all transistors, and $\mathbf{POF_C}$ are updated after each transistor protection is applied. The transistor with maximum POF_{ij} is selected for protection in the next iteration. The process is repeated until the desired protection threshold is reached or the maximum area overhead constrain is met.

The protection threshold Th takes the value between [0%, 100%] and represents the reliability of the circuit required to be achieved. For example, a protection threshold of 99% implies applying the protection until POF of circuit is less

Algorithm 1 : Selective Transistor-Redundancy Algorithm

```
Require: Gate level circuit, $Th$ or $OverHead$
$Th$ : Required circuit reliability in %
$OverHead$ : Required area overhead in %
$POF_{ij}$ : Circuit POF due to fault hit at $j^{th}$ transistor of Gate $i$
$\mathbf{POF_C}$ : Circuit probability of failure

Compute random pattern fault detection probability of each gate $g_i$ using fault simulator
For all transistors compute $POF_{ij}$ using Eqn. (3.2)
Compute $\mathbf{POF_C}$ using Eqn. (3.7)
TargetArea = CircuitArea + (CircuitArea × OverHead)
while $((\mathbf{POF_C} \geq (1 - Th)) or (CircuiArea < TargetArea))$ do
    Pick a transistor $trans_{ij}$ with the highest $POF_{ij}$
    Protect $trans_{ij}$
    Update CircuitArea
    Update $POF_{ij}$ of transistors using Eqn. (3.2)
    Update $\mathbf{POF_C}$ using Eqn. (3.7)
end while
```

than or equal to $(1 - 99\%) = 0.01$. Increasing Th will result in more transistors being protected and vice versa.

3.2.4 Redundancy Models

In light of Algorithm 1, let's now explain the proposed CMOS implementations of a 2-input NAND gate in Table 3.2. In this work, a library consisting of 2-, 3- and 4-input NAND/NOR gates and an Inverter is considered. For brevity, the case of a 2-input NAND gate will be discussed.

The proposed CMOS implementations of a 2-input NAND gate are shown in Table 3.2. The first row shows the names of different implementations of a 2-input NAND gate, while the second row shows their corresponding implementations at the transistor level. The first numeric value "2" in the gate name (e.g. NAND21) denotes the number of inputs of a gate and the second numeric value, which ranges from 1 to 5, is used to select the proper transistor level implementation of a 2-input NAND gate.

In CMOS implementation of NAND21, `pmos` transistor P1 is duplicated, scaled and connected in parallel to protect a fault that hits the drain of `nmos` transistor N1. Similarly, to protect `nmos` transistors N1 and N2, `pmos` transistors P1 and P2 are duplicated and scaled to protect from a fault that can occur at the drain of any of the `nmos` transistors. Hence, the protection type for that gate will be NAND22.

To protect from faults hitting the drain of `pmos` transistors, all the `nmos` transistors are required to be scaled and duplicated. This is due to the fact that `pmos` transistors P1 and P2 are in parallel which makes them equally sensitive to a fault. This implementation is called NAND23. NAND24 provides protection from faults that can occur at any of the `pmos` transistors or at the `nmos` transistor N1. Finally, to fully protect the 2-input NAND gate, all the transistors are duplicated along with their widths increased. This type of protection is called NAND25. So, for a 2-input NAND gate, there are 5 distinct redundancy models.

For a 2-input NOR gate, similar arrangements can be used to create its redundancy model. For 3- and 4-input NAND/NOR gates, 7 and 9 redundancy models are created, respectively. The necessity to create a variety of redundancy models for every possible scenario is to achieve as much area savings as possible. The number of redundancy models is technology independent and allows the proposed

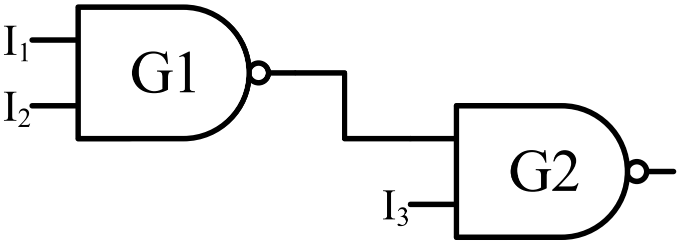


Figure 3.5: Example circuit.

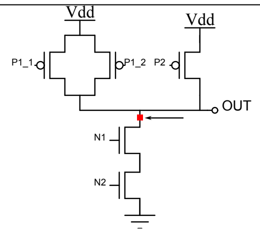
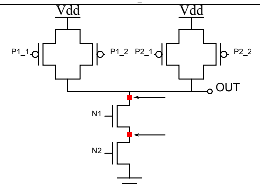
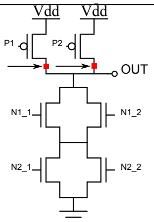
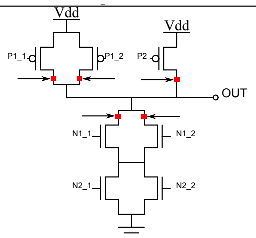
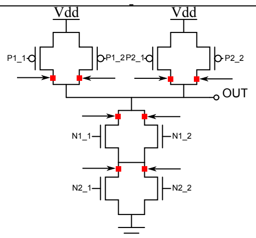
algorithm to improve the reliability of circuit by applying fine grained protection (protecting one transistor at a time) instead of protecting the whole gate at once as has been proposed by other techniques [1,68]. It will be shown that due to this fine granularity of protection, the area overhead can be significantly reduced.

3.3 An Illustrative Example

Let us now consider an illustrative example to elaborate the application of the proposed selective transistor redundancy scheme given in Algorithm 1. Fig. 3.5 shows a simple benchmark circuit consisting of 3 inputs, 1 output and a pair of 2-input NAND gates. First, let's compute all the required parameters as discussed in previous sections. Table 3.3 and Table 3.4 show the fault detection probabilities, controllability probabilities (PC0, PC1) and input pattern probabilities (`ipp`) of all gates in the circuit.

Let's first compute the POF of the circuit using Eqn. (3.7). With $NW_{ij} = 0.26\mu$ and $PW_{ij} = 0.52\mu$, the total area of the circuit is 3.12μ, which is the summation of drain areas of all the transistors. So, initially for `nmos` transistors, $P_{HIT_N} = \frac{0.26}{3.12} = 0.083$ and for `pmos` transistors, $P_{HIT_P} = \frac{0.52}{3.12} = 0.166$.

Table 3.2: Proposed CMOS implementations of 2-input NAND gate.

Gate	CMOS [1]
	
	
	
	
	

[1] Arrows indicate fault hit at the transistor that is protected.

Table 3.3: Stuck-at fault detection probabilities.

Gate	**sa0 det. prob.**	**sa1 det. prob.**	**PC0**	**PC1**
G1	0.375	0.125	0.25	0.75
G2	0.625	0.375	0.375	0.625

Table 3.4: `ipp` at G1 and G2.

Input	**G1**	**G2**
00	0.25	0.125
01	0.25	0.125
10	0.25	0.375
11	0.25	0.375

Keeping Table 3.2 in perspective, transistor G2-N1 refers to the `nmos` transistor directly connected to the output of gate G2, G2-N2 refers to the `nmos` transistor in gate G2 with drain connected to N1 and source connected to ground. With $\sum_{k=1}^{|\mathbf{S}|} Prob.\ \mathbf{S}_k = Prob_{00} + Prob_{01} + Prob_{10} = 0.125 + 0.125 + 0.375 = 0.625$, sa0 detection prob. $= 0.625$, $PC_1 = 0.625$, POF_{G2-N1} is computed using Eqn. (3.2) as follows:

$$\begin{aligned} POF_{G2-N1} &= \frac{0.625 \times 0.625}{0.625} \times \frac{0.26}{3.12} \\ &= 0.625 \times 0.083 \\ &= 0.0521 \end{aligned}$$

For G2-N2, $\sum_{k=1}^{|\mathbf{S}|} Prob.\ \mathbf{S}_k = Prob_{10} = 0.375$. Therefore, POF_{G2-N2} will be:

$$\begin{aligned} POF_{G2-N2} &= \frac{0.375 \times 0.625}{0.625} \times \frac{0.26}{3.12} \\ &= 0.375 \times 0.083 \end{aligned}$$

$$= \quad 0.031$$

For G2-P1 & G2-P2, $\sum_{k=1}^{|\mathbf{S}|} Prob.\ \mathbf{S}_k = Prob_{11} = 0.375$, sa1 detection prob. $=$ 0.375 and $PC_0 = 0.375$. POF_{G2-P1} and POF_{G2-P2} will be:

$$\begin{aligned} POF_{G2-P1,G2-P2} &= \frac{0.375 \times 0.375}{0.375} \times \frac{0.52}{3.12} \\ &= 0.375 \times 0.167 \\ &= 0.063 \end{aligned}$$

Similarly, POF of all transistors in gate G1 are:

$$\begin{aligned} POF_{G1-N1} &= \frac{0.75 \times 0.375}{0.75} \times \frac{0.26}{3.12} \\ &= 0.375 \times 0.083 \\ &= 0.0311 \\ POF_{G1-N2} &= \frac{0.25 \times 0.375}{0.75} \times \frac{0.26}{3.12} \\ &= 0.125 \times 0.083 \\ &= 0.0104 \\ POF_{G1-P1} &= \frac{0.25 \times 0.125}{0.25} \times \frac{0.52}{3.12} \end{aligned}$$

$$
\begin{aligned}
&= 0.125 \times 0.167 \\
&= 0.0208 \\
POF_{G1-P2} &= \frac{0.25 \times 0.125}{0.25} \times \frac{0.52}{3.12} \\
&= 0.125 \times 0.167 \\
&= 0.0208
\end{aligned}
$$

Finally, POF of circuit, obtained by summing POF of all transistors, is 0.2915 or 29.15%.

Let's now follow few iterations of Algorithm 1 to see how the protection is applied to the circuit. It can be observed that G2-P1 and G2-P2 have the highest POF, therefore, a fault on G2-P1 and G2-P2 is protected by duplicating and increasing the widths of all the `nmos` transistors, as explained in Section 3.2.4. Such kind of protection is referred to as NAND23, where both `pmos` transistors are protected. Protecting G2-P1 and G2-P2 will lead to $\sum_{k=1}^{|\mathbf{S}|} Prob.\, \mathbf{S}_k = 0$ for both G2-P1 and G2-P2 and ultimately $POF_{G2-P1} = 0$ and $POF_{G2-P2} = 0$. The area of `nmos` transistors G2-N1 and G2-N2 will increase from 0.26 to 1.612, as each transistor is duplicated and scaled by a factor of 3.1, implying an area overhead of 1.352 and a circuit area of 5.824. The resulting POF of transistors after protecting $G2-P1$ & $G2-P2$ will be:

$$
\begin{aligned}
POF_{G2-N1} &= \frac{0.625 \times 0.625}{0.625} \times \frac{1.612}{5.824} \\
&= 0.625 \times 0.276 \\
&= 0.173 \\
POF_{G2-N2} &= \frac{0.375 \times 0.625}{0.625} \times \frac{1.612}{5.824} \\
&= 0.375 \times 0.3239 \\
&= 0.104 \\
POF_{G1-N1} &= \frac{0.75 \times 0.375}{0.75} \times \frac{0.26}{5.824} \\
&= 0.375 \times 0.045 \\
&= 0.017 \\
POF_{G1-N2} &= \frac{0.25 \times 0.375}{0.75} \times \frac{0.26}{5.824} \\
&= 0.125 \times 0.045 \\
&= 0.0056 \\
POF_{G1-P1} &= \frac{0.25 \times 0.125}{0.25} \times \frac{0.52}{5.824} \\
&= 0.125 \times 0.089 \\
&= 0.0111 \\
POF_{G1-P2} &= \frac{0.25 \times 0.125}{0.25} \times \frac{0.52}{5.824} \\
&= 0.125 \times 0.089 \\
&= 0.0111
\end{aligned}
$$

For transistors other than $G2 - P1$ and $G2 - P2$, only the hit rate at the transistors will change as the circuit area is increased. The circuit POF after protecting $G2 - P1$ and $G2 - P2$ is increased from 0.2915 to 0.3584 with gate G2 protected as NAND23. The increase in circuit POF is due to the fact that protecting `pmos` transistors of a NAND gate requires all the `nmos` transistors to be scaled and duplicated, which makes `nmos` transistors more sensitive to the particle strike due to increase in their area.

For the next iteration, $G2 - N1$ has the highest POF and is selected for protection by duplicating and scaling the $P1$ `pmos` transistor of gate G2. The required scaling factor is 2.4, computed using SPICE. The resulting POF of the circuit is decreased from 0.3584 to 0.1895 with a circuit area of 7.8. The POF of remaining transistors is computed to be:

$$\begin{aligned}
POF_{G2-N1} &= \frac{0.375 \times 0.625}{0.625} \times \frac{1.612}{7.8} \\
&= 0.375 \times 0.206 \\
&= 0.077 \\
POF_{G2-N2} &= \frac{0.375 \times 0.625}{0.625} \times \frac{1.612}{7.8} \\
&= 0.375 \times 0.206 \\
&= 0.077 \\
POF_{G1-N1} &= \frac{0.75 \times 0.375}{0.75} \times \frac{0.26}{7.8}
\end{aligned}$$

$$
\begin{aligned}
&= 0.375 \times 0.033 \\
&= 0.0124 \\
POF_{G1-N2} &= \frac{0.25 \times 0.375}{0.75} \times \frac{0.26}{7.8} \\
&= 0.125 \times 0.033 \\
&= 0.0041 \\
POF_{G1-P1} &= \frac{0.25 \times 0.125}{0.25} \times \frac{0.52}{7.8} \\
&= 0.125 \times 0.066 \\
&= 0.0083 \\
POF_{G1-P2} &= \frac{0.25 \times 0.125}{0.25} \times \frac{0.52}{7.8} \\
&= 0.125 \times 0.066 \\
&= 0.0083
\end{aligned}
$$

The gate G2 is now marked as NAND24. The POF of $G2 - N1$ is still not 0 as there remains an input pattern 10 for which it is not protected. For the next iteration, $G2 - N2$ is selected. Protecting $G2 - N2$ will make G2 marked as NAND25 i.e., fully protected against a single fault, with POF of $G2 - N1$ and $G2 - N2$ equal to 0 and the POF of circuit reduced to 0.0265 with a circuit area of 9.7760. Now, only gate G1 remains for protection.

If the desired circuit POF to achieve is 1% or alternatively 99% reliability and continuing with applying Algorithm 1, both G1 and G2 will be marked as

0. Initial POF_C = 0.2915, Area (A) = 3.120
1. POF_C = 0.3584, A = 5.82, Tr = G2-P1,G2-P2
2. POF_C = 0.1895, A = 7.80, Tr = G2-N1
3. POF_C = 0.0265, A = 9.77, Tr = G2-N2
4. POF_C = 0.0378, A = 11.75, Tr = G1-N1
5. POF_C = 0.0275, A = 14.45, Tr = G1-P2,G1-P2
6. POF_C = 0, A = 16.43, Tr = G1-N2

Figure 3.6: Complete iteration log of example circuit.

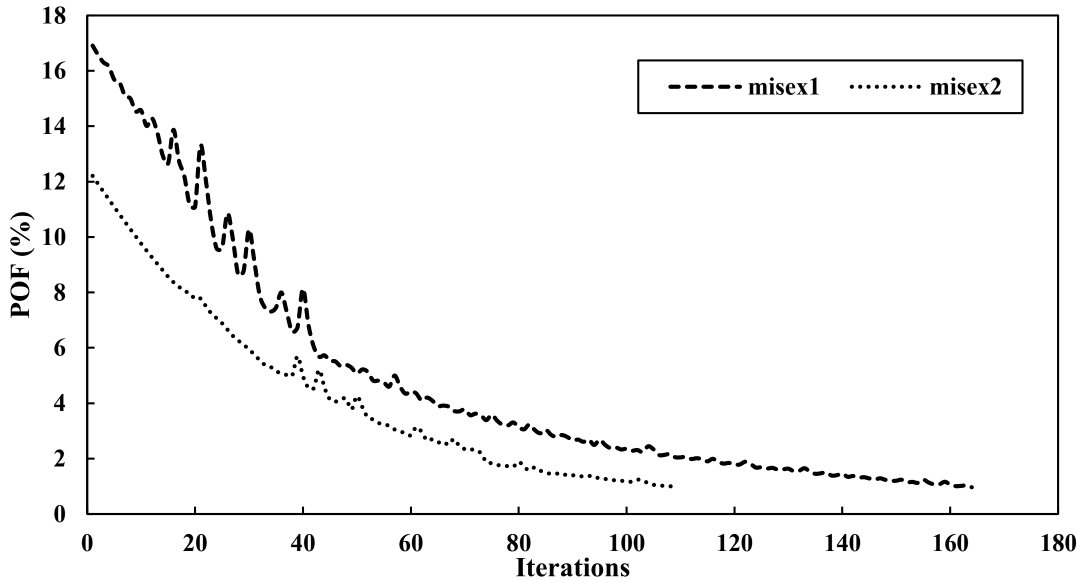


Figure 3.7: Circuit POF vs. iteration.

NAND25 i.e., fully protected to achieve 100% reliability against a single fault. The complete iterations are shown in Fig. 3.6 and include the circuit POF (POF_C), circuit Area (A) and the transistor (Tr) selected for protection in each iteration of the algorithm.

Fig. 3.7 shows the POF of two circuits plotted over the duration of achieving 1% POF (99% reliability). It can be seen that $misex1$ requires around 165 iterations, whereas $misex2$ requires around 110 iterations of the algorithm to achieve 99% reliability against a single fault.

3.4 Experimental Results

In this section, the impact of the proposed algorithm on the area and reliability of LGSynth'91 benchmarks [84] is evaluated. The benchmarks consist of circuits with varying complexity in terms of area, number of inputs and outputs. Sensitive nodes (transistors/gates) in a circuit are identified based on the fault simulation of random input vectors using the parallel fault simulator Hope [83]. The input patterns are applied until stuck-at fault coverage of 95% is achieved. It was found that 1 million random input patterns achieved more than 95% stuck-at fault coverage for all benchmarks in this work.

Whenever cell hardening against soft errors is considered, the first step is to select a range of particles energy against which the tolerance is sought. In this work, it is assumed that energy of the incident particle will always result in the maximum deposition of charge. For that matter and to compare with other sizing techniques, the values of $Q = 0.3pC$, $\tau_f = 0.2ns$ and $\tau_r = 0.05ns$ are used for all simulations in this paper. The value of charge $Q = 0.3pC$ is the maximum charge that could be collected by the 130nm process technology [68]. The simulations are performed for varying protection thresholds to find the best tradeoff between area and reliability for each circuit. The number of faults injected in a protected circuit is prorated according to its area overhead. Algorithm 9 is used to compute the reliability of each circuit. The number of simulations count SIM is 5000 iterations for each fault injection scenario.

The LGSynth'91 benchmark circuits used in this work are represented in two-level `pla` formats, therefore, they are synthesized with single output optimization using *Espresso* [85] tool and then mapped to 130nm technology using *SIS* [86] to get the proper gate level representation of the circuit. The library used for mapping consists of an Inverter and 2-, 3- and 4-input NAND and NOR gates. The parameter `phase` in the logic synthesis process defines whether the output function should be synthesized as an ON-set (`phase=1`) or an OFF-set (`phase=0`). By default, each output is synthesized as an ON-set by the *Espresso* tool. Each output is synthesized by synthesizing the phase with higher probability i.e., if the output probability of 1 is higher than the probability of 0, then the value of (`phase=1`) is set, otherwise, (`phase=0`) is set. This produces circuits with higher reliability as shown in Table 3.5.

In Table 3.5, the first column denotes the circuit names along with the number of inputs and outputs in each circuit. The second and third major columns report the reliability of circuits using default synthesis settings and the proposed majority phase synthesis mechanism. The reliability of circuits is evaluated against 1, 2 and 5 faults, respectively. The *Area* of a benchmark is computed by summing the drain area of all the `nmos` and `pmos` transistors.

Table 3.5: Reliability of original benchmark circuits.

Circuit	**True Phase Synthesis**				**Majority Phase Synthesis**				**ΔArea[2]**
(Inputs, Outputs)	**Area1 (μ)[1]**	**1 Fault**	**2 Faults**	**5 Faults**	**Original Area**	**1 Fault**	**2 Faults**	**5 Faults**	
alu4 (14, 8)	1831.44	96.78%	92.86%	83.44%	1429.74	97.89%	95.86%	87.44%	−21.93%
apex1 (45, 45)	4282.2	96.80%	92.72%	82.64%	4602.00	96.72%	94.20%	86.40%	7.47%
apex2 (39, 3)	804.18	99.10%	97.38%	94.54%	609.96	99.20%	98.04%	95.42%	−24.15%
apex3 (54, 50)	3091.92	96.00%	93.28%	84.12%	3025.62	96.88%	94.76%	85.66%	−2.14%
apex4 (9, 19)	4754.1	95.78%	92.32%	82.30%	4575.48	96.20%	92.74%	84.16%	−3.76%
b12 (15, 9)	130.26	88.36%	78.18%	54.28%	121.68	89.10%	78.22%	55.30%	−6.59%
clip (9, 5)	372.84	93.24%	86.40%	71.44%	372.84	93.24%	86.40%	71.44%	0 %
cordic (23, 2)	238.68	98.02%	96.24%	90.54%	241.02	98.10%	96.28%	92.14%	0.98%
ex5 (8, 63)	948.48	93.28%	88.64%	70.70%	977.34	93.50%	88.26%	71.34%	3.04%
misex1 (8, 7)	139.62	83.40%	68.92%	41.74%	152.88	84.34%	71.70%	48.44%	9.50%
misex2 (25, 18)	225.42	93.36%	88.02%	71.80%	230.88	93.20%	88.32%	69.48%	2.42%
misex3 (14, 14)	2410.98	97.48%	95.56%	88.58%	1886.82	97.64%	95.40%	88.90%	−21.74%
rd84 (8, 4)	368.94	91.36%	84.94%	65.18%	496.08	93.38%	87.22%	71.40%	34.46%
seq (41, 35)	4693.26	98.92%	98.26%	95.96%	4970.94	99.05%	98.22%	95.74%	5.92%
squar5 (5, 8)	111.54	82.50%	70.04%	41.04%	101.40	83.44%	69.46%	42.68%	−9.09%
table3 (14, 14)	3073.2	98.12%	95.52%	88.54%	3475.68	98.68%	97.78%	94.28%	13.10%
table5 (17, 15)	3230.76	97.48%	96.00%	88.52%	3535.74	98.70%	97.72%	94.52%	9.44%
z5xp1 (7, 10)	248.82	85.96%	75.38%	49.36%	251.16	87.96%	75.38%	52.36%	0.94%
Avg.		**93.66%**	**88.37%**	**74.71%**		**94.29%**	**89.22%**	**77.06%**	**-0.12%**

[1] Summation of `nmos` and `pmos` drain widths

[2] $\Delta\text{Area} = \left(\frac{Original\ Area - Area1}{Area1}\right) \times 100$

Table 3.5 highlights the increase in reliability when the circuits are synthesized w.r.t the majority phase. This is due to the increase in fault masking that may occur as if the final gate is an OR gate and has a value of “1”, any fault propagating through any of the other inputs will be masked. If the OR gate has at least two inputs with a value of logic “1”, all faults propagating through any of the inputs will be masked. Thus, synthesizing the circuit to maximize the probability of getting a logic 1 at the final gate by synthesizing the majority phase will maximize the probability of fault masking and hence improve reliability. Therefore, circuits synthesized with the majority phase are the baseline circuits used in all simulations in this work. It can be observed that for few benchmarks, reliability is above 90% for all fault injection scenarios. These benchmarks promise great reliability improvement with slight area overhead.

Table 3.6 shows the results of applying Algorithm 1 on the benchmark circuits and highlights the reliability of circuits for varying protection thresholds. A protection threshold of 98% implies that the circuit POF must be less than or equal to $(1 - 98\%) = 0.02$. Therefore, the applied protection threshold highly correlates with the reliability achieved by the circuit for a single fault. In Table 3.6 under the 99% column, the minimum area overhead required for $ex5$ circuit to achieve a reliability greater than or equal to 99% against a single fault is $\approx 140\%$. For $alu4$, $apex1$, $apex2$, $apex3$, $apex4$, $cordic$, $misex3$, seq, $table3$ and $table5$ benchmarks under the 95% column in Table 3.6, zero area overhead implies that these benchmarks achieve 95% reliability against single fault without any area

Table 3.6: Reliability of circuits based on proposed STR technique with varying protection thresholds against a single fault.

Circuit	**95%**		**98%**		**99%**	
	OH[1]	**Rel**	**OH**	**Rel**	**OH**	**Rel**
alu4	0%	97.89%	20.04%	98.44%	51.84%	99.02%
apex1	0%	96.72%	12.64%	98.10%	48.81%	99.10%
apex2	0%	99.20%	0%	99.20%	0%	99.20%
apex3	0%	96.88%	13.93%	98.05%	51.57%	99.08%
apex4	0%	96.20%	25.88%	98.30%	71.88%	99.02%
b12	86.50%	95.01%	149.02%	98.03%	197.69%	99.05%
clip	14.64%	95.00%	68.61%	98.11%	119.87%	99.08%
cordic	0%	98.10%	0%	98.10%	19.53%	99.00%
ex5	19.53%	95.62%	81.73%	98.12%	139.89%	99.02%
misex1	85.68%	95.07%	175.48%	98.01%	249.35%	99.02%
misex2	14.12%	95.24%	50.29%	98.36%	92.48%	99.20%
misex3	0%	97.64%	3.33%	98.10%	34.27%	99.18%
rd84	14.16%	95.10%	56.49%	98.02%	101.21%	99.24%
seq	0%	99.05%	0%	99.05%	0%	99.05%
squar5	107.79%	95.10%	206.15%	98.06%	286.51%	99.00%
table3	0%	98.68%	0%	98.68%	0.34%	99.24%
table5	0%	98.70%	0%	98.70%	1.70%	99.32%
z5xp1	68.22%	95.06%	143.69%	98.11%	202.88%	99.06%
Avg.	**22.81%**	**96.68%**	**55.96%**	**98.31%**	**92.77%**	**99.11%**

[1] Area overhead (OH) = $\left(\frac{Area\ After\ Protection}{Original\ Area} - 1\right) \times 100$

overhead. Similarly, *apex*2, *seq*, *table*3 and *table*5 benchmarks also achieve 98% reliability against single fault without any protection/area overhead. Only *seq* circuit achieves 99% reliability without any overhead. The average area overhead required by the proposed algorithm to achieve 95%, 98% and 99% reliability is 22.81%, 55.96% and 92.77%, respectively.

Next, the comparison is made between the proposed technique and the asymmetric transistor sizing technique of sensitive gates in [1]. The transistor sizing technique proposed by Lazzari et al. [1] asymmetrically sizes the transistors connected to the output of a gate i.e., `nmos` and `pmos` networks are sized independently. The technique proposed by Lazzari et al. [1] is implemented as follows. The sensitivity of a gate is measured by considering sa0 and sa1 fault detection probabilities independently. Gates are then sorted according to their detection probabilities. Algorithm 1 is then applied, but now the possible protections that can be applied to a gate are restricted to transistors connected to the output of a gate. For example, for a 2-input NAND gate possible protections are NAND21, NAND23 and NAND24 only. After each protection applied to a gate, $\mathbf{POF_C}$ of circuit is updated using Eqn. 3.7. The process is repeated until the reliability/area overhead requirement is met or all possible protections are applied to all gates.

The results of this technique are shown in Table 3.7. It can be observed that by selectively protecting the transistors connected to the output of a gate, benchmarks such as $b12$, $clip$, $ex5$, $misex1$, $misex2$, $rd84$, $squar5$ and $z5xp1$ are unable to achieve 99% reliability against single fault. Benchmarks $b12$, $misex1$, $squar5$ and $z5xp1$ are also unable to achieve 98% reliability. In addition to that, the area overhead becomes significantly higher in comparison to the proposed STR technique even if the required reliability measure of 99% is achieved against single fault.

The technique in [68] protects all sensitive gates symmetrically, i.e. all transistors in a sensitive gate are protected and are equally scaled. Comparison is also made with the technique similar to [68] based on fully protecting sensitive gates but with protecting transistors asymmetrically. Protecting transistors asymmetrically has an advantage over symmetric protection due to the difference in

Table 3.7: Reliability of circuits based on Lazzari [1] gate sizing technique against a single fault.

