A GENERALIZED MODULAR REDUNDANCY SCHEME FOR ENHANCING FAULT TOLERANCE OF COMBINATIONAL CIRCUITS

BY

FERAS M. CHIKH OUGHALI

A Thesis 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

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

October 2012

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by

Feras M. Chikh Oughali

under the direction of his thesis advisor 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

MASTER OF SCIENCE IN COMPUTER ENGINEERING

Thesis Committee

[illegible]

Dr. Aiman H. El – Maleh (Advisor)

[illegible]

Dr. Sadiq M. Sait (Member)

[illegible]

Dr. Abdelhafid Bouhraoua (Member)

[illegible]

Dr.Basem AL – Madani
Department Chairman

[illegible]

Dr. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

30/12/12

Date

Dedicated to

My Parents & dearest Sisters

Acknowledgements

All sincere praises and thanks are due to Allah (SWT), for His limitless blessings on us. May Allah bestow his peace and blessings be upon his Prophet Mohammad (P.B.U.H) and his family. Acknowledgements are due to King Fahd University of Petroleum & Minerals for providing the computing resources for this research.

I would like to express my profound gratitude and appreciation to my thesis advisor Dr. Aiman H. El-Maleh for his support and endless patience in improving this work. His continuous guidance, advice and encouragement can never be forgotten. I would also like to express my appreciation to my thesis committee members, Dr. Sadiq M. Sait and Dr. Abdelhafid Bouhraoua for their feedback and constructive comments. Also, I would like to express my deepest thanks to faculty and staff members of Computer Engineering Department for their cooperation. I would like to address special thanks to my friends and fellow graduate students for their help. I am very grateful to all of you. Especially, my best friends Abdalrahman Arafeh, Mouheddin Alhaffar, Abdulrahman Idlbi, Abdulnaser Alsharaa and Mohammad Tamim.

I also thank my beloved parents and my cherished sisters for their everlasting love, support, and constant encouragement throughout my academic career. I undoubtedly could not have done this without you. Finally, thanks to everybody who contributed to this achievement in a direct or an indirect way.

Contents

List of Tables

List of Figures

THESIS ABSTRACT

Name:	Feras M. Chikh Oughali
Title:	A Generalized Modular Redundancy Scheme for Enhancing Fault Tolerance of Combinational Circuits
Major Field:	Computer Engineering
Date of Degree:	October 2012

Nano-scale devices are continuously shrinking, operating at lower voltages and higher frequencies. This makes them more susceptible to environmental perturbations and distinguished by their high dynamic fault rates. Redundancy techniques are widely used to increase the reliability of combinational logic circuits. In this work, soft error reliability is improved by using such techniques, and based on probability of occurrence for states at the outputs of circuits. A generalized modular redundancy scheme to enhance the reliability of combinational circuits is proposed. Additionally, several aspects regarding the application of this scheme are explored. This comprises types of redundant modules, complexity of correction logic and single versus multiple outputs protection. Also, a methodology for applying the generalized modular redundancy scheme is developed. Reliability analysis for various benchmarks from the LGSynth91 suite shows that the proposed methodology can achieve reliability figures higher than that of triple modular redundancy. In general, significant overhead savings are accomplished in addition to that superior reliability.

Keywords: *Reliability, Soft errors, Combinational logic circuits, Modular redundancy, Low area overhead, Logic design.*

MASTER OF SCIENCE DEGREE

King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.

October 2012

ملخص الرسـالة

الاسم: فراس محمد معتز شيخ أوغلي

العنوان: مخطط عام لاستخدام الوحدات المتكررة من أجل زيادة وثوقية الدارات التركيبية

التخصص: هندسة الحاسب الآلي

تاريخ الدرجة العلمية: تشرين الأول 2012

لا تزال الأجهزة النانوية آخذة بالتقلص مع مرور الوقت، إضـافة إلـى تطلبهـا إلـى جهد تشغيل أقل وترددات عمل أعلى. هذا ما جعلها أكثر عرضة للمؤثرات الخارجية وأصبح من سماتها ارتفـاع معدلات الخطـأ الديناميكيـة. لحل هذه المشكلة سنلجأ لتقنيات التكرار، فهي تستخدم على نطاق واسع في زيادة وثوقية الدارات التركيبية. وقد قمنا في بحثنا هذا بتحسين الوثوقية ضد الأخطاء اللينة (أو الديناميكية) بالاعتماد على تقنيات التكرار وعلى احتمالات الحالة على مخارج هذه الدارات. يقترح البحث مخططاً عاماً لاستخدام الوحدات المتكررة من أجل زيـادة وثوقيـة الدارات التركيبية، وسيبحث إضافة إلى ذلك في عدة جوانب متعلقة بتطبيق هذا المخطط. ويشتمل ذلك علـى ثلاثـة أمور: أنواع الوحدات المتكررة، تعقيد منطق التصحيح، والمقارنـة بين حمايـة مخرج واحد لقـاء حمايـة مخـارج متعددة. سنقوم علاوة على ذلك بتطوير منهجية لتطبيق المخطط العام لاستخدام الوحدات المتكررة. ويظهر تحليل الوثوقية الذي تم إجرائه لدارات مرجعيـة عدة مـن مجموعـة LGSynth91 علـى أن المنهجيـة المقترحـة قـادرة علـى تحقيـق وثوقية أعلى من تلك التـي يحققها تكرار الوحدات الثلاثـي. وعلـى العموم، فقد حققنـا وفرات كبيـرة فـي الوحـدات المتكررة المضافة باستخدام المنهجية المقترحة بالإضافة إلى تحقيق الوثوقية العالية.

Chapter 1

Introduction

In future nano-scale technology, studies indicate that high-density chips, up to an order of 10^{12} devices/cm^2 [7], will be increasingly accompanied by manufacturing defects and susceptible to dynamic faults during chip operation [8][9]. The reduced noise margin of nano-scale devices increases the effect of external fault sources such as electromagnetic interferences, thermal perturbations, and cosmic radiations. Moreover, operating at low voltages and high frequencies makes these devices more fragile and sensitive to environmental influences. Essentially, 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 will be no way to tell that a device has changed its state.

1.1 Motivation

In nanometric technologies, circuits are increasingly sensitive to various kinds of perturbations. Soft errors, a concern in the past for space applications, become a reliability issue at ground level. Alpha particles and atmospheric neutrons induce single-event upsets affecting memory cells, latches, and flip-flops, and single-event transients initiated in the combinational logic and captured by the associated latches and flip-flops. To overcome this challenge, a designer must utilize a variety of soft error mitigation schemes adapted to various circuit structures, design architectures, and design constraints [10].

The soft error rate (SER) produced by these effects may exceed the failure in