Circuit	**95%**		**98%**		**99%**	
	OH	**OH**	**OH**	**Rel**	**OH**	**Rel**
alu4	0%	97.89%	70.33%	98.01%	218.64%	99.01%
apex1	0%	96.72%	55.42%	98.01%	312.95%	99.00%
apex2	0%	99.20%	0%	99.20%	0%	99.20%
apex3	0%	96.88%	43.37%	98.00%	333.11%	99.01%
apex4	0%	96.20%	108.59%	98.00%	483.00%	99.34%
b12	186.03%	95.05%	358.85%	96.78%	358.84%	96.78%
clip	43.15%	95.07%	203.47%	98.00%	321.62%	98.47%
cordic	0%	98.10%	0%	98.10%	122.59%	99.00%
ex5	33.97%	95.02%	349.78%	98.00%	429.32%	98.22%
misex1	197.69%	95.00%	322.86%	97.06%	322.86%	97.06%
misex2	17.05%	95.01%	394.03%	98.00%	504.89%	98.36%
misex3	0%	97.64%	4.18%	98.02%	211.21%	99.10%
rd84	43.98%	95.10%	237.37%	98.07%	456.35%	98.64%
seq	0%	99.05%	0%	99.05%	0%	99.05%
squar5	197.74%	95.02%	321.85%	97.13%	321.85%	97.13%
table3	0%	98.68%	0%	98.68%	0.82%	99.01%
table5	0%	98.70%	0%	98.70%	1.30%	99.01%
z5xp1	151.57%	95.02%	323.83%	97.62%	323.83%	97.62%
Avg.	**48.40%**	**96.63%**	**155.22%**	**98.02%**	**262.40%**	**98.50%**

characteristics of `nmos` and `pmos` transistors. The sensitivity of a gate is measured as the sum of sa0 and sa1 fault detection probabilities. Gates are then sorted according to their detection probabilities. Algorithm 1 is then applied by fully protecting the gate with the highest detection probability. For example, a 2-input NAND gate will be implemented as NAND25 in Table 3.2. After each protection applied to a gate, $\mathbf{POF_C}$ of circuit is updated using Eqn. 3.7. The process is repeated until the reliability/area overhead requirement is met or all gates are fully protected.

Table 3.8 highlights the area overhead incurred by fully protecting sensitive gates asymmetrically against a single fault. It can be observed from Table 3.6

Table 3.8: Reliability of circuits based on asymmetric gate sizing technique a against single fault.

Circuit	**95%**		**98%**		**99%**	
	OH	**Rel**	**OH**	**Rel**	**OH**	**Rel**
alu4	0%	97.89%	24.06%	98.01%	60.48%	99.18%
apex1	0%	96.72%	19.54%	98.20%	64.64%	99.28%
apex2	0%	99.20%	0%	99.20%	0%	99.20%
apex3	0%	96.88%	28.04%	98.02%	72.03%	99.18%
apex4	0%	96.20%	32.92%	98.05%	94.94%	99.12%
b12	89.40%	95.30%	173.50%	98.20%	264.83%	99.20%
clip	14.64%	95.22%	79.34%	98.11%	144.09%	99.19%
cordic	0%	98.10%	0%	98.10%	21.17%	99.06%
ex5	41.19%	95.11%	117.69%	98.19%	195.46%	99.10%
misex1	95.68%	95.19%	209.66%	98.13%	287.38%	99.10%
misex2	43.76%	95.34%	118.00%	98.06%	156.35%	99.11%
misex3	0%	97.64%	6.15%	98.10%	43.36%	99.20%
rd84	19.07%	95.01%	66.67%	98.11%	133.41%	99.02%
seq	0%	99.05%	0%	99.05%	0%	99.05%
squar5	112.46%	95.22%	231.08%	98.21%	327.28%	99.21%
table3	0%	98.68%	0%	98.68%	3.99%	99.24%
table5	0%	98.70%	0%	98.70%	4.41%	99.28%
z5xp1	72.59%	95.12%	156.79%	98.20%	235.18%	99.26%
Avg.	**27.16%**	**96.70%**	**70.19%**	**98.30%**	**117.17%**	**99.17%**

and Table 3.8 that the proposed technique offers less area overhead as compared to the asymmetric technique for all protection threshold scenarios. Also, under the 99% column header in Table 3.6 and Table 3.8, it is evident that the proposed technique achieves significant area savings for 13 out of 18 benchmark circuits with similar reliability measures.

The simulations are further extended to analyze circuits reliability against multiple faults. The number of faults injected is correlated to the area of a circuit. Table 3.9 shows the reliability achieved by prorating the 1, 2 and 5 faults for each circuit according to its area. For example, if the area overhead is 131%, then the actual area is increased by a factor of 2.31. So, 1, 2 and 5 faults in the original circuit will prorate to 2.31, 4.62 and 11.55 faults in the protected circuit. For each prorated fault, the circuit is simulated twice. For example, if the prorated faults to be injected are 4.62, then the circuit is simulated twice, once by injecting 4 faults and another by injecting 5 faults. The failure rate achieved by both fault injection scenarios is then computed based on a weighted average to compute the final failure rate/reliability. It is interesting to observe that with the prorated faults, the average reliability achieved by the proposed method with 99% protection is above 96% for 1 and 2 prorated faults. The reliability measures achieved by asymmetric sizing technique against prorated faults is shown in Table 3.10. It can be observed that the average reliability achieved by the proposed scheme under all fault injection scenarios and for all protection thresholds are better/close to the asymmetric gate sizing technique.

To further illustrate the advantage of the proposed STR technique against techniques that fully protect sensitive gates, Table 3.11 shows the percentage distribution of gates that have been protected with *Single-Transistor protection (1T)* e.g., NAND21 from Table 3.2, *Full Protection (FP)* e.g., NAND25 from Table 3.2 and *No Protection (NP)* for each circuit when Algorithm 1 is applied for target reliability of 98% and 99%. It is clear from the table that for some circuits the percentage of protected gates without full protection is significant. This percentage is even higher than the percentage of fully protected gates such as $apex2$, $apex3$, $cordic$, $misex2$, $table3$ and $table5$.

Table 3.12 shows the reliability achieved by TMR algorithm. TMR algorithm is evaluated under the same conditions as for Algorithm 1. The average area incurred by TMR is always more than three times the original area. Comparing to TMR, it can be observed that the average reliability achieved by the proposed scheme under all fault injection scenarios and for all protection thresholds are far better. With 95% protection threshold and an area overhead of just 22.81%, better reliability is achieved by the proposed algorithm than TMR. This is due to the fact that voters in TMR technique are not protected.

To improve the reliability of the TMR technique, the majority voters are protected by fully protecting the voters using proposed STR scheme. The results for TMR with voter protection are shown in Table 3.12b. It can be observed that the average reliability results have significantly improved for different fault injection scenarios as compared to TMR without voter protection at the expense of additional average area overhead of $\approx 28.5\%$. In comparison to TMR with voter protection, the proposed STR technique with 99% protection threshold achieves comparable reliability with a significantly lower area overhead.

For further evaluation, the proposed scheme is then compared to the simulation-based synthesis technique [2]. The technique is based on maximizing the probability of logical masking when a soft error occurs. This is done by extracting sub-circuits from an original multi-level circuit, and then re-synthesizing each extracted sub-circuit to increase fault masking against a single fault, taking advantage of input probabilities and don't care conditions. Table 3.13 shows the reliabilities obtained based on the original circuit, the circuits synthesized by [2] and by the application of the proposed STR technique for the same area overhead obtained by [2]. From Table 3.13, it is clear that the final synthesized circuits from [2] are unable to achieve 95% reliability against single fault except for $ex1010$. This is a limitation of the technique in [2] as it improves reliability but cannot achieve a given target reliability. The proposed STR technique achieves slightly better results for all fault injection scenarios in comparison to the circuit synthesized by the technique in [2]. However, the proposed STR technique has the advantage that it can be applied to achieve any given target reliability or under any given area overhead constraint.

It is worth mentioning that the technique in [2] and the proposed STR technique are complementary to each other. This is because the technique in [2] is based on enhancing logical masking and is applied at the gate level while thr proposed STR technique is based on protecting sensitive transistors at the transistor level through transistor sizing. Hence, applying both techniques could produce better results than applying any of the techniques separately. To illustrate this, Algorithm 1 is applied on both the original circuits and the synthesized circuits obtained by [2] with target reliability of 99%. From Table 3.14, it is clear that the proposed technique applied on top of the synthesized circuits obtained by [2] result in significant area savings as compared to applying STR alone on the original circuits. This clearly indicates that the proposed method is scalable and can be used to further improve other techniques.

3.5 Conclusion

In this chapter, a selective transistor-redundancy based fault tolerance technique for combinational circuits is proposed. The technique can be applied to achieve a given circuit reliability or enhance the reliability of a circuit under a given area constraint. The technique is based on estimating the failure probability of each transistor and iteratively protecting transistors with the highest failure probability until the desired objective is achieved. Transistors are protected based on duplicating and scaling a subset of transistors necessary for providing the protection. Experimental results on LGSynth91 benchmarks demonstrate the effectiveness of the proposed technique. Compared to existing transistor sizing techniques, the proposed algorithm incurs significantly less area overhead with similar reliability measures. Better reliability results are also achieved in comparison to TMR with lower area overhead. Unlike TMR which has an area overhead of at least 3 times the area overhead of the original circuit, the area overhead of the proposed technique varies depending on the reliability of the original circuit. For some circuits, high reliability ($> 99\%$) is achieved with small area overhead ($< 10\%$). In addition, the reliability of the TMR technique has been enhanced significantly by protecting the voters based on applying the proposed technique. Additionally, comparison with simulation-based synthesis technique further highlights the merit of the proposed method.

Table 3.9: Reliability of circuits based on the proposed STR technique against prorated faults.

(a) 1 prorated fault.

Circuit	**95%**	**98%**	**99%**
alu4	97.89%	98.11%	98.63%
apex1	96.72%	98.16%	98.66%
apex2	99.20%	99.20%	99.20%
apex3	96.88%	98.21%	98.43%
apex4	96.20%	97.85%	98.41%
b12	92.50%	95.74%	98.18%
clip	95.19%	97.39%	97.60%
cordic	98.10%	98.10%	98.98%
ex5	93.78%	96.25%	97.64%
misex1	92.62%	95.50%	97.15%
misex2	95.57%	97.33%	98.32%
misex3	97.64%	98.28%	98.61%
rd84	94.68%	97.96%	98.29%
seq	99.05%	99.05%	99.05%
squar5	91.50%	95.81%	97.10%
table3	98.68%	98.68%	99.20%
table5	98.70%	98.70%	99.08%
z5xp1	92.17%	96.22%	96.83%
Avg.	**95.95%**	**97.59%**	**98.30%**

(b) 2 prorated faults.

Circuit	**95%**	**98%**	**99%**
alu4	95.86%	96.46%	97.69%
apex1	94.20%	96.12%	97.07%
apex2	98.04%	98.04%	98.04%
apex3	94.76%	95.67%	97.30%
apex4	92.74%	95.21%	96.71%
b12	82.69%	91.39%	95.16%
clip	94.00%	94.50%	95.86%
cordic	96.28%	96.28%	97.73%
ex5	89.15%	93.49%	96.11%
misex1	83.51%	90.87%	94.57%
misex2	90.75%	94.40%	96.75%
misex3	95.40%	95.86%	97.53%
rd84	90.70%	93.83%	95.95%
seq	98.22%	98.22%	98.22%
squar5	81.17%	89.63%	94.73%
table3	97.78%	97.78%	98.05%
table5	97.72%	97.72%	97.58%
z5xp1	84.41%	91.08%	95.01%
Avg.	**92.08%**	**94.81%**	**96.67%**

(c) 5 prorated faults.

Circuit	**95%**	**98%**	**99%**
alu4	87.44%	90.26%	93.92%
apex1	86.40%	89.98%	93.47%
apex2	95.42%	95.42%	95.42%
apex3	85.66%	90.20%	93.52%
apex4	84.16%	88.12%	92.40%
b12	60.89%	79.48%	88.80%
clip	76.29%	85.16%	91.40%
cordic	92.14%	92.14%	95.25%
ex5	77.95%	83.77%	89.60%
misex1	62.12%	76.53%	86.88%
misex2	77.93%	88.25%	92.05%
misex3	88.90%	90.34%	94.37%
rd84	77.46%	86.07%	90.99%
seq	95.74%	95.74%	95.74%
squar5	58.77%	72.58%	84.07%
table3	94.28%	94.28%	95.15%
table5	94.52%	94.52%	95.99%
z5xp1	65.57%	80.45%	86.19%
Avg.	**81.20%**	**87.40%**	**91.96%**

Table 3.10: Reliability of circuits based on asymmetric gate sizing technique against prorated faults.

(a) 1 prorated fault.

Circuit	**95%**	**98%**	**99%**
alu4	97.89%	97.93%	98.72%
apex1	96.72%	97.92%	98.58%
apex2	99.20%	99.20%	99.20%
apex3	96.88%	97.46%	98.59%
apex4	96.20%	97.62%	98.10%
b12	92.81%	96.13%	98.19%
clip	94.77%	97.31%	98.26%
cordic	98.10%	98.10%	99.07%
ex5	92.90%	96.70%	98.15%
misex1	92.02%	95.40%	97.28%
misex2	93.46%	97.00%	98.72%
misex3	97.64%	98.40%	98.49%
rd84	94.45%	97.51%	98.70%
seq	99.05%	99.05%	99.05%
squar5	91.59%	95.56%	97.93%
table3	98.68%	98.68%	98.93%
table5	98.70%	98.70%	99.06%
z5xp1	92.03%	95.37%	97.77%
Avg.	**95.73%**	**97.45%**	**98.49%**

(b) 2 prorated faults.

Circuit	**95%**	**98%**	**99%**
alu4	95.86%	95.71%	98.18%
apex1	94.20%	95.18%	97.36%
apex2	98.04%	98.04%	98.04%
apex3	94.76%	95.64%	97.04%
apex4	92.74%	95.43%	96.27%
b12	84.23%	91.48%	95.76%
clip	88.73%	93.99%	96.38%
cordic	96.28%	96.28%	97.92%
ex5	88.39%	92.77%	96.18%
misex1	82.55%	91.37%	95.28%
misex2	87.84%	93.75%	96.51%
misex3	95.40%	96.14%	97.60%
rd84	90.22%	93.86%	96.64%
seq	98.22%	98.22%	98.22%
squar5	82.49%	90.35%	95.40%
table3	97.78%	97.78%	98.07%
table5	97.72%	97.72%	98.25%
z5xp1	85.25%	91.48%	94.73%
Avg.	**91.71%**	**94.73%**	**96.88%**

(c) 5 prorated faults.

Circuit	**95%**	**98%**	**99%**
alu4	87.44%	90.69%	94.00%
apex1	86.40%	90.05%	93.84%
apex2	95.42%	95.42%	95.42%
apex3	85.66%	88.26%	93.16%
apex4	84.16%	88.26%	91.94%
b12	62.81%	80.45%	90.68%
clip	76.06%	85.34%	91.15%
cordic	92.14%	92.14%	94.63%
ex5	73.93%	83.93%	90.25%
misex1	63.54%	77.23%	88.88%
misex2	71.17%	83.51%	90.92%
misex3	88.90%	91.27%	93.78%
rd84	76.22%	86.37%	91.58%
seq	95.74%	95.74%	95.74%
squar5	59.87%	74.52%	86.39%
table3	94.28%	94.28%	94.89%
table5	94.52%	94.52%	95.10%
z5xp1	66.50%	81.27%	88.09%
Avg.	**80.82%**	**87.40%**	**92.25%**

Table 3.11: Distribution of protection schemes.

Circuit	**# of Gates**	**98%**			**99%**		
		1T[1]	**Full**[2]	**NP**[3]	**1T**	**Full**	**NP**
alu4	832	0.36%	3.25%	95.91%	0.96%	9.74%	87.98%
apex1	2723	2.75%	1.54%	95.41%	5.25%	7.27%	86.49%
apex2	372	0%	0%	100%	0.54%	0%	99.46%
apex3	1791	6.09%	1.34%	92.29%	7.26%	6.98%	83.53%
apex4	2539	3.54%	4.37%	91.77%	8.82%	12.60%	77.79%
b12	88	1.14%	22.73%	73.86%	1.14%	36.36%	60.23%
clip	228	1.32%	14.04%	83.77%	0.88%	24.12%	70.61%
cordic	163	0%	0%	100%	1.23%	1.23%	96.32%
ex5	648	10.49%	7.10%	80.09%	12.35%	17.59%	67.75%
misex1	108	0.93%	35.19%	62.04%	0%	52.78%	44.44%
misex2	151	17.22%	8.61%	70.86%	17.22%	14.57%	64.24%
misex3	1100	2.18%	0.09%	97.64%	1.73%	3.91%	93.64%
rd84	296	1.69%	14.86%	82.77%	3.38%	22.97%	72.30%
squar5	71	0%	42.25%	53.52%	0%	54.93%	40.85%
table3	1953	0%	0%	100%	0.77%	0%	99.23%
table5	2020	0%	0%	100%	1.39%	0.05%	98.51%
z5xp1	176	1.70%	33.52%	64.20%	1.70%	46.02%	49.43%

[1] % of gates with single transistor protection

[2] % of gates with full protection

[3] % of gates with no protection

Table 3.12: Reliability of circuits based on TMR technique with prorated faults.

(a) TMR without voter protection.

Circuit	**OH**	**1 Fault**	**2 Faults**	**5 Faults**
alu4	203.93%	99.24%	98.69%	94.67%
apex1	206.56%	98.23%	95.54%	89.50%
apex2	203.45%	98.84%	98.17%	94.09%
apex3	211.60%	96.18%	93.22%	81.57%
apex4	202.76%	99.49%	98.15%	95.67%
b12	251.92%	90.69%	80.89%	54.49%
clip	209.41%	98.11%	95.78%	84.77%
cordic	205.83%	98.83%	97.21%	95.60%
ex5	245.25%	91.15%	83.74%	62.89%
misex1	232.14%	90.81%	79.86%	51.91%
misex2	254.73%	81.55%	67.68%	36.96%
misex3	205.21%	98.78%	96.78%	90.70%
rd84	205.66%	98.75%	96.08%	84.78%
seq	204.94%	98.24%	97.28%	92.79%
squar5	255.38%	85.50%	73.56%	36.54%
table3	202.83%	99.00%	98.36%	95.33%
table5	202.98%	99.03%	98.26%	94.76%
z5xp1	225.16%	93.91%	87.78%	65.90%
Avg.	**218.32%**	**95.35%**	**90.94%**	**77.94%**

(b) TMR with voter protection.

Circuit	**OH**	**1 Fault**	**2 Faults**	**5 Faults**
alu4	207.53%	99.78%	99.30%	97.07%
apex1	215.41%	99.72%	99.59%	98.05%
apex2	207.51%	99.93%	99.37%	98.44%
apex3	228.91%	99.59%	99.36%	98.22%
apex4	205.21%	99.60%	99.16%	97.91%
b12	329.44%	98.64%	95.30%	76.26%
clip	222.78%	99.75%	98.10%	90.49%
cordic	211.43%	99.79%	98.65%	95.97%
ex5	305.03%	99.67%	98.30%	96.34%
misex1	288.88%	98.53%	94.35%	73.16%
misex2	347.30%	99.37%	98.02%	92.85%
misex3	212.74%	99.73%	99.15%	97.47%
rd84	212.58%	99.47%	98.05%	89.84%
seq	210.86%	99.76%	99.63%	98.77%
squar5	356.21%	98.16%	91.33%	60.67%
table3	205.66%	99.84%	99.38%	98.53%
table5	206.14%	99.84%	99.44%	98.61%
z5xp1	268.10%	98.91%	95.87%	81.39%
Avg.	**246.76%**	**99.45%**	**97.91%**	**91.11%**

Table 3.13: Comparison of circuit reliability for proposed STR technique with the technique in [2].

Circuit	Original			Synthesized by [2]				STR applied to Original[3]		
	Area (μ)	**S[1]**	**2**	**OH**	**S**	**1P[2]**	**2P**	**S**	**1P**	**2P**
apex3	1994.46	84.16%	68.96%	31.68%	92.60%	89.18%	78.52%	93.15%	91.45%	82.78%
apex4	2532.66	87.06%	76.70%	50.79%	95.74%	92.63%	86.37%	96.20%	93.95%	87.60%
bench1	1313.52	82.32%	67.98%	34.98%	92.86%	91.18%	83.17%	93.46%	91.69%	84.10%
cps	1452.36	78.14%	59.78%	46.35%	91.64%	88.51%	77.85%	92.82%	89.33%	81.43%
duke2	535.86	79.22%	64.10%	30.86%	91.06%	87.83%	77.43%	91.90%	89.55%	78.42%
ex1010	4219.02	87.64%	79.18%	42.17%	95.52%	94.85%	89.88%	96.22%	95.75%	90.41%
exp	363.48	75.34%	56.72%	27.90%	89.48%	85.86%	74.10%	89.38%	86.98%	75.53%
misex3	883.74	87.36%	76.18%	29.30%	94.30%	93.44%	86.15%	95.12%	94.35%	85.88%
spla	475.8	81.08%	65.36%	18.85%	87.62%	86.02%	74.83%	89.92%	87.61%	75.82%
table3	991.38	85.74%	73.18%	29.19%	93.28%	92.08%	83.93%	94.12%	92.34%	85.47%
table5	1106.04	82.64%	68.56%	38.22%	94.30%	92.45%	85.69%	94.78%	92.26%	86.78%
test1	1040.52	82.00%	68.18%	37.18%	92.82%	89.95%	82.01%	93.78%	90.09%	83.60%
Avg.		**82.73%**	**68.74%**	**34.79%**	**92.60%**	**90.33%**	**81.66%**	**93.40%**	**91.28%**	**83.15%**

[1] Single fault
[2] 1 prorated faults
[3] STR applied to Original with area overhead constraint mentioned in column header “OH”

Table 3.14: Reliabilities of circuits based on applying proposed STR technique to circuits obtained by the technique in [2].

Circuit	STR Applied to Original with 99%		STR Applied to Syntheized by [2] with 99%	
	OH	**Rel**	**OH**	**Rel**
apex3	170.27%	99.10%	141.93%	99.05%
apex4	109.29%	99.02%	96.83%	99.10%
bench1	148.65%	99.01%	100.04%	99.09%
cps	164.66%	99.00%	160.33%	99.01%
duke2	140.69%	99.06%	135.10%	99.03%
ex1010	90.39%	99.10%	60.54%	99.01%
exp	168.03%	99.04%	149.10%	99.02%
misex3	81.65%	99.05%	75.92%	99.12%
spla	115.83%	99.05%	112.21%	99.00%
table3	89.77%	99.10%	67.00%	99.12%
table5	116.56%	99.10%	71.00%	99.13%
test1	165.14%	99.08%	115.49%	99.02%
Avg.	**130.08%**	**99.06%**	**107.12%**	**99.06%**

CHAPTER 4

DOUBLE MODULAR REDUNDANCY (DMR) BASED FAULT TOLERANCE TECHNIQUE FOR COMBINATIONAL CIRCUITS

The objective in this chapter is to improve reliability of combinational circuits based on the double modular redundancy scheme. As opposed to TMR, where each module is triplicated followed by a voter, each module in the proposed Double Modular Redundancy (DMR) scheme is duplicated followed by a NAND/AND masking gate. Modules are synthesized by either synthesizing the true or the complement function to maximize soft error masking. Secondly, the proposed selective transistor redundancy technique in Chapter 3 is also applied to the proposed DMR technique in order to further improve its reliability. In addition, improved application of DMR based on the use of C-element is illustrated.

The rest of the chapter is organized as follows. Section 4.1 discusses the motivation and implementation of the proposed DMR technique, Section 4.2 discusses modular redundancy technique that employs C-element as a voter, simulation results are elaborated in Section 5.4 and finally, the chapter is concluded in Section 7.1.

4.1 Proposed Double Modular Redundancy Fault Tolerance Technique

In this Section, a double modular redundancy (DMR) fault tolerance technique is proposed targeting the achievement of high reliability with reduced area overhead. The technique is based on identifying the probability of occurrence of logic values "0" and "1" at each primary output of a circuit. The consideration made here is that the circuit consists of two-level logic, represented in sum-of-products form. Next, different cases in relation to the output value probability of occurrence to illustrate the proposed DMR technique are discussed.

If ($Prob_0 > Prob_1$) for an output Y, then the true value of Y will be implemented as a sum-of-products circuit. Furthermore, the logic cone that has Y as an output will be duplicated and the two duplicate outputs will be combined using an AND gate as a masking gate. To elaborate this rule, consider the case when the output of a module produces a logic “0” value (case with highest probability). In this case, it is guaranteed t hat all single or multiple faults occurring in a single module will be tolerated as the AND masking gate will produce the correct output value due to the logic “0” value produced by the other module at its second input. However, when the module output produces a logic “1” value (case with lower probability), then at least one of the AND gates in the sum-of-products representation of the Y module produces a logic “1” value. This will mask all errors occurring on the other AND gates in the module as they are combined using an OR gate which retains the correct output value. The only fault that may not be protected for the AND gates is the one occurring on the AND gate that produces logic value “1”, assuming that a single AND gate produces a logic “1” value. If there are more than one AND gate producing logic value “1”, then all faults occurring on AND gates will be protected. Faults occurring on the OR gate in the sum-of-products module when the module produces an output logic value “1” are not protected. Such unprotected faults occurring on the OR gate and on the masking AND gate can be protected based on applying the selective transistor redundancy (STR) technique from Chapter 3, which is based on protecting sensitive transistors that have high detection probability by transistor duplication and sizing. Assuming two-level NAND-NAND implementation of sum-of-products expression, faults occurring on the 2nd level NAND gate (implementing the OR gate) needs to be protected when the output produced a logic value “1”. This implies that the gate needs to be protected for soft errors occurring on `nmos` transistors. As illustrated in Chapter 3, protecting `nmos` transistor faults is much cheaper than protecting `pmos` transistor faults as for each protected `nmos` transistor only the corresponding `pmos` transistor needs to be duplicated and scaled.

If ($Prob_1 > Prob_0$) for an output Y, then the complement function $\overline{Y}$ will be implemented for the output Y. The implemented $\overline{Y}$ function will then be duplicated and the two duplicate outputs are combined using a NAND gate as a masking gate. The rationale behind this rule is that if $Prob_1 > Prob_0$ and the logic is implemented as $\overline{Y}$, then $\overline{Y}$ will feed logic value “0” most of the time to the NAND masking gate, which will provide protection against all faults occurring in a single module. When the output produces the least probable value “0”, $\overline{Y}$ will produce the logic value “1”, which will provide the protection of faults occurring at all AND gates implementing the sum-of-products expression that are not producing a logic “1” value in the case of a single AND gate producing a logic “1” value. Implementing the function as $\overline{Y}$ for $Prob_1 > Prob_0$ using a NAND masking gate case makes it equivalent to implementing the function Y using an AND masking gate for the $Prob_0 > Prob_1$ case. It is worth noting that for this case (i.e., $Prob_1 > Prob_0$), the proposed DMR technique is more effective than the technique proposed in [7] which implements the true value of Y and combines the two duplicate outputs using a masking OR gate. While using a masking OR gate provides protection when the output produces the dominant logic “1” value, it provides no fault protection when the output produces the logic “0” value.

Algorithm 2 highlights the steps of the proposed DMR algorithm. Circuits are synthesized using the *Espresso* [85] tool and then mapped using the *SIS* [86] tool to a library consisting of 2-, 3-, 4-input NAND/NOR gates and an Inverter. While synthesizing using *Espresso*, having `phase=0` synthesizes the OFF-set of the corresponding output function while having `phase=1` synthesizes the ON-set. By default, the ON-set of each output is synthesized i.e., `phase=1` for each output, as shown in line 6. Once the proper gate level representation of a circuit is obtained, $Prob_0$ and $Prob_1$ at each output i is computed using simulations of 1 million random input vectors, as shown in line 7. Then, $Phase_i$ for each output i is assigned a value based on the condition in line 9. After $Phase_i$ is computed, the original circuit is re-synthesized with the new/updated `phase` value for each output. The synthesized module for each output is then duplicated and the two outputs are combined using AND masking gate when $Phase_i = 1$ and using NAND masking gate when $Phase_i = 0$.

As an illustrative example of the proposed DMR algorithm, let us consider a single output, Y_0, sum-of-products circuit composed of 4 AND gates, G_0, G_1, G_2, G_3, and one OR gate, F_0, as shown in Fig. 4.1.

For the sake of argument, let’s consider that $Prob_0(= 60\%) > Prob_1(= 40\%)$ at the output Y_0. In this case, the logic cone having Y_0 as an output will be synthesized in sum-of-products form. Two copies of the circuit will be created leading to two outputs Y_{01} and Y_{02}. The two outputs are combined using a masking

Algorithm 2 : Proposed DMR Algorithm

```
Require: Circuit in pla format
$Prob_{i0}$ : Probability of 0 at output $i$
$Prob_{i1}$ : Probability of 1 at output $i$
$Phase_i$ : Phase/polarity of output $i$ to be synthesized
$|outputs|$ : Number of primary outputs in circuit

Synthesize ON-set of each output of the circuit
Compute $Prob_{i0}$ and $Prob_{i1}$ of each output using simulation
for ($i = 1 \rightarrow |outputs|$) do
    if ($Prob_{i0} > Prob_{i1}$) then
        Synthesize output $i$ with $Phase_i = 1$
        Duplicate synthesized output $i$ module and connect the two outputs using $AND$ masking gate
    else
        Synthesize output $i$ with $Phase_i = 0$
        Duplicate synthesized output $i$ module and connect the two outputs using $NAND$ masking gate
    end if
end for
```

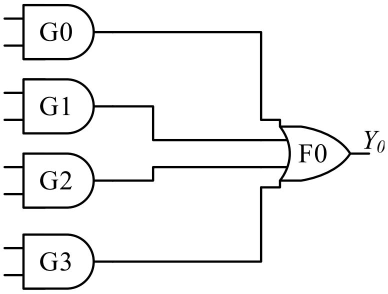


Figure 4.1: A simple two-level circuit.

AND gate, implemented as a NAND gate followed by an inverter as shown in Fig. 4.2. Since 60% of the time, output values from gates $F01$ and $F02$ will be "0", therefore, faults occurring on any single output module will be masked 60% of the time as the output of the non-faulty module will produce logic value "0", which will mask the propagation of the fault effect to the output.

Now, consider the case when 40% of the time the output of gates $F01$ and $F02$ produce logic value "1" as depicted in Fig. 4.3. To get logic "1" at the output of

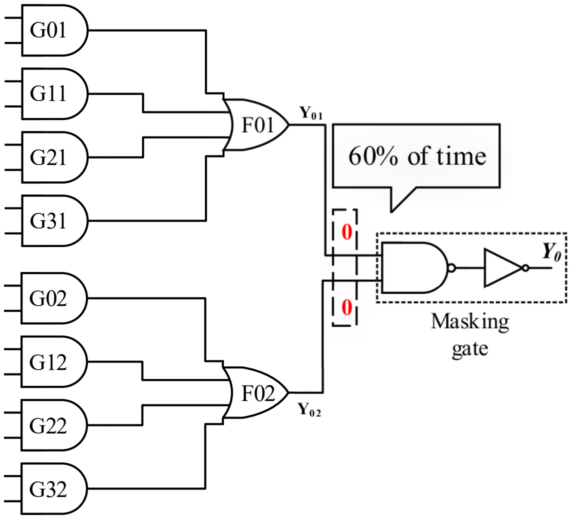


Figure 4.2: Circuit after output module duplication and addition of masking AND gate.

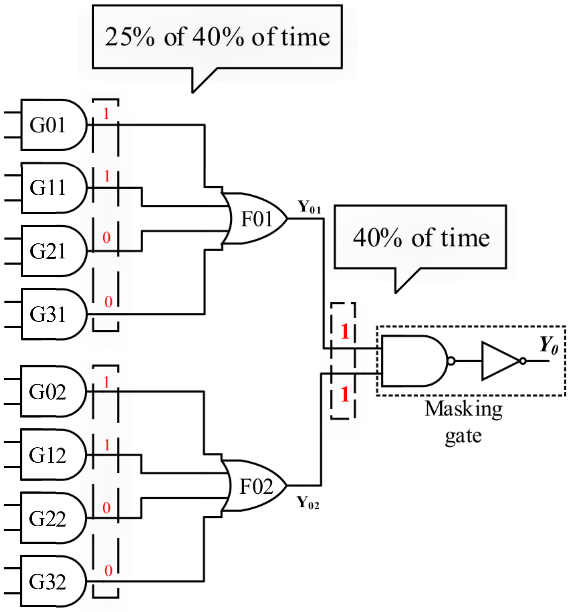


Figure 4.3: Case 1 of 40% of the time logic “1” value is produced at the masking gate input.

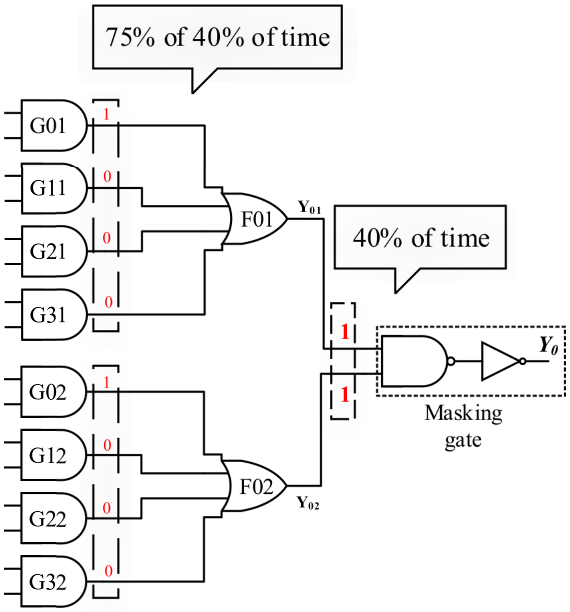


Figure 4.4: Case 2 of 40% of the time logic “1” value is produced at the masking gate input.

gates $F01$ and $F02$, at least one logic “1” value is required at their inputs. Let’s assume that at least 25% of 40% of the time, constituting 10% of the total time, at least two AND gates in each logic cone produce a logic “1” value as highlighted in Fig. 4.3. No protection is required in this scenario because any single fault striking any AND gate will be masked by the OR gates $F01$ or $F02$ due to having at least one logic “1” value at their inputs.

Finally, consider the case when only one *AND* gate produces logic “1” value in each logic cone as shown in Fig. 4.4, and assume that this happens for the 75% of the 40% of time, i.e. 30% of the total time. Even in this case, for faults hitting any of the AND gates producing logic value “0” will be masked. However, if a fault strikes at the *AND* gate producing logic value “1”, then this fault will not be masked and will be observable at the output. Considering all these scenarios, it can be seen that based on the proposed DMR approach high reliability can be achieved with nearly double the area overhead.

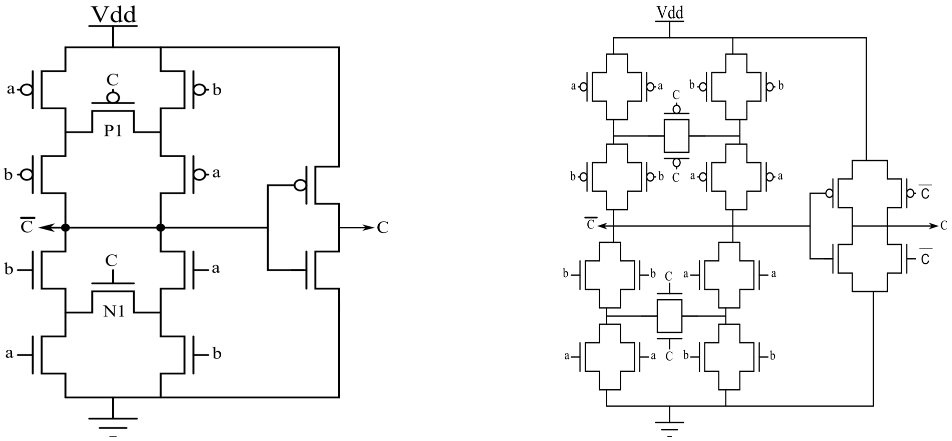
4.2 Improved C-Element Based DMR (DMR-CEL)

In this section, an improved implementation of DMR is discussed where C-element is used to combine the outputs of the two duplicate modules (DMR-CEL). Furthermore, investigation is also performed to study the implact of transistor-level protection applied to the C-element.

A C-element [54] consists of two inputs and one output. If both inputs are 0(1), then the output of C-element will be 0(1); otherwise the output preserves the previously stored value. For the correct operation of the C-element, it is assumed that once the input values become stable, they will not change their values until the output changes. The output c of the C-element can be expressed in terms of the inputs a and b and the complement of the current state of the output $\overline{c}$ by the following boolean function [53]:

$$c = \overline{c} \cdot (a + b) + a \cdot b \tag{4.1}$$

Several C-element implementations have been discussed and analyzed in detail by Shams et al. [53]. In this work, the C-element implementation by Van Berkel [11] is utilized and is shown in Fig. 4.5a. The advantages of this imple-



(a) Without protection.

(b) With protection.

Figure 4.5: Van Berkel C-Element [11].

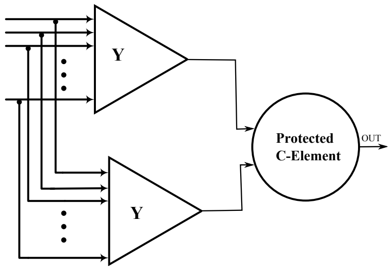


Figure 4.6: DMR-CEL: Logic cone synthesized based on true form.

mentation include being ratioless and symmetric with respect to the inputs. It has been shown by Shams et al. [53] that symmetric implementation is the best candidate for energy-efficient and high-speed designs.

The implementation of DMR-CEL for a circuit with a single output Y is shown in Fig. 4.6. For this scheme, circuits are synthesized differently than the proposed DMR scheme mentioned in Algorithm 2. Here, each output i is synthesized based on the majority phase i.e., $if(Prob_{i0} > Prob_{i1})$ then $Phase_i = 0$ and vice versa. For the case when $Prob_{i0} > Prob_{i1}$, an inverter is added at the output Y_i to get the true output value. The logic cone of Y_i is duplicated and the two outputs are fed to the C-element to generate the final output. Synthesizing the majority phase for each module has the advantage of masking many faults occurring at the first level AND gates in a sum-of-products implementation of each output as has been illustrated in Section 4.1.

The protected version of C-element against soft errors is shown in Fig. 4.5b, where each transistor is duplicated and connected in parallel with each transistor scaled (widths increased) with necessary scaling factors to provide protection. The area of C-element using 130nm Predictive Technology Model (PTM) [79], which is the drain width of all transistors, is 4.68μ. The protected version of C-element shown in Fig. 4.5b has an area of 21.216μ. Therefore, the area overhead of the protected C-element is $\left(\frac{21.216}{4.68} - 1\right) \times 100 = 353\%$.

The protected C-element provides protection against soft errors provided that both it's inputs are error free i.e., having the same logic value. In case input values to the C-element are not the same, i.e., one of the inputs is faulty, then a fault hitting transistors in the C-element may or may not excite a fault at the output. Fault excitation due to a particle strike at any of the transistors in the C-element depends on the location of the strike and the true output value. For example, if both inputs to the C-element (without protection) have logic value "1", then the true output will be logic value "1". Now, if a particle strikes any of the `pmos` transistors in the first stage (not including the transistors in the inverter), then $\overline{C}$ will be have the logic value "1" and then C will have the logic value "0" and the fault will propagate to the output. But, in case of the protected C-element in Fig. 4.5b, a fault hitting any of the `pmos` transistors will keep the node $\overline{C}$ having logic value “0” as the corresponding duplicated and scaled `nmos` transistors will suppress the fault effect, which makes the output C retain its true logic value “1”. For faults hitting transistors in the inverter connected to the output C, they are fully protected in the protected C-element while they are not in the unprotected version. The same arguments apply for the case when both inputs to the C-element have logic value “0” and faults strike any of the `nmos` transistors in the first stage.

4.3 Experimental Results

In this section, the impact of the proposed DMR and the improved DMR-CEL algorithms on the area and reliability of LGSynth’91 benchmarks is evaluated. The benchmarks consist of circuits with varying complexity in terms of area, number of inputs and outputs. The LGSynth’91 benchmark circuits used in this work are synthesized with single output optimization using *Espresso* [85] tool and then mapped to a library that consists of an Inverter, and 2-, 3- and 4-input NAND and NOR gates using *SIS* [86] tool. The reliability of a circuit is computed using the method discussed in Chapter 6, which is based on gate-level simulation with an accuracy of transistor-level simulation using SPICE.

The reliability of a circuit is computed against a single fault and prorated 1 and 2 faults for each circuit. The number of prorated faults is correlated to the area overhead of a circuit. For example, if the area overhead is 131%, then the actual area is increased from 1 to 2.31 times the original circuit. Therefore, 1, and

2 faults in the original circuit will prorate to 2.31 and 4.62 faults in the protected circuit. For each prorated fault, the circuit is simulated twice and the failure rate (FR) is computed based on a weighted average to compute the final failure rate/reliability. For example, if the prorated faults to be injected are 4.62, then the circuit is simulated twice, first by injecting 4 faults and then by injecting 5 faults. The failure rate is then computed as $0.38 * FR(4faults) + 0.62 * FR(5faults)$. For each fault injection scenario, faults are injected randomly and simulation is performed for 5000 iterations to compute the failure rate.

Table 4.1 shows the area overhead and reliabilities of circuits based on the proposed DMR technique against a single fault and the prorated 1, and 2 faults. The first column denotes the circuit name along with the number of primary inputs and outputs. The second and third columns show the area of the original circuit and the circuit obtained by the proposed DMR technique, based on the summation of `nmos` and `pmos` drain widths, respectively. The average area overhead of the proposed DMR technique is 105.69%. The average reliability against a single fault and 1 and 2 prorated faults is 98.48%, 97.02% and 94.20%, respectively.

Reliability of circuits obtained based on the proposed DMR technique can be further enhanced based on the application of the STR technique from Chapter 3. This technique can enhance the reliability of a circuit to any given reliability requirement based on transistor duplication and sizing. The STR technique is applied to the circuits obtained based on the proposed DMR technique with two reliability thresholds: 99.5% and 99.8%. The threshold value denotes the reliabil-

Table 4.1: Circuit reliability and area overhead based on the proposed DMR technique.

Circuit (In, Out)	**Area (μ)**[1]	**DMR Area (μ)**	**DMR OH**[2]	**1F**[3]	**P1**[4]	**P2**[5]
alu4 (14, 8)	1429.74	3772.86	163.88%	99.20%	97.49%	96.45%
apex1 (45, 45)	4602.00	8661.12	88.20%	98.90%	97.69%	95.30%
apex2 (39, 3)	609.96	1615.38	164.83%	99.70%	99.04%	98.49%
apex3 (54, 50)	3025.62	6300.84	108.25%	98.40%	97.45%	94.59%
apex4 (9, 19)	4575.48	9550.32	108.73%	99.05%	98.27%	96.31%
clip (9, 5)	372.84	745.68	100.00%	97.35%	94.35%	88.05%
cordic (23, 2)	241.02	361.14	49.84%	99.15%	98.60%	97.50%
ex5 (8, 63)	977.34	1994.46	104.07%	96.85%	93.72%	87.42%
misex2 (25, 18)	230.88	492.96	113.51%	95.85%	90.88%	83.88%
misex3 (14, 14)	1886.82	5370.30	184.62%	99.20%	98.62%	97.02%
rd84 (8, 4)	496.08	703.56	41.82%	96.25%	94.32%	88.75%
seq (41, 35)	4970.94	9463.74	90.38%	99.65%	99.09%	97.70%
table3 (14, 14)	3475.68	6179.16	77.78%	99.50%	99.26%	99.14%
table5 (17, 15)	3535.74	6496.62	83.74%	99.60%	99.46%	98.20%
Avg.			**105.69%**	**98.48%**	**97.02%**	**94.20%**

[1] Summation of `nmos` and `pmos` drain widths

[2] Area overhead (OH)=$\left(\frac{DMR\ Area}{Area} - 1\right) \times 100$

[3] Single Fault

[4] Prorated 1 fault

[5] Prorated 2 faults

ity of a circuit required to be achieved against a single fault. It is evident from Table 4.2 that the desired reliability threshold is achieved for all circuits with average area overhead of 137.82% and 180.57%, respectively.

Table 4.3 shows the area overhead and reliabilities of circuits obtained based on applying the TMR technique without and with voter protection. Voters are fully protected based on applying the STR technique from Chapter 3. In comparison to the TMR without voter protection, it can be observed that the proposed DMR technique achieves better reliability for prorated faults with significantly lower area overhead. Reliabilities of circuits designed based on TMR are reduced due to soft errors hitting the non-protected voters especially when a circuit has a large

Table 4.2: Circuit reliability and area overhead based on the combined application of the proposed DMR and STR (Chapter 3) techniques.

Circuit	**99.5%**				**99.8%**			
	OH	**1F**	**P1**	**P2**	**OH**	**1F**	**P1**	**P2**
alu4	181.39%	99.60%	99.12%	98.45%	224.53%	99.92%	99.45%	98.62%
apex1	97.78%	99.55%	99.08%	97.45%	133.86%	99.88%	99.23%	98.50%
apex2	164.83%	99.70%	99.04%	98.49%	171.27%	99.96%	99.70%	98.99%
apex3	122.23%	99.52%	99.20%	97.56%	179.18%	99.80%	98.81%	98.13%
apex4	123.14%	99.51%	99.13%	97.33%	184.60%	99.82%	99.33%	98.26%
clip	217.66%	99.58%	99.03%	95.45%	308.56%	99.84%	99.23%	96.45%
cordic	57.11%	99.61%	99.45%	98.89%	101.55%	99.85%	99.50%	96.03%
ex5	205.90%	99.56%	98.77%	96.16%	301.05%	99.87%	99.21%	96.44%
misex2	170.61%	99.66%	99.20%	97.10%	244.46%	99.88%	99.40%	97.80%
misex3	184.91%	99.57%	99.13%	97.67%	218.99%	99.86%	99.22%	98.54%
rd84	151.91%	99.51%	99.22%	97.88%	201.69%	99.84%	99.38%	98.20%
seq	90.46%	99.60%	99.39%	98.35%	93.84%	99.81%	99.57%	99.06%
table3	77.78%	99.50%	99.26%	99.14%	78.91%	99.83%	99.85%	99.59%
table5	83.74%	99.60%	99.46%	98.96%	85.52%	99.81%	99.74%	99.29%
Avg.	**137.82%**	**99.58%**	**99.18%**	**97.78%**	**180.57%**	**99.86%**	**99.40%**	**98.14%**

number of outputs. For TMR with voters protection, it can be observed that the average reliability has significantly improved for different fault injection scenarios as compared to TMR without voter protection at the expense of an additional average area overhead of ≈ 28.5%.

Table 4.3: Circuit reliability and area overhead based on TMR technique.

Circuit	**TMR without Voter Protection**				**TMR with Voter Protection**			
	OH	**1F**	**P1**	**P2**	**OH**	**1F**	**P1**	**P2**
alu4	203.93%	99.70%	99.24%	98.69%	207.53%	100%	99.78%	99.30%
apex1	206.56%	99.18%	98.23%	95.54%	215.41%	100%	99.72%	99.59%
apex2	203.45%	99.60%	98.84%	98.17%	207.51%	100%	99.93%	99.37%
apex3	211.60%	98.68%	96.18%	93.22%	228.91%	100%	99.59%	99.36%
apex4	202.76%	99.72%	99.49%	98.15%	205.21%	100%	99.60%	99.16%
clip	209.41%	99.42%	98.11%	95.78%	222.78%	100%	99.75%	98.10%
cordic	205.83%	99.62%	98.83%	97.21%	211.43%	100%	99.79%	98.65%
ex5	245.25%	96.90%	91.15%	83.74%	305.03%	100%	99.67%	98.30%
misex2	254.73%	93.56%	81.55%	67.68%	347.30%	100%	99.37%	98.02%
misex3	205.21%	99.12%	98.78%	96.78%	212.74%	100%	99.73%	99.15%
rd84	205.66%	99.60%	98.75%	96.08%	212.58%	100%	99.47%	98.05%
seq	204.94%	99.36%	98.24%	97.28%	210.86%	100%	99.76%	99.63%
table3	202.83%	99.66%	99.00%	98.36%	205.66%	100%	99.84%	99.38%
table5	202.98%	99.66%	99.03%	98.26%	206.14%	100%	99.84%	99.44%
Avg.	**211.80%**	**98.84%**	**96.81%**	**93.92%**	**228.51%**	**100%**	**99.70%**	**98.96%**

The average reliability of circuits designed using the combined application of the proposed DMR technique and STR technique with 99.5% protection threshold is slightly lower than the TMR technique with voter protection for all fault injection scenarios with significantly less area overhead. However, with 99.8% protection threshold, reliability of circuits is comparable to the TMR technique with voter protection but with significantly less area overhead i.e., 180.57 as compared to 228.51.

Table 4.4: Circuit reliability and area overhead based on DMR-CEL.

Circ.	**DMR-CEL (un-protected CEL)**				**DMR-CEL (protected CEL)**			
	OH	**1F**	**P1**	**P2**	**OH**	**1F**	**P1**	**P2**
alu4	102.62%	99.34%	98.60%	97.02%	111.87%	100%	99.96%	99.84%
apex1	104.37%	99.10%	98.16%	96.73%	119.82%	100%	99.42%	99.98%
apex2	102.30%	99.62%	98.95%	97.91%	110.43%	100%	99.96%	99.90%
apex3	107.73%	98.26%	96.72%	93.77%	135.06%	100%	98.70%	99.88%
apex4	101.84%	99.54%	99.25%	98.39%	108.35%	100%	99.55%	99.86%
clip	106.28%	98.34%	97.08%	93.34%	128.45%	100%	98.90%	99.02%
cordic	103.88%	98.84%	97.89%	95.97%	117.61%	100%	99.10%	99.63%
ex5	130.17%	93.70%	85.79%	74.75%	236.76%	100%	97.97%	94.68%
misex2	136.49%	93.68%	85.81%	74.54%	265.41%	100%	98.79%	95.32%
misex3	103.47%	99.54%	98.52%	96.79%	115.74%	100%	99.80%	98.90%
rd84	103.77%	99.14%	98.07%	95.66%	117.11%	100%	99.80%	97.50%
seq	103.30%	99.32%	98.86%	96.89%	114.94%	100%	99.80%	99.47%
table3	101.89%	99.64%	99.39%	98.62%	108.55%	100%	99.88%	99.50%
table5	101.99%	99.48%	99.19%	98.23%	109.00%	100%	99.89%	99.50%
Avg.	**107.86%**	**98.40%**	**96.59%**	**93.47%**	**135.65%**	**100%**	**99.39%**	**98.78%**

The area overhead and reliabilities of circuits based on the application of the improved DMR-CEL technique without and with C-element protection are shown in Table 4.4. In comparison with the results obtained for the proposed DMR technique in Table 4.1, the proposed DMR technique achieves better results as compared to DMR-CEL with un-protected C-element. However, the reliability of circuits based on DMR-CEL technique with protected C-element significantly improve as evident from Table 4.4. DMR-CEL with protected C-element offers 100% reliability against a single fault and achieves better reliability as compared to TMR with voter protection with significantly less area overhead. It also achieves slightly better reliability than the one based on combined application of the proposed DMR and STR techniques with 99.5% protection threshold for similar area overhead. However, due to the complexity of the C-element, the impact on performance is higher in comparison to the proposed DMR technique where only a masking 2-input AND or 2-input NAND gate is added at each output.

4.4 Conclusion

In this chapter, a soft error tolerant combinational circuit based on double modular redundancy is proposed. The technique is based on identifying the probability of occurrence of logic values “0” and “1” at each primary output of a circuit. Based on this, each output is synthesized in either the true or the complement form, and is then duplicated and a masking AND or NAND gate is used to combine the two duplicate outputs. The technique achieves higher circuit radiabilities than TMR without voter protection with significantly lower area overhead. It is also demonstrated that the combined application of the proposed DMR technique with the STR technique achieves comparable circuit reliabilities with significantly lower area overhead in comparison to TMR with fully protected voters.

Furthermore, an improved DMR based on the use of C-element (DMR-CEL) to combine the duplicate outputs is also proposed. This scheme applies redundancy by implementing the original and duplicated logic with majority phase. Reliabilities of circuits based on the proposed DMR are higher than those obtained based on DMR-CEL without C-element protection. Reliabilities of circuits based on DMR-CEL with protected C-element are similar to those obtained based on TMR with voter protection with significantly lower area overhead. It also achieves slightly better reliabilities than the combined application of the proposed DMR technique and STR technique for similar area overhead with a protection threshold of 99.5%. The advantage of the proposed DMR technique over DMR-CEL is that it uses primitive gates as masking gates which have lower impact on performance in comparison to the use of C-element.

CHAPTER 5

IMPLICATIONS BASED FAULT TOLERANCE TECHNIQUE FOR COMBINATIONAL CIRCUITS

In this chapter, an integrated soft error tolerance technique based on logical implications and transistor sizing is proposed. A set of source and target nodes with predefined thresholds are selected and implications between these nodes are extracted. Then, the impact of adding a functionally redundant wire (FRW) due to each implication is evaluated. This is done based on identifying an implication path and the gates along the implication path whose detection probabilities will be reduced due to adding the implication FRW. Then, the gain of an implication is computed in terms of reduction in fault detection probabilities of gates along an implication path. The implication with the highest gain is selected. The process is repeated until the gain is less than a predetermined threshold. The proposed implication-based fault tolerance technique enhances the circuit reliability with minimal area overhead based on enhancing logical masking. However, its effectiveness depends on the existence of such relations in a circuit and can enhance circuit reliability upto a certain level. To enhance circuit reliability to any required level, selective-transistor redundancy (STR) based technique is then applied. This technique is based on providing fault tolerance for individual transistors with high detection probability based on transistor duplication and sizing. Experimental results show that the proposed integrated fault tolerance technique achieves similar reliability in comparison to applying STR alone with lower area overhead.

The proposed implication-based fault tolerance technique is similar to the technique [63] in estimating the value of adding a FRW due to an implication. However, it is based on estimating the impact on fault detection probabilities. In addition, detailed algorithm for identifying an implication path and the gates along the path is provided. Furthermore, the fanouts of gates along an implication path and their reachability to the target gate is taken into consideration during the estimation of the impact of adding an implication FRW on detection probabilities of gates along the implication path. FRWs due to implications are added either at the target gate or at its fanout gate allowing both invert and non-invert implications without any restriction to the number of FRWs added at any gate. This is justified as the value of each implication is estimated and a FRW due to an implication is added if the improvement in detection probabilities of gates along an implication path is greater than a given threshold. Also, implications are sorted in descending order of their values and the implication with the highest value is added first. After adding an implication, values of all other implications are updated and the implication with the highest value is then selected.

The rest of the chapter is organized as follows. In Section 5.1 the motivation of the proposed method is presented, in Section 5.2 the proposed fault tolerance technique is described, an illustrative example is given in Section 5.3, experimental results are elaborated in Section 5.4 and the paper is concluded in Section 7.1.

5.1 Motivation

In this section, the basic principles of logic implications are briefly explained. Then, the differences between proposed technique and other related techniques are highlighted. These are illustrated based on the examples given in [62].

A logic implication from a source gate S to a target gate T indicates a fine-grained invariant relationship that a logic value assignment at the source gate will always enforce a consistent value assignment at the target gate i.e., $(S = u) \Rightarrow (T = v)$, where $u, v \in (0, 1)$. Such forced relations can be used to mask SETs. To illustrate this, let us consider the example circuit in Fig. 5.1. For the target gate G8, one obvious implication is $(e = 1) \Rightarrow (G8 = 0)$. This is because when $e = 1$, then $G_3 = 1$ and $G_8 = 0$. Now, if a SET changes the value of any gate

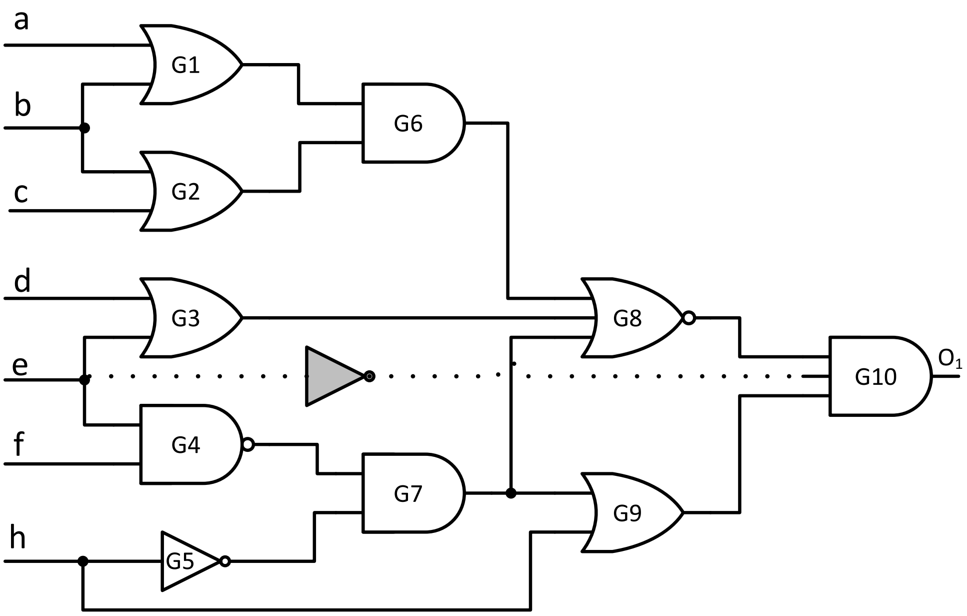


Figure 5.1: An example circuit.

along the implication path i.e., G_3 or G_8, then the faulty value will be $G_8 = 1$. However, the fault will only be observable at the primary output O_1 only when $G_9 = 1$. Now, if an inverted FRW, dotted line in Fig. 5.1, is added to realize the implication $(e = 1) \Rightarrow (G8 = 0)$, then the SET on gate G_3 or G_8 will always be masked at gate G_{10} before being latched at the primary output O_1. Similarly, a fault hitting the added inverter flipping its value from 0 to 1 will also be masked by G_8 and will not propagate at the circuit output. However, protection is not guaranteed when $e = 0$.

To reduce the impact on performance, the authors in [63] proposed to limit the addition of FRWs to two wires for any target gate. However, this could have a limitation on enhancing circuit fault tolerance. Hence, the proposed work is not limited to such a restriction. For example, in the circuit of Fig. 5.1, there are two additional implications to the target gate G_8 i.e., $(b = 1) \Rightarrow (G8 = 0)$

and $(h = 0) \Rightarrow (G8 = 0)$. So, realizing these three implications by adding FRWs to the masking gate G_{10} will improve the SET of gates along three different implication paths. A masking gate is a gate that is driven by the target gate. In the hindsight, now the masking gate G_{10} becomes a 5-input AND gate. If there exists a technology limitation or performance impact, then this gate can be split into smaller gates.

Determination of gates along an implication path is not a trivial task. Although the authors in [63] used an implication path in their analysis, no details were given for determining an implication path and the gates along the path. An implication might propagate from the source gate forward to the target gate, might traverse backward then forward or might not have a single path. In Fig. 5.1, the implication path for the implication $(b = 1) \Rightarrow (G8 = 0)$, is $b \rightarrow \{G_1, G_2\} \rightarrow G_6 \rightarrow G_8$. However, the implication path for the implication $(G_4 = 0) \Rightarrow (G8 = 0)$, is $G_4 \rightarrow e \rightarrow G_3 \rightarrow G_8$. Furthermore, the implication $(h = 0) \Rightarrow (G8 = 0)$ does not have a single implication path. The detection probabilities of gates G_3, G_4, G_5 and G_7 will partially improve depending on the applied input values. For example, when $e = 1$ and $f = 0$, detection probabilities of these gates will not be improved by adding a FRW due to this implication at the masking gate G_{10}. This is because faults on these gates are already masked by having a 1 at the output of G_3. In this chapter, an algorithm for identifying an implication path and the gates whose detection probabilities are impacted due to adding a FRW based on a given implication is proposed.

The techniques in [62] and [63] add FRWs only to the masking gate driven by the target gate. In this paper, the proposed technique adds FRWs either to the masking gate or to the target gate after evaluating the implication type and the type of masking gate and the target gate.

Addition of FRWs improve the reliability of logic circuits by increasing logical masking. However, the masking gates will remain unprotected and other techniques need to be employed to enhance their fault tolerance. Furthermore, fault tolerance enhancement is constrained by the existence of such relations. In order to further improve circuit reliability, the selective transistor sizing (STR) technique discussed in Chapter 3 is employed, which is based on transistor duplication and sizing to provide protection of sensitive transistors with high detection probability. The combined application of the proposed implication based fault tolerance technique and the STR technique requires lesser area overhead as compared to applying STR alone.

5.2 Proposed Fault Tolerance Technique

The proposed fault tolerance technique is based on the integration of an implication based fault tolerance technique and selective transistor fault tolerance technique. Firstly, a brief discussion is made regarding the proposed selective-transistor redundancy (STR) technique from Chapter 3. Then, the proposed implication based fault tolerance technique is presented.

5.2.1 Selective Transistor Redundancy Based Fault Tolerance Technique

The transistor level fault tolerance technique in Chapter 3 protects individual sensitive transistors of a circuit. A sensitive transistor is a transistor whose soft error detection probability is relatively high. A transistor is protected if a required criteria such as circuit probability of failure or area overhead is not met and the probability of failure of the transistor is greater than the probability of failure of other transistors in the circuit. The protection is applied to a transistor by duplicating and sizing one or more transistors necessary for providing the protection against particle strikes. This approach is effective in achieving a target circuit reliability with lower area overhead as compared to other related fault tolerance techniques.

5.2.2 Implications Based Fault Tolerance Technique

The proposed implication-based fault tolerance technique is shown in Algorithm 3. Initially, the circuit is simulated with 1 million random input patterns using HOPE [83] to get the probability of having a value of 1 and the probability of having a value of 0 and stuck-at (i.e., stuck-at-0 and stuck-at-1) fault detection probabilities for all gates in the circuit.

Next, the set of source gates **S** and the set of target gates **T** are identified. To reduce the computation time and to focus on identification of potentially useful implications, only source gates that have probability of output value i.e, probability of zero (P0) or probability of one (P1), greater than or equal to $Th1$ are selected. It is observed from simulations that source gates that have P0 or P1 ≥ 0.3 are good candidates for implication FRW addition. Similarly, target gates that have fault detection probability $\geq Th2$ are selected. From simulations, a good value of $Th2$ is found to be 0.4. Then, implications are identified between gates in **S** and **T** using any implication learning technique such as direct implications learning [87] and indirect implications learning [88–90].

Once implications are discovered, Algorithm 3 then evaluates each implication by identifying the implication path for each implication and computing the gain in the form of reduction of gate stuck-at fault detection probabilities for gates along the implication path. Then the the implication with the best gain is selected and a FRW based on the implication is added. The gain of all remaining implications is then updated. The process of implication FRW addition is repeated until the best implication gain $Gain < 0.02$. The details of each step of Algorithm 3 are elaborated in the following subsections.

Implications Learning

There are two main methods to identify implications in a circuit and any one of them can be employed in the proposed technique.

Direct implications are identified by assigning a value at a gate and iteratively performing backward justification and forward propagation until every unjustified gate is either justified, or there exist more than one possible justification for it. FAN [87] is a well know algorithm to discover direct implications in a circuit.

Algorithm 3 : Implications Based Fault Tolerance Technique

```
$Th1$: Gate probability threshold
$Th2$: Gate Stuck-at fault detection probability threshold
$\mathbf{\Phi}$: Set of implications b/w $\mathbf{S}$ and $\mathbf{T}$
$G_{s_i} \rightarrow G_{t_i}$: Gates along the $i^{th}$implication path
$Gain$: Path gain due to implication FRW addition

Simulate circuit to get initial gate value and stuck-at fault detection probabilities
Identify set of source gates, $\mathbf{S}$, with $Prob(s_{i0})$ or $Prob(s_{i1}) \geq Th1$
Identify set of target gates, $\mathbf{T}$, with $Pdet(t_i) \geq Th2$
Extract implication relations between gates in $\mathbf{S}$ and $\mathbf{T}$
//Implications evaluation
for (each implication $\phi_i \in \mathbf{\Phi}$) do
    Identify implication path                              ▷ Algorithm 6 & 7
    Compute implication $Gain$                                  ▷ Eqn. 5.3
end for
//End of Implications evaluation block
repeat
    //Implications addition
    Select implication $\phi_i$ with best gain
    if ($Gain \geq 0.02$) then
        Add FRW for implication $\phi_i$                   ▷ Algorithm 4 & 5
        Update stuck-at fault detection prob. $G_{s_i} \rightarrow G_{t_i}$
        Update $Gain$ for all remaining implications
    end if
    //End of Implications addition block
until ($Gain < 0.02$)
```

The identification of *indirect implications* is much harder than the direct implications and implications are identified through learning by injecting temporary values at certain gates in the circuit and then examine their logical consequences [88–90].

Rules for Addition of Implications FRWs

Once an implication is selected, a FRW is added between the source and the destination gates. Previous methods of implications FRWs addition [62, 63] only

Algorithm 4 : FRW Addition Rule for $(S = u) \Rightarrow (T = 1)$

```
M : Masking gate; $M_i$: Set of inputs of Masking gate
T: Target gate; $T_i$: Set of inputs of Target gate
$\widehat{M}$: New masking gate; $\widehat{T}$: New target gate
$FO_T$: Number of fanouts of target gate
$\bigoplus$: Logic XOR

if (M = AND/NAND or $FO_T > 1$ or M=$\phi$) then
    if (T = NOT) then
        $\widehat{T} \leftarrow$ NAND($T_i$, $(S \bigoplus u)$)
    else if (T = NAND) then
        $\widehat{T} \leftarrow$ NAND($T_i$, $(S \bigoplus u)$)
    else if (T = OR) then
        $\widehat{T} \leftarrow$ OR($T_i$, $(S \bigoplus \overline{u})$)
    end if
else
    if (M = NOT) then
        $\widehat{M} \leftarrow$ NOR($M_i$, $(S \bigoplus \overline{u})$)
    else if (M = OR) then
        $\widehat{M} \leftarrow$ OR($M_i$, $(S \bigoplus \overline{u})$)
    else if (M = NOR) then
        $\widehat{M} \leftarrow$ NOR($M_i$, $(S \bigoplus \overline{u})$)
    end if
end if
```

add FRW to the masking gate. A masking gate is a gate that is connected to the output of the target gate of an implication. In the proposed technique, a FRW can be added to either the masking gate or the target gate. When an implication is added to the masking gate, the masking gate is considered as the new target gate. Algorithm 4 and Algorithm 5 illustrate all rules required to add a FRW due to an implication $(S = u) \Rightarrow (T = v)$.

Algorithm 4 illustrates the implication FRW addition rules when a source value u implies a target value $v = 1$. First, the algorithm checks whether the FRW can be added to the masking gate or not. The FRW cannot be added to the masking gate if the target gate is connected to an output (i.e., there is no making gate) or if the masking gate is of type AND/NAND. In addition, a FRW to a masking gate is not added if the fanout of the target gate is > 1. This is because soft errors on the target gate and other gates on the implication path may prorogate across other fanout branches. If the FRW cannot be added to the masking gate, then an attempt is made to add the FRW to the target gate. If the target gate is of type NOT, then it will be changed into a NAND gate with the input of T (i.e., T_i) and the input $S \bigoplus u$. The XOR operator $\bigoplus$ performs the inversion of the source gate S when required. For example, if $S = u = 1$ and $T = 1$ and the target gate is an inverter, then the FRW from source S must first be inverted before being added to the NAND gate. The relation $(S \bigoplus u)$ will become $S \bigoplus 1 \Rightarrow \overline{S}$, which satisfies the condition for adding the FRW from source gate S to target gate T. Similarly, when the target is a NAND gate, then an extra input with an inverted or non-inverted source gate $(S \bigoplus u)$ is added to the target NAND gate. Similarly, if the target gate is an OR gate, then an extra input with an inverted or non-inverted source gate $(S \bigoplus \overline{u})$ is added to the target OR gate.

However, if the FRW can be added to the masking gate, then it is added as follows. For $T = 1$, the FRW can only be added to the masking gate if it is of type NOT, OR or NOR. Other cases will require the insertion of an OR gate, which is avoided in this work. If the masking gate is a NOT gate, then it is converted to a NOR gate with new inputs M_i (i.e., “T”) and $(S \bigoplus \overline{u})$. If the masking gate is of type OR or NOR gate, then the inverted or non-inverted source gate $(S \bigoplus \overline{u})$

Algorithm 5 : FRW Addition Rule for $(S = u) \Rightarrow (T = 0)$

```
M : Masking gate; $M_i$: Set of inputs of Masking gate
T: Target gate; $T_i$: Set of inputs of Target gate
$\widehat{M}$: New masking gate; $\widehat{T}$: New target gate
$FO_T$: Number of fanouts of target gate
$\bigoplus$: Logic XOR

if (M = OR/NOR or $FO_T > 1$ or M=$\phi$) then
    if (T = NOT) then
        $\widehat{T} \leftarrow$ NOR($T_i$, ($S \bigoplus \overline{u}$))
    else if (T = NOR) then
        $\widehat{T} \leftarrow$ NOR($T_i$, ($S \bigoplus \overline{u}$))
    else if (T = AND) then
        $\widehat{T} \leftarrow$ AND($T_i$, ($S \bigoplus u$))
    end if
else
    if (M = NOT) then
        $\widehat{M} \leftarrow$ NAND($M_i$, ($S \bigoplus u$))
    else if (M = AND) then
        $\widehat{M} \leftarrow$ AND($M_i$, ($S \bigoplus u$))
    else if (M = NAND) then
        $\widehat{M} \leftarrow$ NAND($M_i$, ($S \bigoplus u$))
    end if
end if
```

is just added to the masking gate as an extra input.

Algorithm 5 illustrates the FRW addition rules when a source value u implies a target value of $v = 0$. The discussion regarding Algorithm 4 can be well extended for the rules given in Algorithm 5.

Implication Path Identification

Once an implication is identified, the next important step is to evaluate the impact of adding a FRW due to the implication. Almukhaizim et al. [62] used simulation to quantify the impact on gates sensitivity when an implication FRW is added. Although they have tried to reduce the number of simulated input values, performing simulation to evaluate the value of each implication is a very time consuming process and will not scale for large circuits. Rather than simulating the circuit to evaluate the value of adding an implication FRW, Zhou et al. [63] employed a relation to estimate the soft error rate (SER) of gates along an implication path. However, they didn't elaborate on how an implication path is identified. We will show next the details of how an implication path is determined and the gates along the implication path whose fault detection probabilities are reduced when an implication FRW is added.

Two algorithms are proposed to determine an implication path. The first algorithm, value propagation (VP), traverses gates and marks them from source gate S to target gate T. Once the gates are marked, the implication path identification (IPI) algorithm then determines the implication path. The VP and IPI algorithms are illustrated in Algorithm 6 and Algorithm 7, respectively.

Value Propagation (VP) Algorithm Algorithm 6 illustrates the steps of the VP algorithm. C and NC denote that for the current gate CG under consideration, an input of the gate is having either a controlling or non-controlling logic value. For example, an input value equal to 0 is a controlling value to a NAND gate, whereas a value equal to 1 is a non-controlling input value. The propagating P and non-propagating NP markings hold information regarding the probable implication path. IP indicates the inversion polarity for the current gate, which is 1 for NOT, NAND and NOR gates, while 0 for AND and OR gates. Algorithm 6 starts by identifying all gates reachable to the target gate i.e., gates in the cone logic of T and gates reachable to the source gate. Then, the output of the source gate is marked with the value P_u. If the value $u \bigoplus IP$ is a non-controlling value for the source gate, then all its inputs are marked with $P_{u \bigoplus IP}$. Then, all marked inputs of the source gate reachable to the target gate are added to *processQ*. Furthermore, all gates in the fanout of the source gate which are reachable to the target gate are added to *processQ*. The algorithm then processes gates in *processQ* until the queue becomes empty.

Next, a gate, CG, is selected from *processQ*. If inputs of CG are not marked and $u \bigoplus IP = NC$, then all inputs of CG are marked with the value $P_{u \bigoplus IP}$ and inputs reachable to the target gate are added to *processQ*. Otherwise, if any of the inputs of the CG is propagating with controlling value, the gate will be marked with $P_{C \bigoplus IP}$. For example, if one of the inputs of an OR gate is marked with P_1, then the gate will be marked with P_1. However, for a NOR gate, if one of the inputs is marked with P_1, then the gate will be marked with P_0. If all inputs to a gate are propagating P with non-controlling value NC, then it will be marked with $P_{NC \bigoplus IP}$. The path will switch from P to NP if none of the inputs of the CG has a controlling value and not all of its inputs have non-controlling values. Once the implication path becomes NP, it will remain NP. If the marking computed for CG is different from its previous marking, fanouts of CG reachable to T and not reachable to S will be added into *processQ*

Implication Path Identification (IPI) Algorithm The implication path identification (IPI) algorithm is applied after VP algorithm. The purpose of IPI

Algorithm 6 : Value Propagation (VP) Algorithm

```
$P_0$ : Propagating 0; $P_1$ : Propagating 1;
$NP_0$: Non-propagating 0; $NP_1$ : Non-propagating 1;
$C$ : Controlling value; $NC$: Non-controlling value
$u$: Source gate value
$IP$: Set to 1 when a gate is inverting (i.e., NOT, NAND, NOR), otherwise set to 0
$processQ$ : Queue to process gates
$CG$ : Current Gate being processed
$M_{CG}$: Marking applied to the output of current gate

Mark all gates reachable to the target gate $T$
Mark all gates reachable to the source gate $S$
$CG$ = Source gate
$M_{CG} = P_u$
if ($u \bigoplus IP = NC$) then
    Mark all inputs of $CG$ with the value $P_{u \bigoplus IP}$
    Add all inputs of $CG$ reachable to $T$ to $processQ$
end if
Add all fanouts of $CG$ reachable to $T$ to $processQ$
while ($processQ \neq$ NULL) do
    $CG = pop(processQ)$;$u$ is the value if $CG$
    if (inputs of $CG$ are not marked and $u \bigoplus IP = NC$) then
        Mark all inputs of $CG$ with the value $P_{u \bigoplus IP}$
        Add all inputs of $CG$ reachable to $T$ to $processQ$
    else if (any input to $CG = P_C$) then
        $M_{CG} \leftarrow P_{C \bigoplus IP}$
    else if (all inputs to $CG = P_{NC}$) then
        $M_{CG} \leftarrow P_{NC \bigoplus IP}$
    else if (any input to $CG = P_{NC}$) then
        $M_{CG} \leftarrow NP_{NC \bigoplus IP}$
    else if (any input to $CG = NP_C$) then
        $M_{CG} \leftarrow NP_{C \bigoplus IP}$
    else if (all inputs to $CG = NP_{NC}$) then
        $M_{CG} \leftarrow NP_{NC \bigoplus IP}$
    end if
    if (New marking of $CG \neq$ It's Previous marking) then
        add fanouts of $CG$ reachable to $T$ and not reachable to $S$ into $processQ$
    end if
end while
```

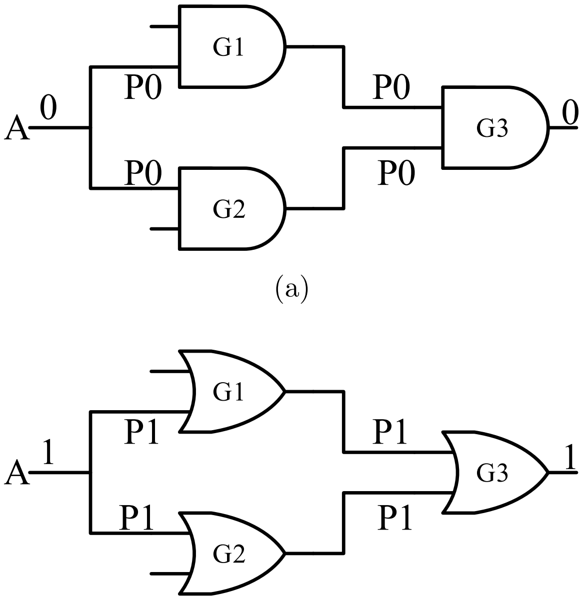
algorithm is to determine the implication path and the gates along the path whose detection probabilities will be impacted by adding the implication FRW. Algorithm 7 highlights the steps of the IPI algorithm.

The *processQ* holds the list of gates to be processed in each iteration, whereas $Path_G$ list contains gates that are part of the implication path. The set of rules established in Algorithm 7 will now be explained. The algorithm starts with adding the target gate to *processQ*. If only one input to a gate is marked with P_C, then that input is added to the *processQ* and the process continues. It should be observed that if two or more inputs of a gate are marked with P_C, then the inputs of this gate are not added to *processQ*. Fig. 5.2 illustrates this case. For both circuits, the value of $A = 0$ or $A = 1$ is propagated to the output. Therefore, if 2 or more inputs of the target gate are marked with P_C, then there is no need to protect the gates along the implication path(s) as faults on gates along one path will be masked at the target gate due to the controlling logic value of the other input. In case if two or more inputs are marked with NP_C as shown in Fig. 5.3, then, there is also no need to protect the gates along the implication path(s) due to the masking at the target gate provided by the controlling value of other input.

Fig. 5.4 illustrates the application of rule 2. If all inputs of the target gate are marked with P_{NC}, then all the inputs are added to the $Path_G$ list. Therefore, both $G1$ and $G2$ are added to the $Path_G$ list in this case. Other rules will be explained through illustrative examples given in the next section.

Algorithm 7 : Implication Path Identification (IPI) Algorithm

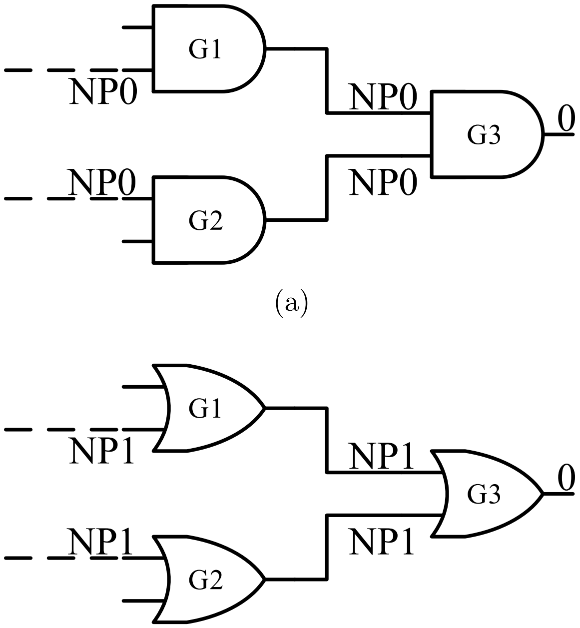
```
$P_0$ : Propagating 0; $P_1$ : Propagating 1;
$NP_0$: Non-propagating 0; $NP_1$ : Non-propagating 1;
$C$ : Controlling value; $NC$: Non-controlling value
$PI$: Primary input
$processQ$ : Queue to process gates
$CG$ : Current Gate being processed
$Path_G$: Implication path gates

Add $T$ to $processQ$
while ($processQ \neq$ NULL) do
    $CG = pop(processQ)$
    if ($CG \neq S$ and $CG \neq T$ ) then
        Add $CG$ to $Path_G$
    end if
    if (One input of $CG$ is marked with $P_C$) then
        if (Input marked with $P_C$ is not PI and $\notin processQ$) then
            Add input to $processQ$
        end if
    else if (All inputs of $CG$ are marked with $P_{NC}$) then
        for (all inputs) do
            if (Input is not PI and $\notin processQ$) then
                Add input to $processQ$
            end if
        end for
    else if (One input of $CG$ is marked with $NP_C$) then
        if (Input marked with $NP_C$ is not PI and $\notin processQ$) then
            Add input to $processQ$
        end if
    else if (All inputs to $CG = NP_{NC}$) then
        for (all inputs) do
            if (Input is not PI and $\notin processQ$) then
                Add input to $processQ$
            end if
        end for
    end if
end while
Return $Path_G$
```



(a)

(b)

Figure 5.2: Illustration of having two inputs with P_C markings.



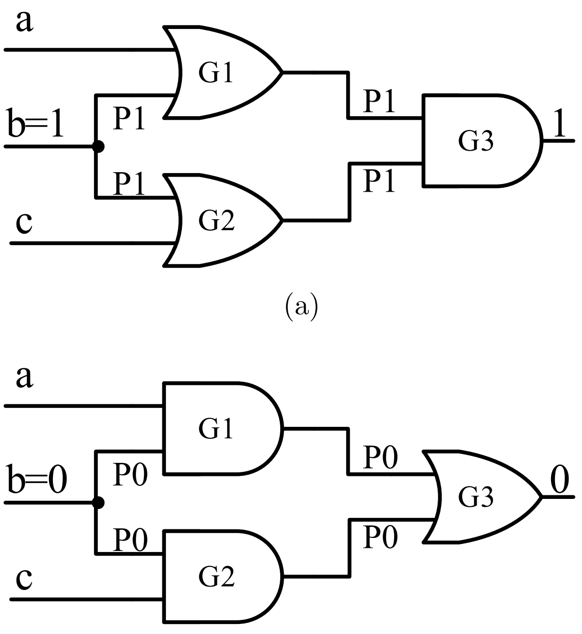
(a)

(b)

Figure 5.3: Illustration of having two inputs with NP_C markings.

Evaluation of Implication FRW Gain

Once the implication path is determined, the next step is to evaluate the value of adding a FRW due to an implication path based on updating the stuck-at



(a)

(b)

Figure 5.4: Illustration of having two inputs with P_{NC} markings.

fault detection probabilities of gates along the implication path. If the value v of a gate G_i along an implication path is "0", then the stuck-at-1 (sa1) fault detection probability of G_i is improved/modified. However, if the value of the gate is "1", then the stuck-at-0 (sa0) fault detection probability of gate G_i is improved/modified. The following equation is used to update or modify the stuck-at fault detection probability of gate G_i along an implication path.

$$\widehat{G_{isa\overline{v}}} = G_{isa\overline{v}} \times (1 - P_u) \times PRP \tag{5.1}$$

Where P_u denotes the probability that the source gate S has a value of u and PRP is the percentage of reachable paths from gate G_i to the target gate T computed using the following relation:

$$PRP = \frac{\#\ of\ reachable\ paths\ G_i \rightarrow T}{\#\ of\ paths\ G_i \rightarrow PO} \tag{5.2}$$

Where "$\#\ of\ paths\ G_i \rightarrow PO$" denote the total number of paths from gate G_i to primary output(s), "*reachable path*" denotes the number of paths from gate G_i that are reachable to target gate T. Eqn. 5.2 is used to estimate the percentage improvement in the fault detection probability of gate G_i as fault detection will be improved for paths propagating through T but not across other paths not propagating through T.

For each implication, Eqn. 5.1 is applied to update the stuck-at fault detection probabilities of gates along the implication path. The gain of an implication is then computed using Eqn. 5.3.

$$Gain_{S\rightarrow T} = \sum_{\forall G_i \in S\rightarrow T} G_{isa} - \sum_{\forall G_i \in S\rightarrow T} G_{i\widehat{sa}} \tag{5.3}$$

Where $G_{i\widehat{sa}}$ denotes the new fault detection probability and G_{isa} denotes the old fault detection probability of gate G_i.

5.3 Illustrative Examples

In this section, two examples are discussed in detail that encompass all aspects of the proposed method discussed in Section 5.2. Consider the example shown in Fig. 5.5, where an implication exists between source gate $G4$ and the target gate $G3$. The masking gate in this example is gate $G10$. The first step is to determine whether the FRW will be added to the target gate $G6$ or the masking gate. Since the implication is of type $(G4 = 0) \Rightarrow (G3 = 0)$ and the masking gate $G10$ is an AND gate, therefore the FRW will be added to the masking gate. The new target gate T is now $G10$. The circuit with the added FRW (dotted line) is also shown in Fig. 5.5. The masking gate $G10$ now becomes a 3-input AND gate with an extra input connected to the source gate $G4$.

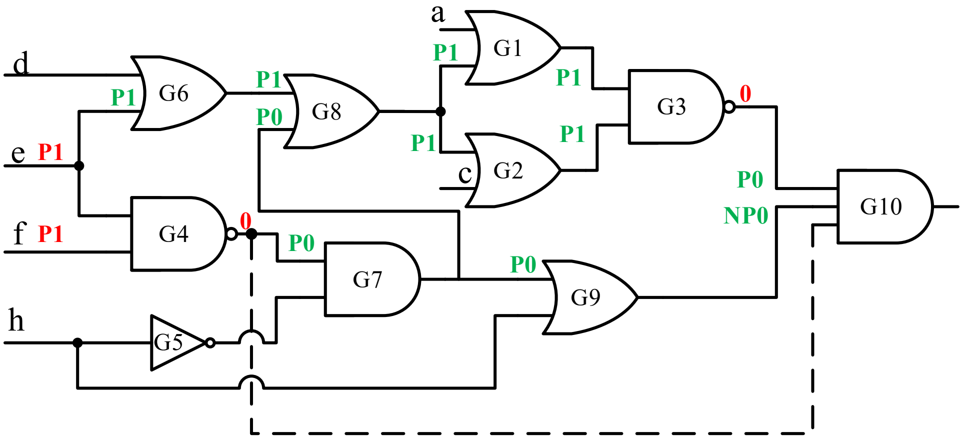


Figure 5.5: Circuit with Implication $(G4 = 0) \Rightarrow (G6 = 0)$.

Once the FRW is added, the next step is to determine the implication path. Starting from the source gate, $G4$ is marked as propagating P with value 0 i.e., P_0. Since source gate $G4$ is a NAND gate, the value of IP is equal to "1". The condition in line 14 of Algorithm 6 i.e., $(0 \bigoplus 1 = 1(NC))$, is satisfied, therefore all inputs of $G4$ are marked with $P_{0 \bigoplus 1} = P_1$. Primary inputs e and f are also added to the *processQ* as they both have a reachable path to the target $G10$. So, initially, $G4$ is marked as P_0 and both primary inputs e and f are marked as P_1. $G4$ has only one fanout gate $G7$, which is reachable to the target $G10$, therefore it is also added to the *processQ* due to the condition in line 18 of Algorithm 6.

The *processQ* now contains values $\{e, f, G7\}$.

Next, e is selected as CG from the *processQ*. Since e is a primary input, all the conditions from line 21 to line 32 are skipped. However, due to the statement in line 35 of Algorithm 6, $G6$ is added to the *processQ*, because it is the only fanout of e reachable to the target gate $G10$. The other fanout of e is the source $G4$, therefore it is not added to the *processQ*. The *processQ* now consists of $\{f, G7, G6\}$. In the next iteration, f is selected as CG. Again, all marking conditions are skipped as f is a primary input. Since the only fanout for f is the source $G4$, nothing is added to the *processQ*. So, the *processQ* now contains $\{G7, G6\}$. In the next iteration, $G7$ is selected as CG. Since one of the inputs of $G7$ is P_0, i.e., satisfying the condition in line 24 of Algorithm 6, $G7$ is marked as P_0 i.e., $M_{G7} = P_{C=0 \bigoplus IP=0} = P_0$. Both fanouts of $G7$ are added to the *processQ* as they are reachable to the target gate $G10$. The updated *processQ* now consists of $\{G6, G8, G9\}$.

Gate $G6$ is marked as P_1 as its input is marked with P_1. Gate $G8$, which is the fanout of $G6$ is now added to the *processQ*. Updated *processQ* is now $\{G8, G9, G8\}$. The inputs to $G8$ are marked as P_1 due to $G6$ and P_0 due to $G7$, therefore due to the condition in line 24 of Algorithm 6, $G8$ is marked as P_1. After both fanouts of $G8$ are added, the updated *processQ* becomes $\{G9, G8, G1, G2\}$. One of the inputs to gate $G9$ is marked as P_0 due to $G7$, therefore, due to the condition in line 28 of Algorithm 6, the output of $G9$ is marked as non-propagating NP with value 0 i.e., $M_{G9} = NP_{NC=0 \bigoplus IP=0} = NP_0$. Nothing is added to the *processQ* as the fanout of $G9$ is the target gate $G10$. In the next iteration, $G8$ is selected again but its marking will stay the same i.e., $M_{G8} = P_1$. Thus, its fanouts $G1$ and $G2$ are not added to the *processQ*. Both $G1$ and $G2$ have inputs marked with P_1, therefore both of them are marked as P_1. $G3$ is now added to the *processQ*. Finally, $CG = G3$ is selected for processing. All inputs of $G3$ are P_1 i.e., propagating with non-controlling value, therefore, due to the condition in line 26 of Algorithm 6, $G3$ is marked as P_0 i.e., $M_{G3} = P_{NC=1 \bigoplus IP=1} = P_0$. Nothing is added to the *processQ* as the fanout of $G6$ is the target $G10$. The algorithm terminates as the *processQ* is now empty. The gate markings due to VP algorithm are also shown in Fig. 5.5

Once the VP algorithm is finished, the IPI Algorithm (Algorithm 7) is applied to determine the implication path and the gates protected by adding the implication FRW. The IPI algorithm starts from the target $G10$. Starting from $G10$, $G3$ is selected and is added to the *processQ*, as it has $P0$ marking, in accordance with the rule mentioned in line 15 of Algorithm 7. Then, $G3$ is processed and is added to $Path_G$. Since both inputs of $G3$ are marked with P_1, P with non-controlling input $NC = 1$, therefore both $G1$ and $G2$ are added to *processQ* in accordance with the rule mentioned in line 19 of Algorithm 7. Then, $G1$ and $G2$ get processed and get added to $Path_G$ and $G8$ gets added to *processQ*. Then, gate $G8$ is processed and is added to $Path_G$. Since $G8$ has two inputs with markings $P1$ and $P0$, the path with the propagating value is selected i.e., $G6$. Finally $G6$ is processed and is added to $Path_G$. The algorithm now terminates as the input to $G6$ is a primary input. The implication path now consists of gates ($G6 \rightarrow G8 \rightarrow (G1, G2) \rightarrow G3$).

Once the implication path gates are identified, the next step is to update the stuck-at fault detection probabilities of gates using Eqn. 5.1. So, for gates along the implication path, $G6sa0$, $G8sa0$, $G1sa0$, $G2sa0$ and $G3sa1$ fault detection probabilities are improved, respectively. Gate $G6$ has two paths to the primary output $G10$ i.e., ($G6 \rightarrow G8 \rightarrow G1 \rightarrow G3 \rightarrow G10$) and ($G6 \rightarrow G8 \rightarrow G2 \rightarrow G3 \rightarrow G10$) and all the paths are reachable to the target gate $G10$. Therefore, PRP of $G3$ is "1". In fact, in this example, the PRP of all gates along the implication path is "1". Before adding the implication FRW, the stuck-at fault detection probabilities of gates along the implication path are $\{0.281, 0.563, 0.781, 0.781, 0.781\}$. After the addition of the implication FRW, the estimated stuck-at fault detection probabilities using Eqn. 5.1 along the implication path are $\{0.211, 0.422, 0.586, 0.586, 0.586\}$. Based on actual fault simulation using HOPE [83], the improved stuck-at fault detection probabilities are found out to be $\{0.188, 0.468, 0.656, 0.656, 0.656\}$. It is clear that the estimated values and the values computed using HOPE [83] are close to each other. The estimated implication path gain computed using Eqn. 5.3 is 0.796 while the path gain obtained using HOPE [83] simulation is 0.563.

Let us next consider the example shown in Fig. 5.6, where an implication exists between source wire h and the target gate $G3$. The masking gate in this example is gate $G10$. The FRW will be added to the masking gate and the new target gate T is now $G10$. The circuit with FRW (dotted line) is also shown in Fig. 5.6.

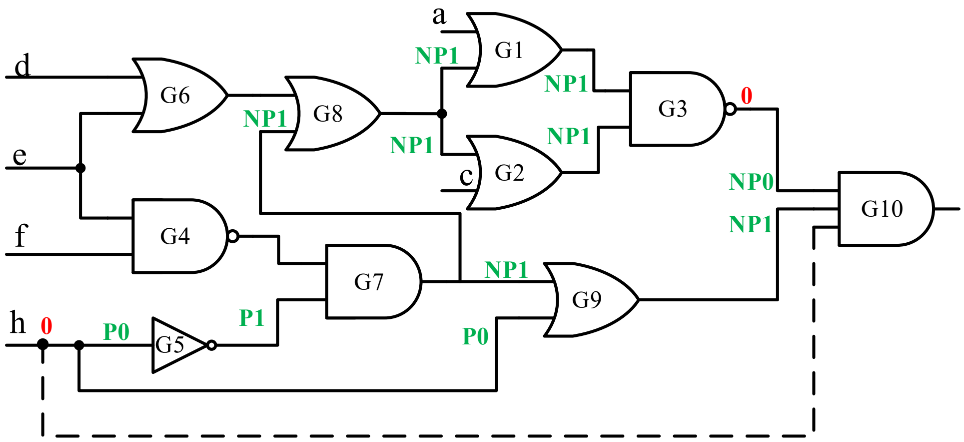


Figure 5.6: Indirect implication path discovery.

Starting from h, $processQ$ consists of $\{G5, G9\}$. When selected, $G5$ is marked with P_1 at its input is marked with P_0. When $G7$ is added, $processQ$ becomes $\{G9, G7\}$. Currently, only one input to $G9$ is marked as $P0$ due to h, therefore, $G9$ is marked NP_0 due to condition in line 26 of Algorithm 6. Since the output of $G9$ is target $G10$, nothing is added to the $processQ$. The $processQ$ now consists of $\{G7\}$ only. $G7$ is marked with NP_1 as its input is a non-controlling value. Both fanouts of $G7$ are added to the $processQ$ and it will become $\{G8, G9\}$. $G8$ is processed next and is marked with NP_1 and its fanouts are added to the queue. $processQ$ becomes $\{G9, G1, G2\}$. $G9$ is selected for the second time now and this time both its inputs are marked as NP_1 due to $G7$ and P_0 due to h. The marking of $G9$ changes from NP_0 to NP_1 due to the controlling input value from $G7$. $G1$ and $G2$ are then processed and are also marked with NP_1. The $processQ$ now consists of $\{G3\}$ only. Finally, when selected, $G3$ is marked with NP_0. VP algorithm now terminates as the target is reached and $processQ$ become empty. The marking of gates due to VP algorithm is also shown in Fig. 5.6. It must be noted that when a path becomes non-propagating NP it remain NP until the target is reached.

The IPI algorithm is now applied to determine the implication path. It can be observed from Fig. 5.6 that both inputs of $G10$ are NP. The only clue for the IPI algorithm now is to take the path with controlling value i.e., condition in line 25 of Algorithm 7. Therefore, $G3$ is selected and then added to the $processQ$ and to $Path_G$. All inputs of $G3$ have NC value, therefore, due to condition in line 29 of Algorithm 7, both $G1$ and $G2$ are added to the $processQ$ and $Path_G$. When processing $G1$ and $G2$, $G8$ is added to the path gates list $Path_G$ and $processQ$. Continuing in this fashion, $G7$ and $G5$ are also added to the $Path_G$. Algorithm 7 now terminates as the source h is reached and also the $processQ$ becomes empty. The implication path now consists of gates $(G5 \rightarrow G7 \rightarrow G8 \rightarrow (G1, G2) \rightarrow G3)$. So, for gates along the implication path, $G5sa0$, $G7sa0$, $G8sa0$, $G1sa0$, $G2sa0$ and $G3sa1$ fault detection probabilities are improved, respectively. Before FRW is added, stuck-at fault detection probabilities of gates along the implication path are $\{0, 0, 0.563, 0.781, 0.781, 0.781\}$. The improved stuck-at fault detection probabilities of implication gates computed using HOPE [83] are $\{0, 0, 0.281, 0.406, 0.406, 0.406\}$. The estimated stuck-at fault detection probabilities computed using Eqn. 5.1 are $\{0, 0, 0.281, 0.391, 0.391, 0.391\}$. The estimated path gain computed using Eqn. 5.3 is 1.452 while the actual gain obtained using HOPE [83] simulation is 1.407. It is clear that the estimated values are very close to the simulated values obtained using HOPE [83].

It should be observed that since this is an indirect implication, not all the identified gates along the implication path will be protected. This is because when $G4 = 1$, all the identified gates by the implication path will be protected by reducing their fault detection probabilities. However, when $G4 = 0$, this implies that $e = 1$ and $G6 = 1$. Thus, faults occurring on $G5$ and $G7$ are already masked by having $G6 = 1$ without adding the FRW. Thus, in this case only gates $\{G8, G1, G2, G3\}$ are protected. The IPI algorithm will identify also gates $G5$ and $G7$ although they are partially protected.

5.4 Experimental Results

In this section, the impact of the proposed technique on the area and reliability of LGSynth'91 [84] benchmarks is evaluated. The benchmarks consist of circuits with varying complexity in terms of area, number of inputs and outputs. The LGSynth'91 benchmark circuits used in this work are synthesized with single output optimization using *Espresso* [85] tool and then mapped to a library that consists of an Inverter and 2-, 3- and 4-input NAND and NOR gates using *SIS* [86] tool to get the proper gate level representation of the circuit. The reliability of a circuit is computed using the method discussed in Chapter 6

Table 5.1: Circuits reliability and area overhead based on proposed implication based fault tolerance technique.

Circuit	**Area**	**1 Fault**	**2 Faults**	**Proposed Method**					
				OH^1	**# Imp.**	**1F**^{2}	**% Red.**	**P1**^{3}	**P2**^{3}
alu4	1429.74	97.89%	95.86%	2.73%	41	97.96%	3.32%	97.90%	96.10%
apex1	4602.00	96.72%	94.20%	3.22%	148	97.50%	23.78%	97.32%	95.68%
apex2	609.96	99.20%	98.04%	1.68%	35	99.57%	46.25%	99.50%	98.70%
apex3	3025.62	96.88%	94.76%	5.26%	133	97.32%	14.10%	97.15%	96.10%
apex4	4575.48	96.20%	92.74%	3.03%	135	96.88%	17.89%	96.40%	93.80%
b12	121.68	89.10%	78.22%	12.18%	15	90.10%	9.17%	89.45%	80.22%
clip	372.84	93.24%	86.40%	4.18%	17	94.44%	17.75%	93.97%	86.40%
cordic	241.02	98.10%	96.28%	2.91%	7	98.66%	29.47%	98.20%	96.70%
ex5	977.34	93.50%	88.26%	13.49%	102	94.76%	19.38%	94.11%	90.03%
misex1	152.88	84.34%	71.70%	2.55%	3	88.89%	29.05%	88.37%	73.09%
misex2	230.88	93.20%	88.32%	7.77%	12	94.51%	19.26%	94.10%	89.57%
misex3	1886.82	97.64%	95.40%	3.06%	50	98.61%	41.10%	98.55%	96.89%
rd84	496.08	93.38%	87.22%	5.35%	30	94.69%	19.79%	94.33%	87.61%
seq	4970.94	99.05%	98.22%	0.35%	10	99.20%	15.79%	99.15%	98.76%
squar5	101.40	83.44%	69.46%	5.38%	5	86.78%	20.17%	86.26%	70.36%
table3	3475.68	98.68%	97.78%	0.96%	27	98.89%	15.91%	98.77%	98.11%
table5	3535.74	98.70%	97.72%	1.57%	46	98.91%	16.15%	98.64%	98.22%
z5xp1	251.16	87.96%	75.38%	6.83%	21	90.23%	18.85%	89.45%	75.88%
Avg.		**94.29%**	**89.22%**	**4.58%**	**46.50**	**95.44%**	**20.96%**	**95.09%**	**90.12%**

[1] Area overhead (OH) $= \left(\frac{Area\ After\ Implications\ FRWs\ Addition}{Original\ Area} - 1\right) \times 100$

[2] Single fault

[3] Prorated 1 and 2 faults

The reliability of a circuit is computed against a single fault and the prorated 1 and 2 faults for each circuit. The number of prorated faults is correlated to the area of a circuit. So, for example, if the area overhead is 131%, then the actual area is increased by 2.31 times. Therefore, 1, 2 and 5 faults in the original circuit will prorate to 2.31, 4.62 and 11.55 faults in the protected circuit. For each prorated fault, the circuit is simulated twice. For example, if the prorated faults to be injected are 4.62, then the circuit is simulated twice, first by injecting 4 faults and then by injecting 5 faults. The failure rate is then computed as $0.38 * FR(4faults) + 0.62 * FR(5faults)$. For each fault injection scenario, faults are injected randomly and simulation is performed for 5000 iterations to compute the failure rate.

The implications are applied to a circuit until an implication gain of a circuit is < 0.02. Based on simulations, it is observed that a threshold value of 0.02 offers the best compromise between area overhead and reliability improvement of a circuit. Once FRWs due to selected implications are added, the probability of failure ($\mathbf{POF_C}$) of the final circuit is then computed using Eqn. 3.7. Table 5.1 shows the circuits area overhead and reliability achieved by the proposed implication based fault tolerance technique. The column header "# Imp." denotes the number of implications applied to each circuit. The column "% Red." denotes the percentage reduction in the failure rate of each circuit. The average reliability of all circuits is improved as compared to the original circuits. The average area overhead incurred as a result of implications FRWs addition is 4.58% and the average number of implications is 46.50. Except for *alu*4 and *b*12, the $\mathbf{POF_C}$ of all the other benchmarks is reduced by at least 14%. The columns "P1" and "P2" show the reliability of circuits against prorated 1 and 2 faults, respectively. It is clear that the average reliability achieved by the proposed technique against prorated faults is better than the reliability of original circuits against 1 and 2 faults.

Table 5.2: Circuits reliability and area overhead based on STR technique (Chapter 3) with varying protection thresholds against a single fault.

Circuit	**95%**		**98%**		**99%**	
	OH	**Rel**	**OH**	**Rel**	**OH**	**Rel**
alu4	0%	97.89%	20.04%	98.44%	51.84%	99.02%
apex1	0%	96.72%	12.64%	98.10%	48.81%	99.10%
apex2	0%	99.20%	0%	99.20%	0%	99.20%
apex3	0%	96.88%	13.93%	98.05%	51.57%	99.08%
apex4	0%	96.20%	25.88%	98.30%	71.88%	99.02%
b12	86.50%	95.01%	149.02%	98.03%	197.69%	99.05%
clip	14.64%	95.00%	68.61%	98.11%	119.87%	99.08%
cordic	0%	98.10%	0%	98.10%	19.53%	99.00%
ex5	19.53%	95.62%	81.73%	98.12%	139.89%	99.02%
misex1	85.68%	95.07%	175.48%	98.01%	249.35%	99.02%
misex2	14.12%	95.24%	50.29%	98.36%	92.48%	99.20%
misex3	0%	97.64%	3.33%	98.10%	34.27%	99.18%
rd84	14.16%	95.10%	56.49%	98.02%	101.21%	99.24%
seq	0%	99.05%	0%	99.05%	0%	99.05%
squar5	107.79%	95.10%	206.15%	98.06%	286.51%	99.00%
table3	0%	98.68%	0%	98.68%	0.34%	99.24%
table5	0%	98.70%	0%	98.70%	1.70%	99.32%
z5xp1	68.22%	95.06%	143.69%	98.11%	202.88%	99.06%
Avg.	**22.81%**	**96.68%**	**55.96%**	**98.31%**	**92.77%**	**99.11%**

The proposed implication based fault tolerance technique is based on enhancing logical masking and is applied at the gate level. One limitation of logical masking techniques is that they are unable to improve the reliability of a circuit beyond a certain point. So, in order to further improve the reliability, circuit-level techniques such as gate/transistor resizing has to be employed. Therefore, the reliability of circuits in Table 5.1 is further enhanced by applying STR technique from Chapter 3. Table 5.2 shows the reliability and area overhead of circuits when STR is applied to the original benchmark circuits for 95%, 98% and 99% protection threshold. Protection threshold refers to the target reliability required to be achieved by a circuit against a single fault. In the proposed integrated approach, the implication based fault tolerance technique is applied first followed by the STR technique if the original circuit does not satisfy a required reliability.

Table 5.3 shows the circuits reliability and area overhead resulting from the application of the proposed integrated approach for 95%, 98% and 99% protection threshold. It can be observed from Table 5.3 that the area overhead is reduced for all protection thresholds in comparison to the application of the STR technique. This is due to the fact that the addition of FRWs increases logical masking of faults and hence when STR is applied, fewer transistors require protection to achieve the desired reliability. Significant area savings are achieved for $b12$, $clip$, $ex5$ and $z5xp1$ benchmarks with 95% threshold as shown in Table 5.3 in comparison to the results in Table 5.2. With 98% threshold, $alu4$, $apex1$, $apex3$, $b12$, $ex5$, and $misex2$ in Table 5.3 achieve significantly lower area overhead with similar reliability in comparison to the results in Table 5.2. Similarly, for 99% threshold, benchmarks such as $alu4$, $apex1$, $apex3$, $apex4$, $b12$, $cordic$, $ex5$, $misex2$ and $misex3$ have significantly lower area overhead as compared to their counterparts in Table 5.2, and again with similar reliability measure.

Table 5.3: Circuits reliability and area overhead based on proposed integrated approach against a single fault.

Circuit	**95%**		**98%**		**99%**	
	OH	**Rel**	**OH**	**Rel**	**OH**	**Rel**
alu4	0%	97.89%	7.23%	98.40%	29.01%	99.03%
apex1	0%	96.72%	4.26%	98.05%	31.94%	99.18%
apex2	0%	99.20%	0%	99.20%	0%	99.20%
apex3	0%	96.88%	6.17%	98.1%	37.97%	99.03%
apex4	0%	96.20%	18.19%	98.20%	57.99%	99.01%
b12	23.80%	95.11%	90.30%	98.10%	154.19%	99.09%
clip	8.45%	95.09%	58.88%	98.07%	104.13%	99.06%
cordic	0%	98.10%	0%	98.10%	13.46%	99.1%
ex5	5.26%	95.55%	58.77%	98.1%	107.86%	99.07%
misex1	77.21%	95.12%	169.29%	98.04%	253.54%	99.01%
misex2	11.26%	95.3%	33.90%	98.28%	77.18%	99.15%
misex3	0%	97.64%	1.20%	98.1%	19.49%	99.09%
rd84	16.13%	95.09%	57.31%	98.03%	98.71%	99.04%
seq	0%	99.05%	0%	99.05%	0%	99.05%
squar5	105.73%	95.10%	196.75%	98.01%	285.13%	99.02%
table3	0%	98.68%	0%	98.68%	0.20%	99.12%
table5	0%	98.70%	0%	98.70%	0.46%	99.20%
z5xp1	58.84%	95.11%	129.34%	98.09%	189.46%	99.05%
Avg.	**17.04%**	**96.70%**	**46.20%**	**98.29%**	**81.15%**	**99.08%**

5.5 Conclusion

In this chapter, an integrated fault tolerance technique based on the combined application of an implication based fault tolerance technique and selective transistor redundancy technique is proposed. An implication based fault tolerance technique has been proposed. It is based on identifying implications between a set of candidate source and target gates. Then, for each implication the gain in reduction of gate fault detection profitabilities is estimated. The implication with the highest gain is selected and its corresponding functionally redundant wire is added. The process is repeated until the gain is less than a given threshold. Experimental results show that the proposed implication based fault tolerance technique reduces the failure rate of circuits with an average of 20% with an average area overhead of less than 5%. Moreover, the integrated application of the proposed implication based fault tolerance technique and the selective transistor redundancy technique achieve significantly lower area overhead in comparison to applying the selective transistor redundancy technique alone with the same achieved reliabilities.

CHAPTER 6

RELIABILITY EVALUATION

A novel method to compute the reliability of a circuit at the gate level is proposed in this chapter. The proposed gate level method provides similar results in comparison to transistor level simulations (using SPICE) with orders of magnitude reduction in CPU time. The effect of a transient fault hitting a transistor will be observable at one of the primary outputs with a certain probability. This probability is a function of controllability probability i.e., existence of an input pattern to excite a fault, and observability probability i.e., a fault excited at a site is observable at one of the primary outputs. The motivation here is to propose a probabilistic model that captures this effect at the gate level. The proposed technique bridges the gap between circuit level simulations performed at the transistor level using SPICE and gate level simulations, which could be done using any gate level simulator. Simulations performed at the gate level make an underlying assumption that the effect of a transient fault results in a bit flip at the output of a gate or in the memory. In this realm, we propose **probability of fault**

injection, which quantifies the probability with which a fault must be injected at the gate level so that SPICE level and gate level simulation results are highly matched

A thorough case study for the 130*nm* process technology is performed to elaborate the proposed reliability evaluation scheme. The basic process related parameters used in this study are shown in Table 6.1.

6.1 Reliability Evaluation Architecture

The reliability evaluation framework, shown in Fig. 6.1, consists of two major blocks; 1) technology independent block, and 2) technology dependent block. The purpose of the technology independent block is to analyze a given benchmark circuit to compute three important parameters for all gates; 1) input pattern probability (`.ipp`) , 2) stuck-at detection probability (`.prob`), and 3) fault injection probability (`.inj`). The input patterns observable at the input of each gate along with their probability of occurrence and stuck-at fault detection probabilities are computed by performing simulation of 1 million random test vectors using the parallel fault simulator Hope [83]. The fault injection probability denotes the probability with which a fault must be injected at the gate level as a stuck-at fault. All of these parameters are saved in a database for later usage.

The technology dependent part consists mainly of the library gates comprising NAND/NOR gates with varying input configurations and an INVERTER. The purpose of this block is to observe the behavior of different process technologies,

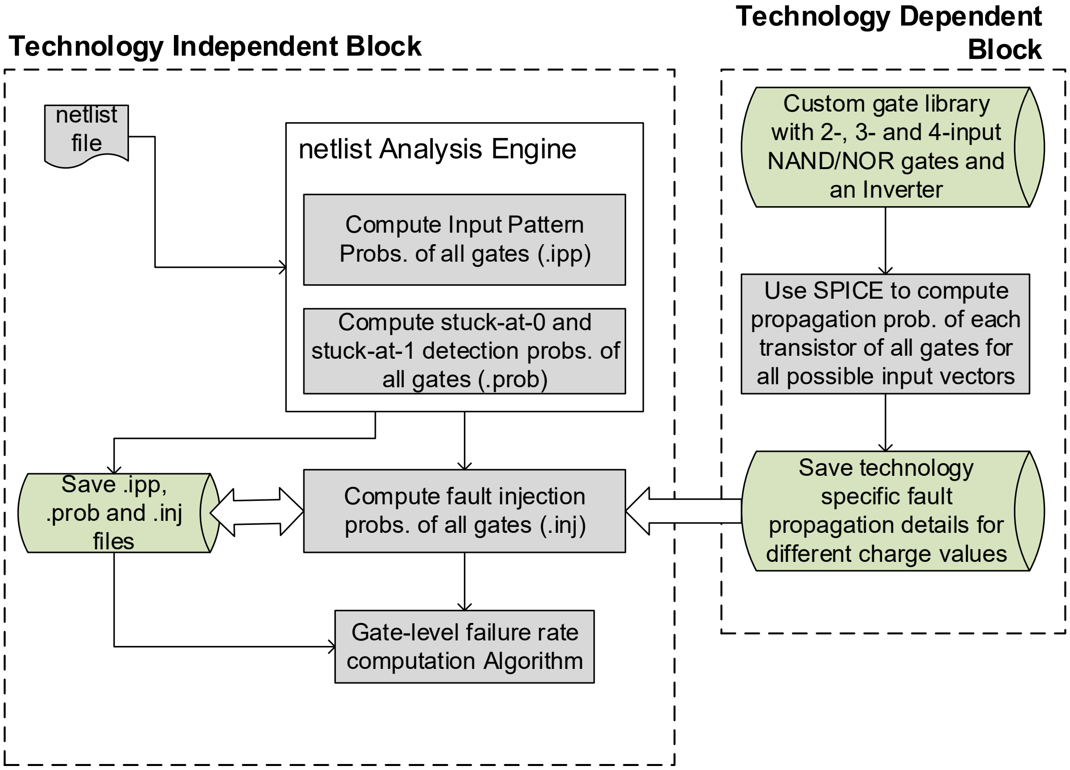


Figure 6.1: Reliability Evaluation Architecture.

e.g. 45nm, 32nm etc., against a specific charge value. This block computes the effect of an induced current of charge (Q) for every transistor of the gate in the library. The input patterns that result in a gate value flip when a transistor is hit are then saved in the propagation (`.prop`) file. In fact, we can compute and save the behavior of different technologies against different charge (Q) values, and this has to be done only once.

Now, the fault injection probability of a gate in a circuit can be computed for any process technology. It must be noted that the initial analysis of a circuit has to be done only once. The only limitation to this approach is that, if a circuit changes, the analysis for the technology independent part has to be repeated as well.

The following subsections contain the detailed elaboration of the reliability evaluation framework.

6.2 Probability of Fault Injection

The fault injection probabilities of a gate depend on the *Conditional Fault Excitation Probability* ($CFEP_{ij}$) and probability of hit/selection. A general relation to compute $CFEP_{ij}$ of j^{th} transistor of a gate i can be derived as follows. Let **S** be a set of patterns for which an error is excited to the output of a gate and PC_i be the controllability probability to produce a logic value opposite to the fault effect at the gate output. Then $CFEP_{ij}$ can be defined as:

$$CFEP_{ij} = \frac{\sum_{k=1}^{|\mathbf{S}|} Prob.\ \mathbf{S_k}}{PC_i} = \frac{P_{Excitation_{ij}}}{PC_i} \tag{6.1}$$

The $CFEP_{ij}$ of any MOS transistor depends on the process technology and the charge of the incident particle. Therefore, in order to get the exact $CFEP_{ij}$ probability for each MOS transistor, transistor level simulations are performed using SPICE.

Now, the sa0 fault injection probability of gate G_i is computed using the following equation:

$$G_i\ sa0\ inj.\ Prob\ = \sum_{j=1}^{n} \left(CFEP_{N_{ij}} \times \frac{NW_{ij}}{\sum\limits_{k=1}^{n} NW_{ik}} \right) \tag{6.2}$$

Where n is the total number of `nmos` transistors in gate G_i, NW_{ij} is the width of the drain of j^{th} `nmos` transistor and $CFEP_{N_{ij}}$ is the conditional fault excitation probability due to a fault hit at the j^{th} `nmos` transistor of gate i.

Similarly, the sa1 fault injection probability of gate G_i is computed as follows:

$$G_i\ sa1\ inj.\ Prob\ = \sum_{j=1}^{p} \left(CFEP_{P_{ij}} \times \frac{PW_{ij}}{\sum\limits_{k=1}^{p} PW_{ik}} \right) \tag{6.3}$$

Where p is the total number of `pmos` transistors in gate G_i, PW_{ij} is the width of the drain of j^{th} `pmos` transistor and $CFEP_{P_{ij}}$ is the conditional fault excitation probability due to a fault hit at the j^{th} `pmos` transistor of gate i.

With $P_{Excitation_{N1}} = 0.75$, $P_{Excitation_{N2}} = 0.25$, $P_{Excitation_{P1}} = 0.25$, $P_{Excitation_{P2}} = 0.25$, $PC0 = 0.25$ and $PC1 = 0.75$, the sa0 and sa1 injection probabilities of a 2-input NAND gate are computed by applying Eqn. 6.2 and Eqn. 6.3 as follows:

$$\begin{aligned} G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{2} CFEP_{N_{ij}} \times \frac{NW_{ij}}{\sum\limits_{k=1}^{2} NW_{ik}} \\ &= \frac{\frac{0.75}{0.75} \times 0.26 + \frac{0.25}{0.75} \times 0.26}{\sum\limits_{i=1}^{2} 0.26} \\ &= 0.667 \end{aligned}$$

$$\begin{aligned} G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{2} CFEP_{P_{ij}} \times \frac{PW_{ij}}{\sum_{k=1}^{2} PW_{ik}} \\ &= \frac{1 \times 0.52 + 1 \times 0.52}{\sum_{i=1}^{2} 0.52} \\ &= 1 \end{aligned}$$

Let's now compute the fault injection probabilities of NAND21 shown in Table 3.2. $P_{DET_{N1}}$ is reduced to 0.25 as there is only one pattern, $\{10\}$, for which "N1" is not protected; therefore, $CFEP_{N1} = \frac{0.25}{0.75} = \frac{1}{3}$. The sa0 fault injection probability will be:

$$\begin{aligned} G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{2} CFEP_{N_{ij}} \times \frac{NW_{ij}}{\sum_{k=1}^{2} NW_{ik}} \\ &= \frac{\frac{0.25}{0.75} \times 0.26 + \frac{0.25}{0.75} \times 0.26}{\sum_{i=1}^{2} 0.26} \\ &= 0.333 \end{aligned}$$

The sa0 fault injection probability of NAND21 is reduced from 0.833 to 0.333 due to protecting the fault that occurs at the drain of N1 transistor.

The area of $P1\_1$ and $P1\_2$ `pmos` transistors is $2.4 \times PW$. The sa1 fault injection probability remains the same as computed below:

$$\begin{aligned}
G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{3} CFEP_{P_{ij}} \times \frac{PW_{ij}}{\sum\limits_{k=1}^{3} PW_{ik}} \\
&= \frac{2 \times 1 \times 2.4 \times 0.52 + 1 \times 0.52}{(\sum\limits_{i=1}^{2} 2.4 \times 0.52) + 0.52} \\
&= 1
\end{aligned}$$

6.3 Fault Injection Mechanisms

Two fault injection mechanisms are applied in this work. The first method performs fault injection at the transistor level and measures the magnitude of voltage V_{out} at the output. The second method deals with injecting the fault at the gate level by injecting a stuck-at-0 or stuck-at-1 fault at the gate output.

6.3.1 Transistor Level

The current I of charge Q is injected at the drain of a transistor. The direction of injected current is from *drain-to-body* in the `nmos` transistor and from *body-to-drain* in the `pmos` transistor. The magnitude and pulse width of injected current is modeled using Eqn. 3.1. Algorithm 8 highlights the steps of failure rate/reliability computation at the transistor level. In this algorithm, a set of **m** transistors are selected for fault injection using Roulette Wheel (RW) algorithm [91,92]. The RW algorithm selects transistors that have higher area with high probability. For each random input vector, the outputs are saved before and after the fault injection and are then compared to check for correctness. The failure rate and reliability of circuit are then computed after *SIM* simulations are performed.

Algorithm 8 : Transistor-level Failure Rate Computation

```
Require: Transistor-level netlist
$SIM$ : Simulation Count
$F_m$ : Failure rate of the circuit with $m$ faults
$Rel_m$ : Reliability (%) of the circuit with $m$ faults
$K$ : Failure Count
$R$ : Output of circuit with no fault injection
$R_f$ : Output of circuit after fault injection
$RW$ : Roulette Wheel algorithm
$K \leftarrow 0$
for ($i = 1 \rightarrow SIM$) do
    Generate a random test vector V
    Apply V to the circuit
    Simulate the circuit and save the output in $R$
    RW(m)            ▷ Select m transistors using Roulette Wheel Algorithm
    Inject faults in selected m transistors
    Apply V to the circuit with faults injected
    Simulate and save the output in $R_f$
    if ($R \neq R_f$) then increment $K$
end for
$F_m = \left(\frac{K}{SIM}\right)$
$Rel_m(\%) = (1 - F_m) \times 100$
```

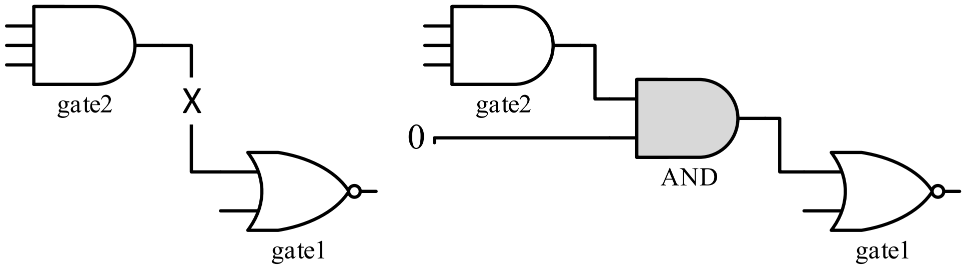
6.3.2 Gate Level

Faults injected at the gate level assume a stuck-at fault model. When a fault is injected at a gate output, it can be either a stuck-at-1 fault (i.e., connected to Vdd) or a stuck-at-0 fault (i.e., connected to ground). Algorithm 9 is used to

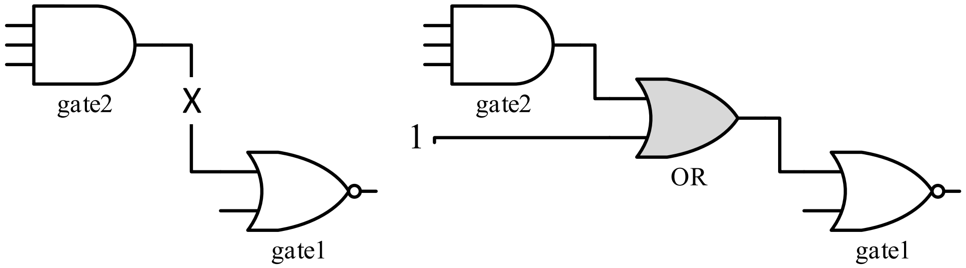
Algorithm 9 : Gate-level Failure Rate Computation

```
Require: Gate level netlist
$SIM$ : Simulation Count
$sa0_{G_i}$ : Stuck-at-0 injection probability of Gate $G_i$
$sa1_{G_i}$ : Stuck-at-1 injection probability of Gate $G_i$
$R$ : Output of circuit with no fault injection
$R_f$ : Output of circuit after fault injection
$RW$ : Roulette Wheel algorithm
$F_m$ : Failure rate of circuit with $m$ faults
$Rel_m$ : Reliability (%) of circuit with $m$ faults
$K$ : Failure Count
$rand1(\cdot)$ : Uniformly distributed random number $\sim$ (0,1)
$rand2(\cdot)$ : Uniformly distributed random number $\sim$ (0,1)

$K \leftarrow 0$
for ($i = 1 \rightarrow SIM$) do
    Generate a random test vector V
    Apply V to the circuit
    Simulate the circuit and save the output in $R$
    G←RW(m)                                          ▷ Select m gates
    for ($j = 1 \rightarrow G$) do                   ▷ Iterate through G gates
        if (($sa0_{G_j} + sa1_{G_j}$) == 0) then
            Don't inject any fault                   ▷ Gates protected
        else if $\left(rand1(\cdot) \leq \frac{sa0_{G_j}}{sa0_{G_j}+sa1_{G_j}}\right)$ then
            if ($rand2(\cdot) \leq sa0_{G_j}$) then
                Inject sa0 fault at the gate $G_j$ output
            end if
        else
            if ($rand2(\cdot) \leq sa1_{G_j}$) then
                Inject sa1 fault at the gate $G_j$ output
            end if
        end if
    end for
    Apply V to the circuit with faults injected
    Simulate and save the output in $R_f$
    if ($R \neq R_f$) then increment $K$
end for
$F_m = \frac{K}{SIM}$
$Rel_m(\%) = (1 - F_m) \times 100$
```

compute the circuit failure rate/reliability at the gate level. To inject **m** faults in a circuit, **m** gates are selected randomly using a Roulette Wheel (RW) algorithm. For each gate selected for fault injection, the following is performed. First, if both the sa0 and sa1 fault inject probabilities are 0, then no fault will be injected as the gate will be be fully protected. Otherwise, a selection is made between injecting a sa0 fault or a sa1 fault according to the ratio of their fault injection probabilities. The selected fault will be injected based on its fault injection probability. Fig. 6.2 illustrates the idea of injecting stuck-at-0 and stuck-at-1 fault at the gate-level.



(a) Stuck-at-0 fault.



(b) Stuck-at-1 fault.

Figure 6.2: Fault injection mechanism at gate-level.

6.3.3 Comparison b/w Transistor Level and Gate Level Simulations

To illustrate the accuracy of gate-level simulations, a comparison between transistor level and gate level simulations is shown in this section for few benchmark circuits. The transistor level simulations are performed using SPICE. Fig. 6.3-6.5 demonstrates the close match between the reliability obtained by SPICE and gate level simulations for the three compared benchmark circuits: apex2, apex3 and apex4. A circuit reliability is evaluated after performing 1000 iterations for each fault injection case.

Time is another factor that must be taken into account while evaluating a circuit for reliability. The time taken by SPICE simulations becomes exorbitantly high as the number of transistors is increased. The $apex4$ benchmark took around 4 days for SPICE simulations, while it took 30 minutes of gate level simulations, hence achieving a speedup of $\approx 167x$. It can be observed from Fig. 6.6 that as the number of transistors is increased, the speedup achieved by gate level simulations also increases significantly.

6.4 Reliability Evaluation of NAND Gates

In this section, we will discuss different protection scenarios applied at the transistor-level for 2-input NAND gate. Two cases have been discussed before, but we will discuss them here again for the sake of completeness. The taxonomy

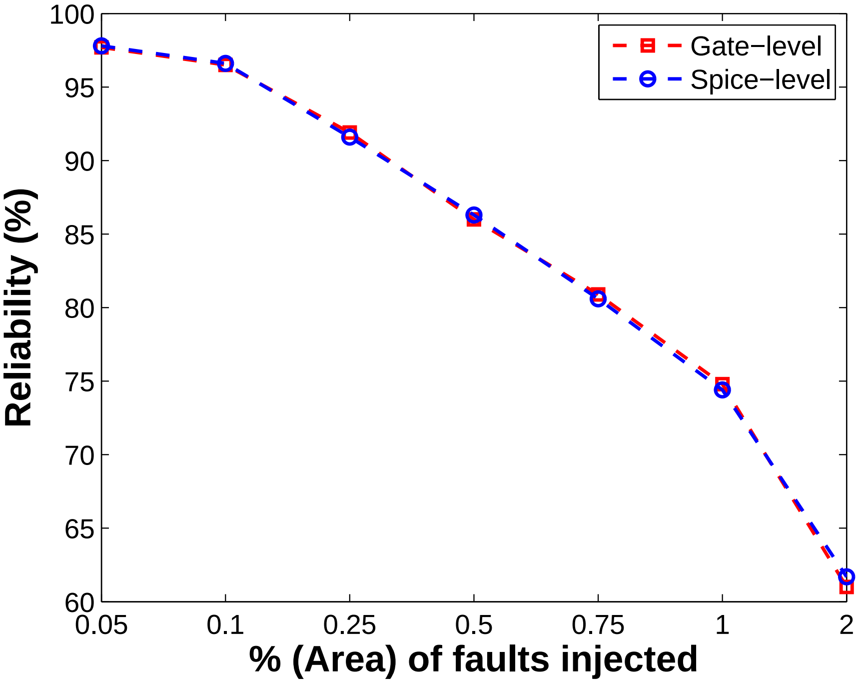


Figure 6.3: apex2 Reliability

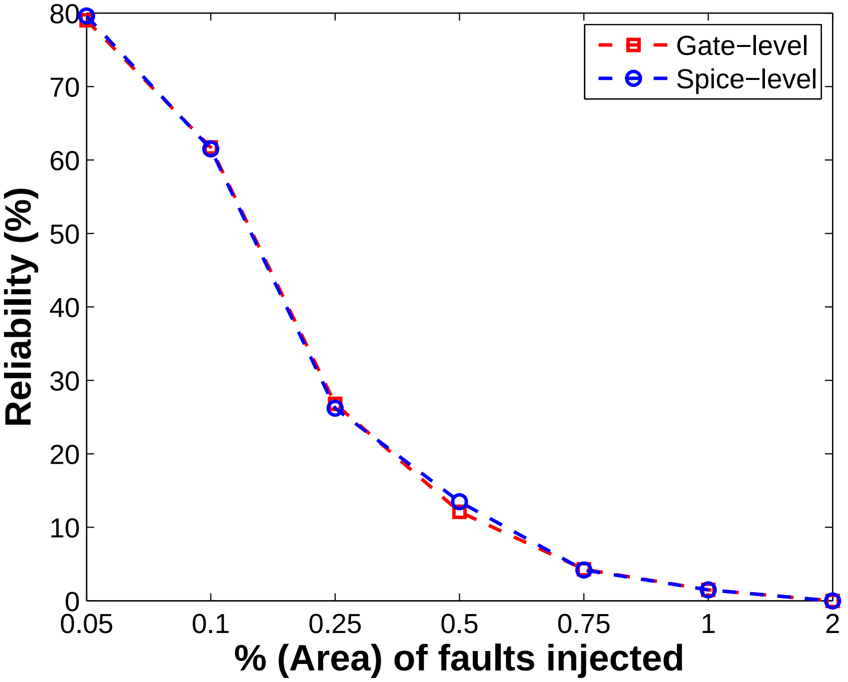


Figure 6.4: apex3 Reliability

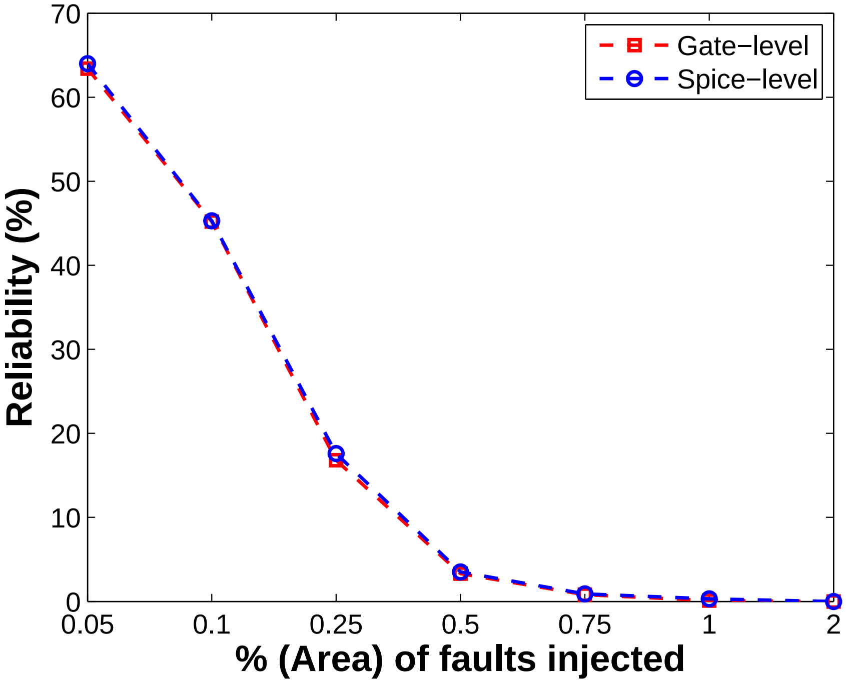


Figure 6.5: apex4 Reliability

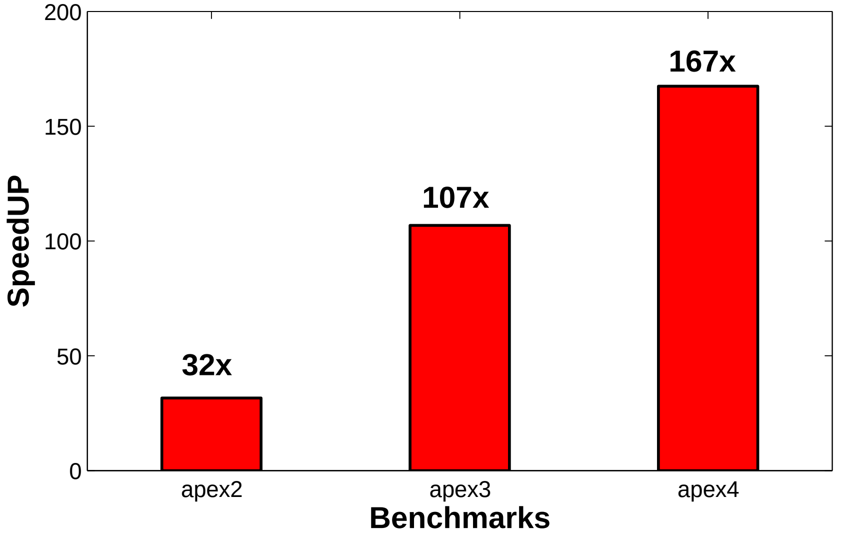


Figure 6.6: Time comparison.

used here is then used by the selection algorithm mentioned in Chapter 3 to mark gates for protection at the gate level. The basic process related parameters are mentioned in Table 6.1.

Table 6.1: Parameters considered in the study.

Technology (T)	$130m = 0.13\mu$
nMOS width (NW_{ij})	$2 \times T = 0.26\mu$
pMOS width (PW_{ij})	$4 \times T = 0.52\mu$
Charge (Q)	0.3pC

6.4.1 NAND2

The NAND2 is a 2-input NAND gate as shown in Fig. 6.7. The total area of a 2-input NAND gate is;

$$\begin{aligned} Area &= 2 \times NW + 2 \times PW \\ &= 2 \times 0.26 + 2 \times 0.52 \\ &= 1.560 \end{aligned}$$

Using SPICE and applying Eqn. 6.1, set **S** for `nmos` transistors N1 and N2 is observed to be;

$$\mathbf{S}_{N1} = \{00, 01, 10\} \implies CFEP_{N1} = \frac{0.75}{0.75} = 1$$

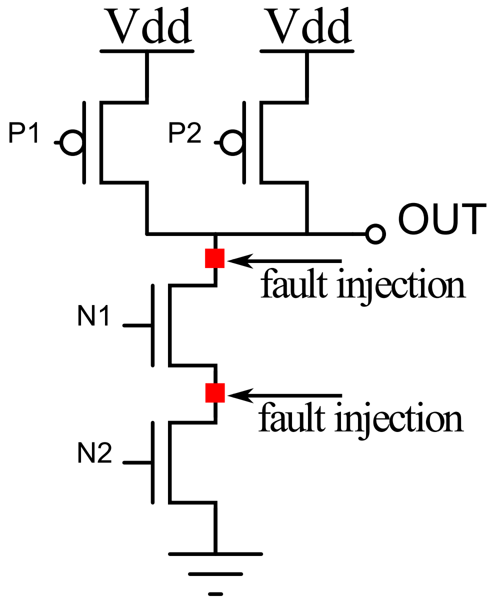


Figure 6.7: 2-input cmos NAND

$$\mathbf{S}_{N2} = \{10\} \implies CFEP_{N2} = \frac{0.25}{0.75} = 0.33$$

Again, it is considered that all input patterns are equally likely to occur. sa1 fault injected at any of the `pmos` transistor of a NAND gate will always be observed at the gate output. Then, set **S** for `pmos` transistors P1 and P2 will be;

$$\begin{aligned} \mathbf{S}_{P1} &= \{11\} \implies CFEP_{P1} = \frac{0.25}{0.25} = 1 \\ \mathbf{S}_{P2} &= \{11\} \implies CFEP_{P2} = \frac{0.25}{0.25} = 1 \end{aligned}$$

Now, applying Eqns. 6.2 and 6.3 to compute sa0 and sa1 injection probabilities of a 2-input NAND gate;

$$
\begin{aligned}
G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{2} CFEP_{N_{ij}} \times \frac{NW_{ij}}{\sum\limits_{k=1}^{2} NW_{ik}} \\
&= \frac{1 \times 0.26 + 0.33 \times 0.26}{\sum\limits_{k=1}^{2} 0.26} \\
&= 0.667
\end{aligned}
$$

$$
\begin{aligned}
G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{2} CFEP_{P_{ij}} \times \frac{PW_{ij}}{\sum\limits_{k=1}^{2} PW_{ik}} \\
&= \frac{1 \times 0.52 + 1 \times 0.52}{\sum\limits_{k=1}^{2} 0.52} \\
&= 1
\end{aligned}
$$

So, at the gate level simulations, the sa0 and sa1 faults must be injected with probabilities 0.667 and 1, respectively. $P_{DET_{ij}}$ for all transistors will be;

$$
\begin{aligned}
P_{DET_{N1}} &= \frac{0.75}{1} = 0.75 \\
P_{DET_{N2}} &= \frac{0.25}{1} = 0.25 \\
P_{DET_{P1}} &= \frac{0.25}{1} = 0.25
\end{aligned}
$$

$$P_{DET_{P2}} = \frac{0.25}{1} = 0.25$$

The theoretical gate failure probability of 2-input NAND gate can be computed using Eqn. 3.7 as follows:

$$\begin{aligned}
Gate\ Failure\ Prob. &= \sum_{j=1}^{2} P_{DET_{Nij}} \times \frac{NW_{ij}}{Area} \\
&\quad + \sum_{j=1}^{2} P_{DET_{Pij}} \times \frac{PW_{ij}}{Area} \\
&= \overbrace{\left(\frac{0.75 \times 0.26 + 0.25 \times 0.26}{1.56}\right)}^{N1,N2} \\
&\quad + \overbrace{\left(2 \times 0.25 \times \frac{0.52}{1.56}\right)}^{P1,P2} \\
&= 0.167 + 0.167 \\
&= 0.333
\end{aligned}$$

Finally, gate failure probability obtained by SPICE and gate level simulations is shown below. The results are obtained by running the reliability evaluation algorithm. The close match between theoretical and simulated failure probability of the 2-input NAND gate verifies the correctness of proposed fault injection method.

$$
\begin{aligned}
Probability\ of\ failure\ (Spice) &= 0.384 \\
Probability\ of\ failure\ (Gate) &= 0.37
\end{aligned}
$$

6.4.2 NAND21

In NAND21, the protection of sa0 faults against the transient fault at the drain of `nmos` transistor N1 is provided by duplicating the `pmos` transistor P1 and scaling the widths of duplicated P1_1 and P1_2 transistors by the factor λ to suppress the transient, as shown in Fig. 6.8. To protect sa0 faults at the drain of N1, the value of λ is increased incrementally until the output voltage $V > VDD/2$ and is derived empirically using SPICE.

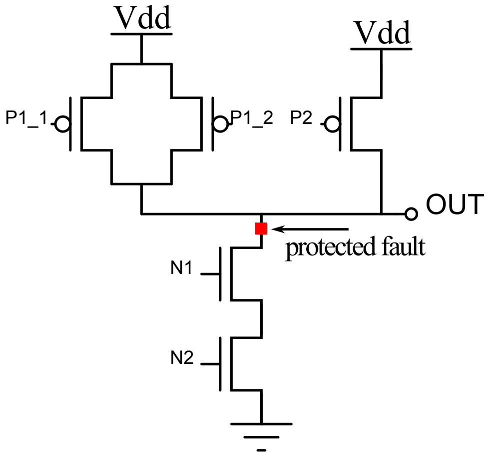


Figure 6.8: CMOS configuration of NAND21 gate

The motivation to duplicate a transistor is to make a circuit more vulnerable to single event multiple upsets (SEMU). In SEMU, the effect of transient fault is no more constrained to a node where the incident particle strikes. This could result in the possibility of deposited charge being simultaneously shared by multiple circuit nodes in the circuit. In fact, this situation is quite beneficial to the proposed scheme. For example, if a fault hits the drain of N1 transistor of a 2-input NAND gate and also makes any of the `pmos` transistor conducting due to its high current value, then, N1 and one of the `pmos` transistor will start conducting. In this case, the fault will never propagate to the output, as the two conducting transistors will just cancel the effect of each other. Let's now evaluate the reliability and compute the fault injection probability of NAND21 gate.

`pmos` Scaling factor (λ) $= 2.4$

Area $= 2 \times NW + 2 \times \lambda \times PW + PW = 3.536$

Then,

$$\begin{aligned}
\mathbf{S}_{N1} &= \{10\} \implies CFEP_{N1} = \frac{0.25}{0.75} = 0.33 \\
\mathbf{S}_{N2} &= \{10\} \implies CFEP_{N2} = \frac{0.25}{0.75} = 0.33 \\
\mathbf{S}_{P1\_1,P1\_2} &= \{11\} \implies CFEP_{P1\_1,P1\_2} = \frac{0.25}{0.25} = 1 \\
\mathbf{S}_{P2} &= \{11\} \implies CFEP_{P2} = \frac{0.25}{0.25} = 1
\end{aligned}$$

Fault injection probabilities will be;

$$\begin{aligned}
G_i \; sa0 \; inj. \; Prob. &= \sum_{j=1}^{2} CFEP_{N_{ij}} \times \frac{NW_{ij}}{\sum\limits_{k=1}^{2} NW_{ik}} \\
&= \frac{0.33 \times 0.26 + 0.33 \times 0.26}{\sum\limits_{k=1}^{2} 0.26} \\
&= 0.333
\end{aligned}$$

The area of P1_1 and P1_2 `pmos` transistors is $\lambda \times PW = 2.4 \times PW$. The sa1 fault injection probability, then, can be computed as follows;

$$\begin{aligned}
G_i \; sa1 \; inj. \; Prob. &= \sum_{j=1}^{3} CFEP_{P_{ij}} \times \frac{PW_{ij}}{\sum\limits_{k=1}^{3} PW_{ik}} \\
&= \frac{2 \times 2.4 \times 0.52 + 1 \times 0.52}{(\sum\limits_{i=1}^{2} 2.4 \times 0.52) + 0.52} \\
&= 1
\end{aligned}$$

The failure probability of NAND21 gate will be;

$$Gate \; Failure \; Prob. = \sum_{j=1}^{2} P_{DET_{Nij}} \times \frac{NW_{ij}}{Area}$$

$$
\begin{aligned}
& + \sum_{j=1}^{3} P_{DET_{Pij}} \times \frac{PW_{ij}}{Area} \\
= \; & \overbrace{\left(\frac{0.25 \times 0.26 + 0.25 \times 0.26}{3.536} \right)}^{N1,N2} \\
& + \overbrace{\left(2 \times 0.25 \times \frac{2.4 \times 0.52}{3.536} \right)}^{P1\_1,P1\_2} \\
& + \overbrace{\left(0.25 \times \frac{0.18}{3.536} \right)}^{P2} \\
= \; & 0.250
\end{aligned}
$$

Finally, gate failure probability obtained by SPICE and gate-level simulations is;

$$
\begin{aligned}
Probability\ of\ failure\ (Spice) &= 0.257 \\
Probability\ of\ failure\ (Gate) &= 0.248
\end{aligned}
$$

6.4.3 NAND22

In NAND22, the protection from sa0 faults against the transient hit at the drain of `nmos` transistors N1 or N2 is provided by duplicating the `pmos` transistors P1 and P2. The widths of duplicated P1_1, P1_2, P2_1 and P2_2 are also scaled by the factor λ as shown in Fig. 6.9.

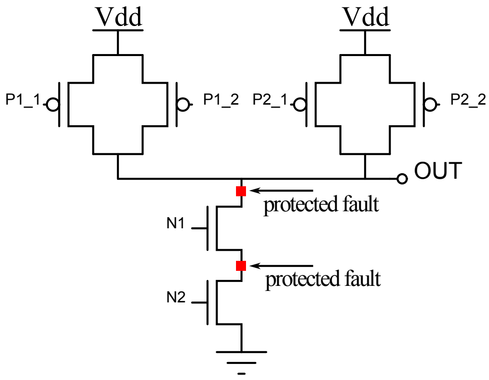


Figure 6.9: CMOS configuration of NAND22 gate

`pmos` Scaling factor $(\lambda) = 2.4$

Area $= 2 \times NW + 4 \times \lambda \times PW = 5.512$

Then,

$$\begin{aligned}
\mathbf{S}_{N1} &= \phi \implies CFEP_{N1} = 0 \\
\mathbf{S}_{N2} &= \phi \implies CFEP_{N2} = 0 \\
\mathbf{S}_{P1\_1,P1\_2} &= \{11\} \implies CFEP_{P1\_1,P1\_2} = 1 \\
\mathbf{S}_{P2\_1,P2\_2} &= \{11\} \implies CFEP_{P2\_1,P2\_2} = 1
\end{aligned}$$

Fault injection probabilities will be;

$$G_i\ sa0\ inj.\ Prob. = \sum_{j=1}^{2} CFEP_{N_{ij}} \times \frac{NW_{ij}}{\sum\limits_{k=1}^{2} NW_{ik}}$$

$$= \quad 0$$

The area of each of P1_1, P1_2, P2_1 and P2_2 `pmos` transistors is $\lambda \times PW = 2.4 \times PW$. The sa1 fault injection probability, then, can be computed as follows;

$$\begin{aligned}
G_i \ sa1 \ inj. \ Prob. \quad &= \quad \sum_{j=1}^{4} CFEP_{P_{ij}} \times \frac{\lambda PW_{ij}}{\sum\limits_{k=1}^{4} \lambda PW_{ik}} \\
&= \quad \frac{2 \times \lambda \times 0.52 + 2 \times \lambda \times 0.52}{\sum\limits_{k=1}^{4} \lambda \times 0.52} \\
&= \quad 1
\end{aligned}$$

The failure probability of NAND22 gate will be;

$$\begin{aligned}
Gate \ Failure \ Prob. \quad &= \quad \sum_{j=1}^{2} P_{DET_{Nij}} \times \frac{NW_{ij}}{Area} \\
&\quad + \sum_{j=1}^{4} P_{DET_{Pij}} \times \frac{PW_{ij}}{Area} \\
&= \quad 0 + \overbrace{\left(4 \times 0.25 \times \lambda \times \frac{0.52}{5.512}\right)}^{P1\_1, P1\_2, P2\_1, P2\_2} \\
&= \quad 0.226
\end{aligned}$$

Finally, gate failure probability obtained by SPICE and gate level simulations is

$$\begin{aligned} Probability\ of\ failure\ (Spice) &= 0.226 \\ Probability\ of\ failure\ (Gate) &= 0.231 \end{aligned}$$

6.4.4 NAND23

NAND23 provides protection from sa1 faults only by duplicating and scaling all the `nmos` transistors. Fig. 6.10 shows the arrangement of transistors in NAND23.

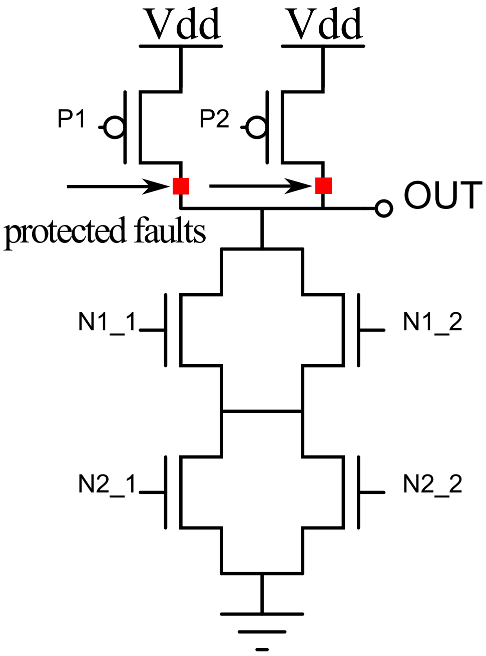


Figure 6.10: CMOS configuration of NAND23 gate

`nmos` Scaling factor (λ) = 3.1

Area $= 4 \times \lambda \times NW + 2 \times PW = 4.264$

Then,

$$\begin{aligned}
\mathbf{S}_{N1\_1,N1\_2} &= \{00, 01, 10\} \implies CFEP_{N1\_1,N1\_2} = \frac{0.75}{0.75} = 1 \\
\mathbf{S}_{N2\_1,N2\_2} &= \{10\} \implies CFEP_{N2\_1,N2\_2} = \frac{0.25}{0.75} = 0.333 \\
\mathbf{S}_{P1} &= \phi \implies CFEP_{P1} = 0 \\
\mathbf{S}_{P2} &= \phi \implies CFEP_{P2} = 0
\end{aligned}$$

Fault injection probabilities will be;

$$\begin{aligned}
G_i\ sa0\ inj.\ Prob &= \sum_{j=1}^{2} CFEP_{N_{ij}} \times \frac{NW_{ij}}{\sum_{k=1}^{2} NW_{ik}} \\
&= \frac{2 \times 1 \times \lambda \times 0.26 + 2 \times 0.333 \times \lambda \times 0.26}{\sum_{k=1}^{4} \lambda \times 0.26} \\
&= 0.833
\end{aligned}$$

$$\begin{aligned}
G_i\ sa1\ inj.\ Prob &= \sum_{j=1}^{2} CFEP_{P_{ij}} \times \frac{PW_{ij}}{\sum_{k=1}^{2} PW_{ik}} \\
&= 0
\end{aligned}$$

The failure probability of NAND23 gate will be;

$$\begin{aligned}
Gate\ Failure\ Prob. &= \sum_{j=1}^{4} P_{DET_{Nij}} \times \frac{NW_{ij}}{Area} \\
&\quad + \sum_{j=1}^{2} P_{DET_{Pij}} \times \frac{PW_{ij}}{Area} \\
&= \frac{\overbrace{2 \times 0.75 \times \lambda \times 0.26}^{N1\_1,N1\_2} + \overbrace{2 \times 0.25 \times \lambda \times 0.26}^{N2\_1,N2\_2}}{4.264} + 0 \\
&= 0.473
\end{aligned}$$

Finally, gate failure probability of NAND23 obtained by SPICE and gate level simulations is found to be;

$$\begin{aligned}
Probability\ of\ failure\ (Spice) &= 0.474 \\
Probability\ of\ failure\ (Gate) &= 0.480
\end{aligned}$$

6.4.5 NAND24

NAND24 provides protection from faults that can occur at any of the `pmos` transistors or at the first `nmos` transistor i.e N1. Fig. 6.11 shows the arrangement of transistors in NAND24.

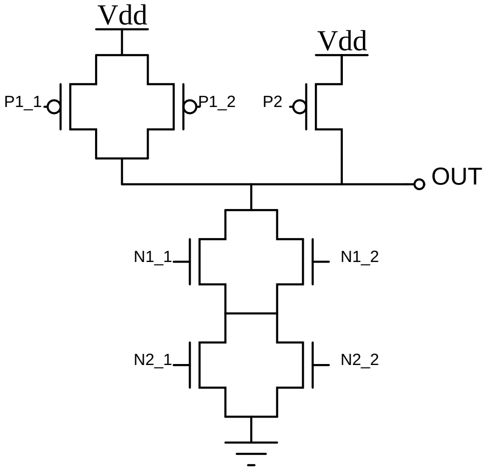


Figure 6.11: CMOS configuration of NAND24 gate

`nmos` Scaling factor (λ_1) = 3.1

`pmos` Scaling factor (λ_2) = 2.4

Area $= 4 \times \lambda_1 \times NW + 2 \times \lambda_2 \times PW + PW = 6.240$

Then,

$$\begin{aligned}
\mathbf{S}_{N1\_1,N1\_2} &= \{10\} \implies CFEP_{N1\_1,N1\_2} = \frac{0.25}{0.75} = 0.33 \\
\mathbf{S}_{N2\_1,N2\_2} &= \{10\} \implies CFEP_{N2\_1,N2\_2} = \frac{0.25}{0.75} = 0.33 \\
\mathbf{S}_{P1\_1,P1\_2} &= \phi \implies CFEP_{P1\_1,P1\_2} = 0 \\
\mathbf{S}_{P2} &= \phi \implies CFEP_{P2} = 0
\end{aligned}$$

Fault injection probabilities will be:

$$\begin{aligned} G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{4} CFEP_{N_{ij}} \times \frac{\lambda_1 NW_{ij}}{\sum_{k=1}^{4} \lambda_1 NW_{ik}} \\ &= \frac{2 \times \frac{1}{3} \times \lambda_1 \times 0.26 + 2 \times \frac{1}{3} \times \lambda_1 \times 0.26}{\sum_{k=1}^{4} \lambda_1 \times 0.26} \\ &= 0.333 \end{aligned}$$

sa1 injection probability will be zero as they are protected.

$$\begin{aligned} G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{3} CFEP_{P_{ij}} \times \frac{\lambda_2 PW_{ij}}{\sum_{k=1}^{3} \lambda_2 PW_{ik}} \\ &= 0 \end{aligned}$$

The failure probability of NAND24 gate will be:

$$\begin{aligned} Gate\ Failure\ Prob. &= \sum_{j=1}^{4} P_{DET_{Nij}} \times \frac{\lambda_1 NW_{ij}}{Area} \\ &\quad + \sum_{j=1}^{2} P_{DET_{Pij}} \times \frac{\lambda_2 PW_{ij}}{Area} \\ &= \frac{\overbrace{2 \times 0.25 \times \lambda_1 \times 0.26}^{N1\_1, N1\_2} + \overbrace{2 \times 0.25 \times \lambda_1 \times 0.26}^{N2\_1, N2\_2}}{6.240} \end{aligned}$$

$$= 0.129$$

Finally, gate failure probability of NAND24 obtained by SPICE and gate level simulations is found to be;

$$\begin{aligned} Probability\ of\ failure\ (Spice) &= 0.130 \\ Probability\ of\ failure\ (Gate) &= 0.135 \end{aligned}$$

6.4.6 NAND25

The redundancy and scaling of transistors in NAND25 combines the NAND22 and NAND23 schemes to protect from both sa0 and sa1 faults. Fig. 6.12 shows the CMOS representation of NAND25.

`nmos` Scaling factor (λ_1) = 3.1

`pmos` Scaling factor (λ_2) = 2.4

Area $= 4 \times \lambda_1 \times NW + 4 \times \lambda_2 \times PW = 8.216$

Fault injection probabilities will be zero for both sa0 and sa1 faults as they will be suppressed by transistors arrangement in NAND25.

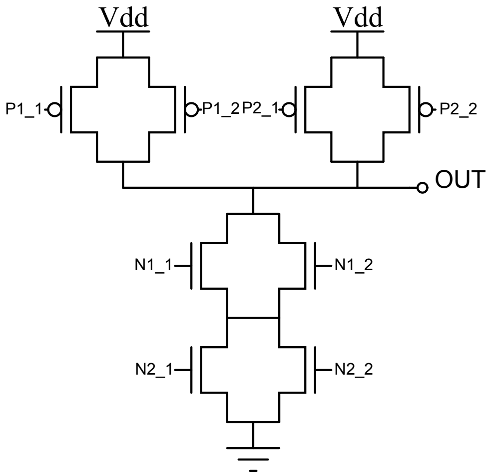


Figure 6.12: CMOS configuration of NAND25 gate

$$\begin{aligned} G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{4} CFEP_{N_{ij}} \times \frac{\lambda_1 NW_{ij}}{\sum\limits_{k=1}^{4} \lambda_1 NW_{ik}} \\ &= 0 \end{aligned}$$

$$\begin{aligned} G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{4} CFEP_{P_{ij}} \times \frac{\lambda_2 PW_{ij}}{\sum\limits_{k=1}^{4} \lambda_2 PW_{ik}} \\ &= 0 \end{aligned}$$

The failure probability of NAND25 gate will be;

$$\begin{aligned} Gate\ Failure\ Prob. &= \sum_{j=1}^{4} P_{DET_{Nij}} \times \frac{\lambda_1 NW_{ij}}{Area} \\ &\quad + \sum_{j=1}^{4} P_{DET_{Pij}} \times \frac{\lambda_2 PW_{ij}}{Area} \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

Finally, gate failure probability of NAND25 obtained by SPICE and gate level simulations is found to be;

$$\begin{aligned} Probability\ of\ failure\ (Spice) &= 0 \\ Probability\ of\ failure\ (Gate) &= 0 \end{aligned}$$

6.5 Reliability Evaluation of NOR Gates

In this section, we will discuss different protection scenarios applied at the transistor level for 2-input NOR gate.

6.5.1 NOR2

The NOR2 is a 2-input NOR gate as shown in Fig. 6.13. The total area of a 2-input NOR gate is;

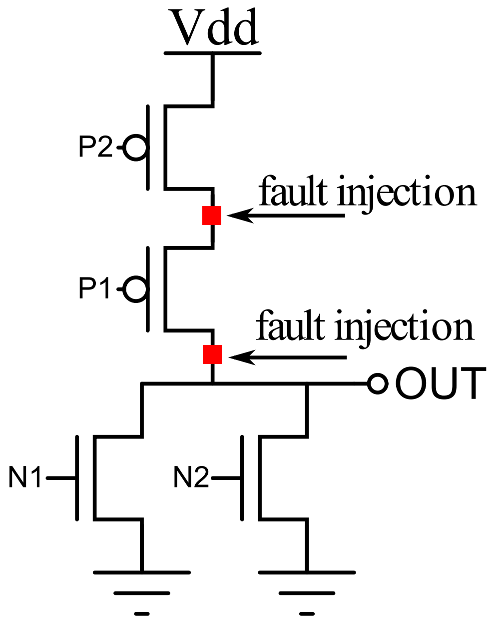


Figure 6.13: 2-input cmos NOR

$$\begin{aligned} Area &= 2 \times NW + 2 \times PW \\ &= 2 \times 0.26 + 2 \times 0.52 \\ &= 1.560 \end{aligned}$$

Using SPICE and applying Eqn. 6.1, set **S** for `pmos` transistors P1 and P2 is observed to be:

$$\begin{aligned}
\mathbf{S}_{P1} &= \{01, 10, 11\} \implies CFEP_{P1} = \frac{0.75}{0.75} = 1 \\
\mathbf{S}_{P2} &= \{01\} \implies CFEP_{P2} = \frac{0.25}{0.75} = 0.333
\end{aligned}$$

sa0 fault injected at any of the `nmos` transistor of a NOR gate will always be observed at the gate output. Then, set **S** for `nmos` transistors N1 and N2 will be;

$$\begin{aligned}
\mathbf{S}_{N1} &= \{00\} \implies CFEP_{N1} = \frac{0.25}{0.25} = 1 \\
\mathbf{S}_{N2} &= \{00\} \implies CFEP_{N2} = \frac{0.25}{0.25} = 1
\end{aligned}$$

Now, applying Eqn. 6.2 and Eqn. 6.3 to compute sa0 and sa1 injection probabilities of a 2-input NOR gate;

$$\begin{aligned}
G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{2} CFEP_{N_{ij}} \times \frac{\lambda NW_{ij}}{\sum_{k=1}^{2} NW_{ik}} \\
&= \frac{1 \times 0.26 + 1 \times 0.26}{\sum_{k=1}^{2} 0.26} \\
&= 1
\end{aligned}$$

$$\begin{aligned}
G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{2} CFEP_{P_{ij}} \times \frac{PW_{ij}}{\sum_{k=1}^{2} \lambda PW_{ij}} \\
&= \frac{1 \times 0.52 + 1 \times 0.52}{\sum_{k=1}^{2} 0.52} \\
&= 0.667
\end{aligned}$$

So, at the gate level simulations, the sa0 fault must always be selected while sa1 fault, if selected, must be injected with probability 0.667.

$P_{DET_{ij}}$ for all transistors will be:

$$\begin{aligned}
P_{DET_{N1}} &= \frac{0.25}{1} = 0.25 \\
P_{DET_{N2}} &= \frac{0.25}{1} = 0.25 \\
P_{DET_{P1}} &= \frac{0.75}{1} = 0.75 \\
P_{DET_{P2}} &= \frac{0.25}{1} = 0.25
\end{aligned}$$

The theoretical failure probability can be computed using Eqn. Circuit Probability of Failure as follows:

$$\begin{aligned}
Gate\ Failure\ Prob. &= \sum_{j=1}^{2} P_{DET_{Nij}} \times \frac{NW_{ij}}{Area} \\
&\quad + \sum_{i=1}^{2} P_{DET_{Pij}} \times \frac{PW_{ij}}{Area} \\
&= \overbrace{\left(2 \times 0.25 \times \frac{0.26}{1.560}\right)}^{N1,N2} + \overbrace{\left(\frac{0.75 \times 0.52 + 0.25 \times 0.52}{1.560}\right)}^{P1,P2} \\
&= 0.083 + 0.333 \\
&= 0.416
\end{aligned}$$

Finally, gate failure probability of NOR2 obtained by SPICE and gate level simulations is shown below and shows that the theoretical failure probability and empirical gate failure probabilities are very close to each other.

$$\begin{aligned}
Probability\ of\ failure\ (Spice) &= 0.413 \\
Probability\ of\ failure\ (Gate) &= 0.418
\end{aligned}$$

6.5.2 NOR21

In NOR21, the protection from sa1 faults against the transient fault at the drain of `pmos` transistor P1 is provided by duplicating the `nmos` transistor N1 and scaling the widths of duplicated N1_1 and N1_2 by the factor λ, as shown in Fig. 6.14. For protection against sa1 faults, the value of λ is increased incrementally until the output voltage (V) $< VDD/2$ and is derived empirically using SPICE.

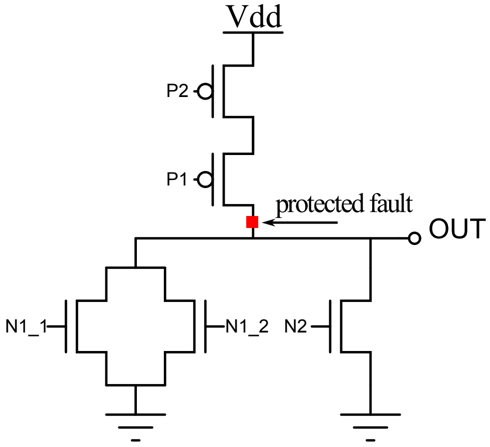


Figure 6.14: CMOS configuration of NOR21 gate

`nmos` Scaling factor (λ) = 2

Area $= 2 \times \lambda \times NW + NW + 2 \times PW = 2.340$

Then,

$$\begin{aligned}
\mathbf{S}_{N1\_1,N1\_2} &= \{00\} \implies CFEP_{N1\_1,N1\_2} = 1 \\
\mathbf{S}_{N2} &= \{00\} \implies CFEP_{N2} = 1 \\
\mathbf{S}_{P1} &= \{01\} \implies CFEP_{P1} = \frac{0.25}{0.75} = 0.33 \\
\mathbf{S}_{P2} &= \{01\} \implies CFEP_{P2} = \frac{0.25}{0.75} = 0.33
\end{aligned}$$

The area of N1_1 and N1_2 `pmos` transistors is $\lambda \times NW = 2 \times NW$. The sa0 and sa1 fault injection probability is computed as follows;

$$\begin{aligned}
G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{3} CFEP_{N_{ij}} \times \frac{\lambda NW_{ij}}{\sum\limits_{i=1}^{3} \lambda NW_{ik}} \\
&= \frac{2 \times \lambda \times 0.26 + 1 \times 0.26}{(\sum\limits_{k=1}^{2} \lambda \times 0.26) + 0.26} \\
&= 1
\end{aligned}$$

$$\begin{aligned}
G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{2} CFEP_{P_{ij}} \times \frac{\lambda PW_{ij}}{\sum\limits_{k=1}^{2} \lambda PW_{ik}} \\
&= \frac{0.33 \times 0.52 + 0.33 \times 0.52}{\sum\limits_{k=1}^{2} 0.52} \\
&= 0.333
\end{aligned}$$

The failure probability of NOR21 gate will be;

$$\begin{aligned}
Gate\ Failure\ Prob. &= \sum_{j=1}^{3} P_{DET_{Nij}} \times \frac{\lambda NW_{ij}}{Area} \\
&\quad + \sum_{i=1}^{2} P_{DET_{Pij}} \times \frac{PW_{ij}}{Area}
\end{aligned}$$

$$
\begin{aligned}
&= \overbrace{\left(2 \times 0.25 \times \lambda \times \frac{0.26}{2.340}\right)}^{N1\_1, N1\_2} + \overbrace{\left(0.25 \times \frac{0.26}{2.340}\right)}^{N2} \\
&\quad + \overbrace{\left(\frac{0.25 \times 0.52 + 0.25 \times 0.52}{2.340}\right)}^{P1, P2} \\
&= 0.250
\end{aligned}
$$

Finally, gate failure probability obtained by SPICE and gate level simulations is

$$
\begin{aligned}
Probability\ of\ failure\ (Spice) &= 0.252 \\
Probability\ of\ failure\ (Gate) &= 0.241
\end{aligned}
$$

6.5.3 NOR22

In NOR22, the protection from sa1 faults against the transient hit at the drain of `pmos` transistors P1 or P2 is provided by duplicating the `nmos` transistor N1 and N2. The widths of duplicated N1_1, N1_2, N2_1 and N2_2 are also scaled by the factor λ as shown in Fig. 6.15.

`pmos` Scaling factor (λ) = 2

Area = $4 \times \lambda \times NW + 2 \times PW = 3.120$

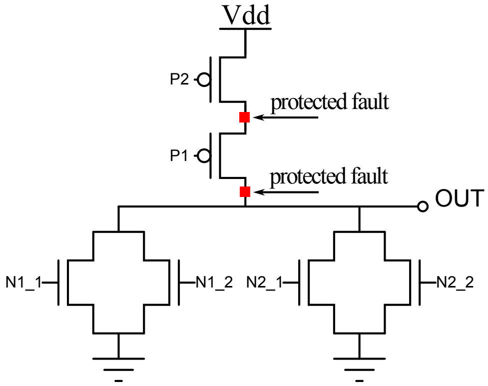


Figure 6.15: CMOS configuration of NOR22 gate

Then,

$$\begin{aligned}
\mathbf{S}_{N1\_1,N1\_2} &= \{00\} \implies CFEP_{N1\_1,N1\_2} = 1 \\
\mathbf{S}_{N2\_1,N2\_2} &= \{00\} \implies CFEP_{N2\_1,N2\_2} = 1 \\
\mathbf{S}_{P1} &= \phi \implies CFEP_{P1} = 0 \\
\mathbf{S}_{P2} &= \phi \implies CFEP_{P2} = 0
\end{aligned}$$

The area of each of N1_1, N1_2, N2_1 and N2_2 `nmos` transistors is $\lambda \times NW = 2 \times NW$. The sa0 fault injection probability, then, can be computed as follows;

$$\begin{aligned}
G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{4} CFEP_{N_{ij}} \times \frac{\lambda NW_{ij}}{\sum_{k=1}^{4} \lambda NW_{ik}} \\
&= \frac{2 \times \lambda \times 0.26 + 2 \times \lambda \times 0.26}{\sum_{k=1}^{4} \lambda \times 0.26}
\end{aligned}$$

$$= \quad 1$$

$$\begin{aligned}
G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{2} CFEP_{P_{ij}} \times \frac{PW_{ij}}{\sum\limits_{k=1}^{2} PW_{ik}} \\
&= \frac{0 \times 0.52 + 0 \times 0.52}{\sum\limits_{k=1}^{2} 0.52} \\
&= 0
\end{aligned}$$

The failure probability of NOR22 gate will be;

$$\begin{aligned}
Gate\ Failure\ Prob. &= \sum_{j=1}^{4} P_{DET_{Nij}} \times \frac{\lambda NW_{ij}}{Area} \\
&\quad + \sum_{i=1}^{2} P_{DET_{Pij}} \times \frac{PW_{ij}}{Area} \\
&= \overbrace{\left(2 \times 0.25 \times \lambda \times \frac{0.26}{3.120}\right)}^{N1\_1,N1\_2} + \overbrace{\left(2 \times \lambda \times 0.25 \times \frac{0.26}{3.120}\right)}^{N2\_1,N2\_2} + 0 \\
&= 0.167
\end{aligned}$$

Finally, gate failure probability obtained by SPICE and gate level simulations is

$$Probability\ of\ failure\ (Spice) = 0.168$$
$$Probability\ of\ failure\ (Gate) = 0.169$$

6.5.4 NOR23

NOR23 provides protection from sa0 faults only by duplicating and scaling the `pmos` transistors. Fig. 6.16 shows the arrangement of transistors in NOR23 gate.

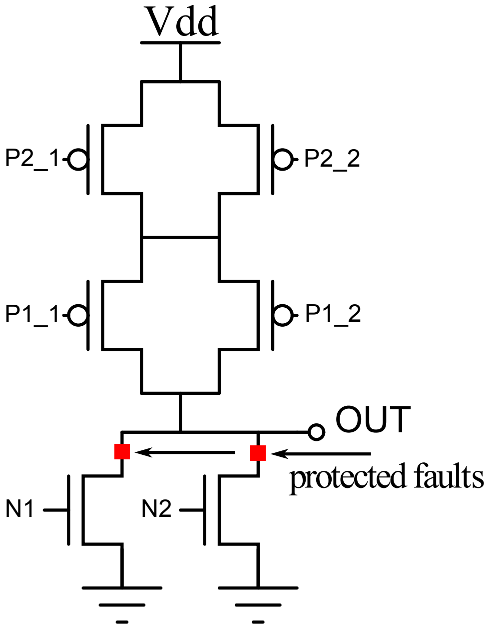


Figure 6.16: CMOS configuration of NOR23 gate

`nmos` Scaling factor (λ) = 4.4

Area = $2 \times NW + 4 \times \lambda \times PW = 9.672$

Then,

$$
\begin{aligned}
\mathbf{S}_{N1} &= \phi \implies CFEP_{N1} = 0 \\
\mathbf{S}_{N2} &= \phi \implies CFEP_{N2} = 0 \\
\mathbf{S}_{P1\_1,P1\_2} &= \{01, 10, 11\} \implies CFEP_{P1\_1,P1\_2} = \frac{0.75}{0.75} = 1 \\
\mathbf{S}_{P2\_1,P2\_2} &= \{01\} \implies CFEP_{P2\_1,P2\_2} = \frac{0.25}{0.75} = 0.333
\end{aligned}
$$

Fault injection probabilities will be;

$$
\begin{aligned}
G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{2} CFEP_{N_{ij}} \times \frac{NW_{ij}}{\sum\limits_{k=1}^{2} NW_{ik}} \\
&= \frac{0 \times 0.26 + 0 \times 0.26}{\sum\limits_{k=1}^{2} 0.26} \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{4} CFEP_{P_{ij}} \times \frac{\lambda PW_{ij}}{\sum\limits_{k=1}^{4} \lambda PW_{ik}} \\
&= \frac{2 \times 1 \times \lambda \times 0.52 + 2 \times 1 \times \lambda \times 0.52}{\sum\limits_{k=1}^{4} \lambda \times 0.52} \\
&= 0.833
\end{aligned}
$$

The failure probability of NOR23 gate will be:

$$\begin{aligned}
Gate\ Failure\ Prob. &= \sum_{j=1}^{2} P_{DET_{Nij}} \times \frac{NW_{ij}}{Area} \\
&\quad + \sum_{i=1}^{4} P_{DET_{Pij}} \times \frac{\lambda PW_{ij}}{Area} \\
&= 0 + \frac{\overbrace{2 \times 0.75 \times \lambda \times 0.52}^{P1\_1, P1\_2} + \overbrace{2 \times 0.25 \times \lambda \times 0.52}^{P2\_1, P2\_2}}{9.672} \\
&= 0.473
\end{aligned}$$

Finally, gate failure probability of NOR23 obtained by SPICE and gate level simulations is found to be:

$$\begin{aligned}
Probability\ of\ failure\ (Spice) &= 0.477 \\
Probability\ of\ failure\ (Gate) &= 0.470
\end{aligned}$$

6.5.5 NOR24

NOR24 provides protection from all sa0 faults by duplicating and scaling the `pmos` transistors, but provide protection from sa1 faults only if the transient hits `pmos` transistor P1. Fig. 6.17 shows the arrangement of transistors in NOR24 gate.

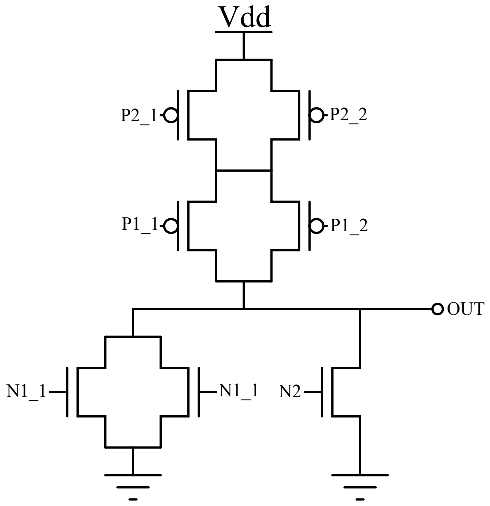


Figure 6.17: CMOS configuration of NOR24 gate

`nmos` Scaling factor (λ_1) = 2

`pmos` Scaling factor (λ_2) = 4.4

Area $= 2 \times \lambda_1 \times NW + NW + 4 \times \lambda_2 \times PW = 10.452$

Then,

$$\begin{aligned}
\mathbf{S}_{N1\_1,N1\_2} &= \phi \implies CFEP_{N1\_1,N1\_2} = 0 \\
\mathbf{S}_{N2} &= \phi \implies CFEP_{N2} = 0 \\
\mathbf{S}_{P1\_1,P1\_2} &= \{01\} \implies CFEP_{P1\_1,P1\_2} = \frac{0.25}{0.75} = 0.33 \\
\mathbf{S}_{P2\_1,P2\_2} &= \{01\} \implies CFEP_{P2\_1,P2\_2} = \frac{0.25}{0.75} = 0.33
\end{aligned}$$

Fault injection probabilities will be;

$$\begin{aligned} G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{3} CFEP_{N_{ij}} \times \frac{\lambda_1 NW_{ij}}{\sum\limits_{k=1}^{3} \lambda_1 NW_{ik}} \\ &= 0 \end{aligned}$$

$$\begin{aligned} G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{4} CFEP_{P_{ij}} \times \frac{\lambda_2 PW_{ij}}{\sum\limits_{k=1}^{4} \lambda_2 PW_{ik}} \\ &= \frac{2 \times 0.33 \times \lambda_2 \times 0.52 + 2 \times 0.33 \times \lambda_2 \times 0.52}{\sum\limits_{k=1}^{4} \lambda_2 \times 0.52} \\ &= 0.333 \end{aligned}$$

The failure probability of NOR24 gate will be:

$$\begin{aligned} Gate\ Failure\ Prob. &= \sum_{j=1}^{2} P_{DET_{N_{ij}}} \times \frac{\lambda_1 NW_{ij}}{Area} \\ &\quad + \sum_{i=1}^{4} P_{DET_{P_{ij}}} \times \frac{\lambda_2 PW_{ij}}{Area} \\ &= 0 + \sum_{i=1}^{4} P_{DET_{P_{ij}}} \times \frac{\lambda_2 PW_i}{Area} \\ &= \frac{\overbrace{2 \times 0.25 \times \lambda_2 \times 0.52}^{P1\_1,P1\_2} + \overbrace{2 \times 0.25 \times \lambda_2 \times 0.52}^{P2\_1,P2\_2}}{10.452} \end{aligned}$$

$$= 0.219$$

Finally, gate failure probability of NOR24 obtained by SPICE and gate level simulations is found to be;

$$\begin{aligned} Probability\ of\ failure\ (Spice) &= 0.218 \\ Probability\ of\ failure\ (Gate) &= 0.220 \end{aligned}$$

6.5.6 NOR25

The redundancy and scaling of transistors in NOR25 combines the NOR22 and NOR23 schemes to protect from both sa0 and sa1 faults. The Fig. 6.18 shows the CMOS representation of NOR25.

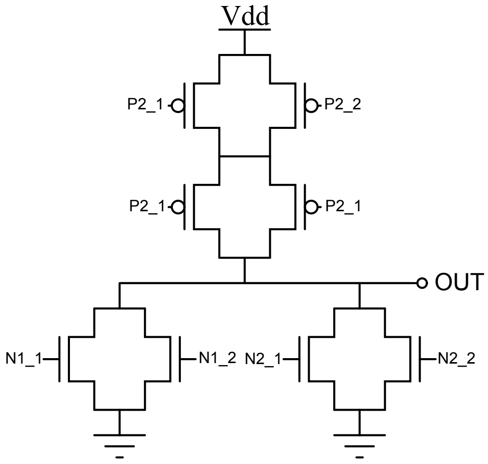


Figure 6.18: CMOS configuration of NOR25 gate

`nmos` Scaling factor $(\lambda_1) = 2$

`pmos` Scaling factor $(\lambda_2) = 4$

Area $= 4 \times \lambda_1 \times NW + 4 \times \lambda_2 \times PW = 11.232$

Fault injection probabilities will be zero for both sa0 and sa1 faults as they will be suppressed by transistors arrangement in NOR25.

$$\begin{aligned} G_i\ sa0\ inj.\ Prob. &= \sum_{j=1}^{4} CFEP_{N_{ij}} \times \frac{\lambda_1 NW_{ij}}{\sum\limits_{k=1}^{4} \lambda_1 NW_{ik}} \\ &= 0 \end{aligned}$$

$$\begin{aligned} G_i\ sa1\ inj.\ Prob. &= \sum_{j=1}^{4} CFEP_{P_{ij}} \times \frac{\lambda_2 PW_{ij}}{\sum\limits_{k=1}^{4} \lambda_2 PW_{ik}} \\ &= 0 \end{aligned}$$

The failure probability of NOR25 gate will be:

$$\begin{aligned} Gate\ Failure\ Prob. &= \sum_{j=1}^{4} P_{DET_{Nij}} \times \frac{\lambda_1 NW_{ij}}{Area} \\ &\quad + \sum_{i=1}^{4} P_{DET_{Pij}} \times \frac{\lambda_2 PW_{ij}}{Area} \end{aligned}$$

$$
\begin{aligned}
&= 0 + 0 \\
&= 0
\end{aligned}
$$

Finally, gate failure probability of NOR25 obtained by SPICE and gate level simulations is found to be:

$$
\begin{aligned}
Probability\ of\ failure\ (Spice) &= 0 \\
Probability\ of\ failure\ (Gate) &= 0
\end{aligned}
$$

6.6 Conclusion

In this chapter we discussed about the reliability evaluation framework employed in this work to compute the reliability of proposed techniques. It is observed that the proposed gate-level technique achieves reliability measures very close to the transistor-level simulations (performed using SPICE) with orders of magnitude less CPU time. Finally, in depth study of each transistor-level protection scheme is discussed in detail for 2-input NAND and NOR gates. The same idea can be extended for the 3,4-input NAND and NOR gates.

CHAPTER 7

CONCLUSION & FUTURE WORK

7.1 Conclusion

The development of an integrated soft error tolerance framework to mitigate soft errors in combinational circuits is implemented in this work. The framework consists of three techniques. Each technique applies redundancy in different design space of digital system for soft error tolerance. Additionally, to evaluate the reliability of logic circuits, a novel reliability evaluation technique is also implemented that achieves reliability measures similar to the circuit level simulations (using SPICE) with orders of magnitude less CPU time. Following objectives are achieved in this work:

Selective Transistor-Redundancy (STR) Based Fault Tolerance Technique

In this method, a selective transistor-redundancy based fault tolerance technique for combinational circuits is implemented. The technique can be applied to achieve a given circuit reliability or enhance the reliability of a circuit under a given area constraint. The technique is based on estimating the failure probability of each transistor and iteratively protecting transistors with the highest failure probability until the desired objective is achieved. Transistors are protected based on duplicating and scaling a subset of transistors necessary for providing the protection. Experimental results on LGSynth91 benchmarks demonstrate the effectiveness of the proposed technique. Compared to existing transistor sizing techniques, the proposed algorithm incurs significantly less area overhead with similar reliability measures. Better reliability results are also achieved in comparison to TMR with lower area overhead. Unlike TMR which has an area overhead of at least 3 times the area overhead of the original circuit, the area overhead of the proposed technique varies depending on the reliability of the original circuit. For some circuits, high reliability (¿ 99%) is achieved with small area overhead (¡ 10%). In addition, the reliability of the TMR technique has been enhanced significantly by protecting the voters based on applying the proposed technique. Additionally, comparison with simulation based synthesis technique further highlights the merit of the proposed method.

Double Modular Redundancy (DMR) Based Fault Tolerance Technique

In this method, a soft error tolerance technique for combinational circuits based on double modular redundancy is implemented. The technique is based on identifying the probability of occurrence of logic values 0 and 1 at each primary output of a circuit. Based on this, each output is synthesized in either the true or the complement form, and is then duplicated and a masking AND or NAND gate is used to combine the two duplicate outputs. The technique achieves higher circuit radiabilities than TMR without voter protection with significantly lower area overhead. We have also demonstrated that the combined application of the proposed DMR technique with STR technique achieves comparable circuit reliabilities with significantly lower area overhead in comparison to TMR with fully protected voters.

Furthermore, an improved DMR method based on the use of C-element (DMR-CEL) to combine the duplicate outputs is also implemented. This scheme applies redundancy by implementing the original and duplicated logic with majority phase. Reliabilities of circuits based on the proposed DMR are higher than those obtained based on DMR-CEL without C-element protection. Reliabilities of circuits based on DMR-CEL with protected C-element are similar to those obtained based on TMR with voter protection with significantly lower area overhead. It also achieves slightly better reliabilities than the combined application of the proposed DMR technique and STR technique for similar area overhead with a protection threshold of 99.5%. The advantage of the proposed DMR technique over DMR-CEL is that it uses primitive gates as masking gates which have lower impact on performance in comparison to the use of C-element.

Implication Based Redundancy Insertion Fault Tolerance Technique An implication based fault tolerance technique has been implemented. It is based on identifying implications between a set of candidate source and target gates. Then, for each implication the gain in reduction of gate fault detection profitabilities is estimated. The implication with the highest gain is selected and its corresponding functionally redundant wire is added. The process is repeated until the gain is less than a given threshold. Experimental results show that the proposed implication based fault tolerance technique reduces the failure rate of circuits with an average of 20% with an average area overhead of less than 5%. Moreover, the integrated application of the proposed implication based fault tolerance technique and the selective transistor redundancy technique achieve significantly lower area overhead in comparison to applying the selective transistor redundancy technique alone with the same achieved reliabilities. The combined application of implication and STR based soft error tolerance technique results in less area overhead in comparison to if STR is applied alone.

An Integrated Soft Error Tolerance Framework Finally, an integrated framework is developed that combines the STR technique with the DMR and the Implication based fault tolerance technique. It is observed that the DMR and the Implication based fault tolerance techniques are unable to improve the reliability of circuits beyond a certain point. Therefore, the hybrid approaches consisting of DMR+STR and Implication+STR are also implemented. It is observed that the hybrid method significantly improves the reliability, and in case of Implication+STR, it even results in less area overhead in comparison to if STR is applied alone.

Gate-level Reliability Evaluation Technique A novel method to compute the reliability of a circuit at the gate level is implemented. The circuit level simulations performed using SPICE are accurate but become very slow as the circuit size increases. However, there isn't much of an impact of circuit size in gate level simulations. The proposed technique bridges the gap between the circuit level simulations and the gate-level simulations. The proposed gate level reliability evaluation method achieves similar results in comparison to transistor level simulations with orders of magnitude reduction in CPU time.

7.2 Future Work

As for the future work, following directions can be adopted:

- So far, researchers are used to apply a single technique for soft error tolerance. It has been observed in this work that if gate level soft error tolerance techniques are combined with the transistor sizing technique, then, this results in better reliability improvement with lower area overhead in comparison to if any of the technique is applied alone. In transistor sizing technique, it is always very costly to protect for the fault on parallel transistors. The reason for this is that, all the corresponding series transistors have to be duplicated and scaled and this results in significantly higher area overhead just to protect only one type of fault. So, the future direction here is to apply protection for the faults on parallel transistors at the gate level (using modular redundancy, enhancing logical masking etc) fault tolerance techniques. In order to protect the faults on transistors connected in series, transistor sizing technique (STR) can be applied.

- Another future direction could be the potential application of the proposed Double-Modular Redundancy (DMR) scheme in high speed circuits like arithmetic circuits. DMR guarantees at least 95% reliability. For the remaining 5% failure rate, a fault detection module can be developed to detect a fault when it occurs and then the previous instruction is executed again.

REFERENCES

[1] C. Lazzari, G. Wirth, F. Kastensmidt, L. Anghel, and R. Reis, "Asymmetric transistor sizing targeting radiation-hardened circuits," *Electrical Engineering*, vol. 94, no. 1, pp. 11–18, 2012.

[2] A. H. El-Maleh and K. A. K. Daud, "Simulation-based method for synthesizing soft error tolerant combinational circuits," *IEEE Transactions on Reliability*, vol. 64, no. 3, pp. 935–948, Sept 2015.

[3] P. Dodd and L. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 583–602, June 2003.

[4] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, Sept 2005.

[5] J. F. Ziegler, H. W. Curtis, H. P. Muhlfeld, C. J. Montrose, B. Chin, M. Nicewicz, C. A. Russell, W. Y. Wang, L. B. Freeman, P. Hosier, L. E. LaFave, J. L. Walsh, J. M. Orro, G. J. Unger, J. M. Ross, T. J. O'Gorman,

B. Messina, T. D. Sullivan, A. J. Sykes, H. Yourke, T. A. Enger, V. Tolat, T. S. Scott, A. H. Taber, R. J. Sussman, W. A. Klein, and C. W. Wahaus, "IBM experiments in soft fails in computer electronics (1978-1994)," *IBM Journal of Research and Development*, vol. 40, pp. 3–18, 1996.

[6] B. S. Gill, "Design and analysis methodologies to reduce soft errors in nanometer vlsi circuits," *Doctoral Dissertation, Dept. Elect. Eng., Case Western Reserve University*, 2006.

[7] K. Mohanram and N. Touba, "Partial error masking to reduce soft error failure rate in logic circuits," in *Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Nov 2003, pp. 433–440.

[8] 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, July 2011.

[9] J. Teifel, "Self-voting dual-modular-redundancy circuits forsingle-event-transient mitigation," *IEEE Transactions on Nuclear Science*, vol. 55, no. 6, pp. 3435–3439, Dec 2008.

[10] E. Kim and N. Shanbhag, "Soft n-modular redundancy," *IEEE Transactions on Computers*, vol. 61, no. 3, pp. 323–336, March 2012.

[11] K. van Berkel, "Beware the isochronic fork," *Integration, the {VLSI} Journal*, vol. 13, no. 2, pp. 103 – 128, 1992.

[12] D. Lyons, “http://www.forbes.com/global/2000/1113/0323026a.html,” 2000.

[13] I. Klotz, “http://news.discovery.com/space/private-spaceflight/nasa-finds-cause-of-voyager-glitch.htm,” 2010.

[14] G. E. Moore, “Cramming More Components Onto Integrated Circuits,” *Proceedings of the IEEE*, vol. 86, 1998.

[15] J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams, “A defect-tolerant computer architecture: Opportunities for nanotechnology,” *Science*, vol. 280, no. 5370, pp. 1716–1721, 1998.

[16] N. Cohen, T. Sriram, N. Leland, D. Moyer, S. Butler, and R. Flatley, “Soft error considerations for deep-submicron cmos circuit applications,” in *International Electron Devices Meeting, IEDM ’99*, Dec 1999, pp. 315–318.

[17] J. Han, “Fault-tolerant architectures for nanoelectronic and quantum devices,” *Doctoral Dissertation, Delft University of Technology*, 2004.

[18] J. Wallmark and S. Marcus, “Maximum packing density and minimum size of semiconductor devices,” in *1961 International Electron Devices Meeting*, vol. 7, 1961, pp. 34–34.

[19] D. Binder, E. Smith, and A. Holman, “Satellite anomalies from galactic cosmic rays,” *IEEE Transactions on Nuclear Science*, vol. 22, no. 6, pp. 2675–2680, Dec 1975.

[20] E. Normand, J. Wert, H. Quinn, T. Fairbanks, S. Michalak, G. Grider, P. Iwanchuk, J. Morrison, S. Wender, and S. Johnson, “First record of single-event upset on ground, cray-1 computer at los alamos in 1976,” *IEEE Transactions on Nuclear Science*, vol. 57, no. 6, pp. 3114–3120, Dec 2010.

[21] T. May and M. H. Woods, “Alpha-particle-induced soft errors in dynamic memories,” *IEEE Transactions on Electron Devices*, vol. 26, no. 1, pp. 2–9, Jan 1979.

[22] D. Mavis and P. Eaton, “Soft error rate mitigation techniques for modern microcircuits,” in *40th Annual Reliability Physics Symposium Proceedings, 2002.*, 2002, pp. 216–225.

[23] *Test procedures for the measurement of single-event effects in semiconductor devices from heavy ion irradiation.* Arlington, VA: EIA, 1996. [Online]. Available: https://cds.cern.ch/record/1373656

[24] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, “Modeling the effect of technology trends on the soft error rate of combinational logic,” in *Proceedings of the International Conference on Dependable Systems and Networks, DSN ’2002*, 2002, pp. 389–398.

[25] B. Olson, D. Ball, K. Warren, L. Massengill, N. Haddad, S. Doyle, and D. McMorrow, “Simultaneous single event charge sharing and parasitic bipolar conduction in a highly-scaled sram design,” *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2132–2136, Dec 2005.

[26] N. Seifert, B. Gill, V. Zia, M. Zhang, and V. Ambrose, "On the scalability of redundancy based ser mitigation schemes," in *Proceedings of the IEEE International Conference on Integrated Circuit Design and Technology, ICICDT '07*, May 2007, pp. 1–9.

[27] O. Amusan, L. Massengill, M. Baze, B. Bhuva, A. Witulski, J. Black, A. Balasubramanian, M. Casey, D. Black, J. Ahlbin, R. Reed, and M. McCurdy, "Mitigation techniques for single-event-induced charge sharing in a 90-nm bulk cmos process," *IEEE Transactions on Device and Materials Reliability*, vol. 9, no. 2, pp. 311–317, June 2009.

[28] H. H. K. Lee, "Circuit and layout techniques for soft-error-resilient digital cmos circuits," Master's thesis, 09/2011 2011.

[29] F. Sexton, "Destructive single-event effects in semiconductor devices and ics," *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 603–621, June 2003.

[30] J. Dirk, M. Nelson, J. Ziegler, A. Thompson, and T. Zabel, "Terrestrial thermal neutrons," *Nuclear Science, IEEE Transactions on*, vol. 50, no. 6, pp. 2060–2064, Dec 2003.

[31] R. Baumann and T. Hossain, "Electronic device and process achieving a reduction in alpha particle emissions from boron-based compounds essentially free of boron-10," Mar. 7 1995, uS Patent 5,395,783. [Online]. Available: http://www.google.com/patents/US5395783

[32] A. Avižienis, “Design of fault-tolerant computers,” in *Proceedings of the Fall Joint Computer Conference, November 14-16, 1967.*, ser. AFIPS ’67 (Fall). New York, NY, USA: ACM, 1967, pp. 733–743. [Online]. Available: http://doi.acm.org/10.1145/1465611.1465708

[33] H. Konoura, Y. Mitsuyama, M. Hashimoto, and T. Onoye, “Implications of reliability enhancement achieved by fault avoidance on dynamically reconfigurable architectures,” in *21st International Conference on Field Programmable Logic and Applications*, Sept 2011, pp. 189–194.

[34] T. Anderson and B. Randell, *Computing Systems Reliability.* Cambridge University Press, 1979.

[35] P. K. Lala, Ed., *Self-checking and Fault-tolerant Digital Design.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[36] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique, M. Tahoori, and N. Wehn, “Reliable on-chip systems in the nano-era: Lessons learnt and future trends,” in *Proceedings of the 50th Annual Design Automation Conference, DAC ’13*, 2013, pp. 99:1–99:10.

[37] S. Rehman, F. Kriebel, M. Shafique, and J. Henkel, “Reliability-driven software transformations for unreliable hardware,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 11, pp. 1597–1610, Nov 2014.

[38] J. von Neumann, “Probabilistic logics and the synthesis of reliable organisms from unreliable components,” *Automata Studies*, vol. 34, pp. 43–98, 1956.

[39] Y. Qi, J. Gao, and J. A. B. Fortes, “Markov chains and probabilistic computation-a general framework for multiplexed nanoelectronic systems,” *IEEE Transactions on Nanotechnology*, vol. 4, no. 2, pp. 194–205, March 2005.

[40] J. Han and P. Jonker, “A defect-and fault-tolerant architecture for nanocomputers,” *Nanotechnology*, vol. 14, no. 2, p. 224, 2003. [Online]. Available: http://stacks.iop.org/0957-4484/14/i=2/a=324

[41] A. S. Sadek, K. Nikoli, and M. Forshaw, “Parallel information and computation with restitution for noise-tolerant nanoscale logic networks,” *Nanotechnology*, vol. 15, no. 1, p. 192, 2004. [Online]. Available: http://stacks.iop.org/0957-4484/15/i=1/a=037

[42] D. P. Siewiorek and R. S. Swarz, *Reliable Computer Systems (3rd Ed.): Design and Evaluation*. Natick, MA, USA: A. K. Peters, Ltd., 1998.

[43] A. Namazi and M. Nourani, “Reliability analysis and distributed voting for nmr nanoscale systems,” in *Proceedings of the 2nd International Design and Test Workshop, IDT ’2007*, Dec 2007, pp. 130–135.

[44] W. Pierce, *Failure-Tolerant Computer Design*. Elsevier Science, 2014. [Online]. Available: https://books.google.com.sa/books?id=B6biBQAAQBAJ

[45] P. A. Jensen, “Quadded nor logic,” *IEEE Transactions on Reliability*, vol. R-12, no. 3, pp. 22–31, Sept 1963.

[46] A. H. El-Maleh and F. C. Oughali, “A generalized modular redundancy scheme for enhancing fault tolerance of combinational circuits,” *Microelectronics Reliability*, vol. 54, no. 1, pp. 316 – 326, 2014.

[47] S. K. Shukla and R. I. Bahar, Eds., *Nano, Quantum and Molecular Computing: Implications to High Level Design and Validation.* Norwell, MA, USA: Kluwer Academic Publishers, 2004.

[48] A. Kleinosowski, V. V. Pai, V. Rangarajan, P. Ranganath, K. Kleinosowski, M. Subramony, and D. J. Lilja, “Exploring fine-grained fault tolerance for nanotechnology devices with the recursive nanobox processor grid,” *IEEE Transactions on Nanotechnology*, vol. 5, no. 5, pp. 575–586, Sept 2006.

[49] S. Mitra and E. McCluskey, “Combinational logic synthesis for diversity in duplex systems,” in *Proceedings of the International Test Conference, ITC '2000*, 2000, pp. 179–188.

[50] P. Reviriego, C. Bleakley, and J. Maestro, “Diverse double modular redundancy: A new direction for soft-error detection and correction,” *IEEE Transactions on Design & Test*, vol. 30, no. 2, pp. 87–95, April 2013.

[51] F. Smith, “A new methodology for single event transient suppression in flash {FPGAs},” *Microprocessors and Microsystems*, vol. 37, no. 3, pp. 313–318, 2013.

[52] S. Rezgui, J. WANG, E. Tung, B. Cronquist, and J. McCollum, "New methodologies for set characterization and mitigation in flash-based fpgas," *IEEE Transactions on Nuclear Science*, vol. 54, no. 6, pp. 2512–2524, Dec 2007.

[53] M. Shams, J. Ebergen, and M. Elmasry, "Modeling and comparing cmos implementations of the c-element," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 4, pp. 563–567, Dec 1998.

[54] D. E. Muller and W. S. Bartky, "A theory of asynchronous circuits," in *Proceedings of an International Symposium on the Theory of Switching. Cambridge, MA: Harvard University Press*, April April 1959, pp. 204–243.

[55] A. H. El-Maleh and A. S. Al-Qahtani, "A finite state machine based fault tolerance technique for sequential circuits," *Microelectronics Reliability*, vol. 54, no. 3, pp. 654 – 661, 2014.

[56] A. El-Maleh, B. Al-Hashimi, A. Melouki, and F. Khan, "Defect-tolerant n2-transistor structure for reliable nanoelectronic designs," *IET Computers & Digital Techniques*, vol. 3, no. 6, pp. 570–580, November 2009.

[57] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," in *Proceedings of the 17th IEEE VLSI Test Symposium, VTS '1999*, 1999, pp. 86–94.

[58] N. Alves, A. Buben, K. Nepal, 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, May 2010.

[59] N. Alves, Y. Shi, J. Dworak, R. I. Bahar, and K. Nepal, “Enhancing online error detection through area-efficient multi-site implications,” in *IEEE 29th VLSI Test Symposium (VTS), 2011*, May 2011, pp. 241–246.

[60] K. Nepal, N. Alves, J. Dworak, and R. I. Bahar, “Using implications for online error detection,” in *IEEE International Test Conference, 2008. ITC 2008.*, Oct 2008, pp. 1–10.

[61] R. I. Bahar, E. T. Lampe, and E. Macii, “Power optimization of technology-dependent circuits based on symbolic computation of logic implications,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, no. 3, pp. 267–293, Jul. 2000. [Online]. Available: http://doi.acm.org/10.1145/348019.348028

[62] 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, March 2008.

[63] B. Zhou, S. Thambipillai, and W. Zhang, “Soft error mitigation through selection of noninvert implication paths,” in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2014*, July 2014, pp. 77–82.

[64] S. Krishnaswamy, S. M. Plaza, I. L. Markov, and J. P. Hayes, “Enhancing design robustness with reliability-aware resynthesis and logic simulation,” in

2007 IEEE/ACM International Conference on Computer-Aided Design, Nov 2007, pp. 149–154.

[65] M. R. C. A. Zukoski and K. Mohanram, "Reliability-driven don't care assignment for logic synthesis," in *Design, Automation & Test in Europe Conf. Exhib. (DATE)*, Mar 2011, pp. 1–6.

[66] K. C. Wu and D. Marculescu, "Soft error rate reduction using redundancy addition and removal," in *Asia and South Pacific Design Automation Conf. (ASPDAC)*, Mar 2008, p. 559564.

[67] D. Shin and S. K. Gupta, "A new circuit simplification method for error tolerant applications," in *2011 Design, Automation & Test in Europe*, March 2011, pp. 1–6.

[68] Q. Zhou and K. Mohanram, "Gate sizing to radiation harden combinational logic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 1, pp. 155–166, Jan 2006.

[69] Y. S. Dhillon, A. U. Diril, A. Chatterjee, and A. D. Singh, "Analysis and optimization of nanometer cmos circuits for soft-error tolerance," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 514–524, May 2006.

[70] V. Joshi, R. R. Rao, D. Blaauw, and D. Sylvester, "Logic ser reduction through flip flop redesign," in *7th International Symposium on Quality Electronic Design (ISQED'06)*, March 2006, pp. 6 pp.–616.

[71] R. R. Rao, D. Blaauw, and D. Sylvester, "Soft error reduction in combinational logic using gate resizing and flipflop selection," in *2006 IEEE/ACM International Conference on Computer Aided Design*, Nov 2006, pp. 502–509.

[72] W. Sootkaneung and K. K. Saluja, "Sizing techniques for improving soft error immunity in digital circuits," in *Proceedings of ISCAS*, vol. 232, 2010.

[73] ——, "On techniques for handling soft errors in digital circuits," in *IEEE International Test Conference (ITC), 2010*, Nov 2010, pp. 1–9.

[74] ——, "Soft error reduction through gate input dependent weighted sizing in combinational circuits," in *12th International Symposium on Quality Electronic Design (ISQED)*, March 2011, pp. 1–8.

[75] 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*, March 2011, pp. 430–434.

[76] T. Calin, M. Nicolaidis, and R. Velazco, "Upset hardened memory design for submicron cmos technology," *IEEE Transactions on Nuclear Science*, vol. 43, no. 6, pp. 2874–2878, Dec 1996.

[77] M. Nicolaidis, R. Perez, and D. Alexandrescu, "Low-cost highly-robust hardened cells using blocking feedback transistors," in *Proceeding of the 26th IEEE VLSI Test Symposium, VTS '2008*, April 2008, pp. 371–376.

[78] S. Lin, Y.-B. Kim, and F. Lombardi, “A 11-transistor nanoscale cmos memory cell for hardening to soft errors,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 5, pp. 900–904, May 2011.

[79] “Predictive technology model for spice, http:// ptm.asu.edu/.”

[80] A. Dharchoudhury, S.-M. Kang, H. Cha, and J. Patel, “Fast timing simulation of transient faults in digital circuits,” in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, ICCAD ’1994*, Nov 1994, pp. 719–726.

[81] G. Messenger, “Collection of charge on junction nodes from ion tracks,” *IEEE Transactions on Nuclear Science*, vol. 29, no. 6, pp. 2024–2031, Dec 1982.

[82] H.-H. K. Lee, K. Lilja, M. Bounasser, P. Relangi, I. Linscott, U. Inan, and S. Mitra, “Leap: Layout design through error-aware transistor positioning for soft-error resilient sequential cell design,” in *Proceedings of the IEEE International Reliability Physics Symposium (IRPS)*, May 2010, pp. 203–212.

[83] H. K. Lee and D.-S. Ha, “Hope: An efficient parallel fault simulator for synchronous sequential circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1048–1058, Sep 1996.

[84] “Lgsynth’91 benchmark circuits, http://ddd.fit.cvut.cz/prj/benchmarks/.”

[85] R. K. Brayton, A. L. Sangiovanni-Vincentelli, C. T. McMullen, and G. D. Hachtel, *Logic Minimization Algorithms for VLSI Synthesis*. Norwell, MA, USA: Kluwer Academic Publishers, 1984.

[86] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Sis: A system for sequential circuit synthesis," EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M92/41, 1992.

[87] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Transactions on Computers*, vol. C-32, no. 12, pp. 1137–1144, Dec 1983.

[88] M. H. Schulz, E. Trischler, and T. M. Sarfert, "Socrates: a highly efficient automatic test pattern generation system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 126–137, Jan 1988.

[89] M. H. Schulz and E. Auth, "Improved deterministic test pattern generation with applications to redundancy identification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 7, pp. 811–816, Jul 1989.

[90] W. Kunz and D. K. Pradhan, "Recursive learning: a new implication technique for efficient solutions to cad problems-test, verification, and optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 9, pp. 1143–1158, Sep 1994.

[91] S. M. Sait and H. Youssef, *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*, 1st ed. Los

Alamitos, CA, USA: IEEE Computer Society Press, 1999.

[92] ——, *VLSI Physical Design Automation: Theory and Practice.* World Scientific, 1999.

Vitae

- Name: Ahmad Tariq Sheikh
- Nationality: Pakistani
- Date of Birth: December 2, 1983
- Email: *atsheikh@protonmail.com*
- Permenant Address: Islamabad, Pakistan.

Education

- Ph.D Computer Science & Engineering, KFUPM, Dhahran, Saudi Arabia. April 2016.
- MS Electrical Engineering, College of Electrical & Mechanical Engineering (NUST), Rawalpindi, Pakistan. September 2008.
- BS Computer Science (*System Engineering*), University of Engineering & Technology, Lahore, Pakistan. April 2005.

Journal Publications

1. Ahmad T. Sheikh, Aiman H. El-Maleh, Muhammad E.S. Elrabaa and Sadiq M. Sait, “Selective Transistor-Redundancy Based Fault Tolerance Technique for Combinational Circuits”. *IEEE Transactions on VLSI.*

2. Ahmad T. Sheikh, Aiman H. El-Maleh, “Double Modular Redundancy (DMR) Based Fault Tolerance Technique for Combinational Circuits”. *Submitted to Integration, the VLSI Journal.*

3. Ahmad T. Sheikh, Aiman H. El-Maleh, “An Integrated Fault Tolerance Technique for Combinational Circuits Based on Implications and Transistor Sizing”. *Submitted to Microelectronics Reliability.*

4. Aiman H. El-Maleh, Ahmad T. Sheikh, Sadiq M. Sait, “Binary particle swarm optimization (BPSO) based state assignment for area minimization of sequential circuits”, Applied Soft Computing, Volume 13, Issue 12, December 2013, Pages 4832-4840, ISSN 1568-4946, `http://dx.doi.org/10.1016/j.asoc.2013.08.004`.

5. Sait, S., Sheikh, A. & El-Maleh, A. “Cell Assignment in Hybrid CMOS/Nanodevices Architecture Using a PSO/SA Hybrid Algorithm”. Journal of Applied Research and Technology, Volume 11, No. 5, Pages 653-664, 2013.

Conference Publications

1. Ahmad T. Sheikh, Aiman H. El-Maleh, “Selective Transistor-Redundancy Based Fault Tolerance Technique for Combinational Circuits”. *Ph.D Forum*, Design Automation and Test in Europe (DATE), Gernoble, France. March 9-13, 2015.

2. Ahmad T. Sheikh, Aiman H. El-Maleh, “An Integrated Approach for Soft Error Tolerance of Combinational Circuits”. *WIP Poster Presentation*, 51^{st} Design Automation Conference (DAC), San Francisco, CA 94103. June 1-5, 2014.

3. Sheikh, A. & Sheikh, S. “Efficient Variants of Square Contour Algorithm for Blind Equalization of QAM Signals”. Proceedings of World Academy of Science: Engineering & Technology, Volume 51, Pages 184-192, 2009.

Patents

1. Ahmad T. Sheikh and Aiman H. El-Maleh, “METHOD OF FAULT TOLERANCE IN COMBINATIONAL CIRCUITS”, USPO application number 15/015654 (*Patent Pending*)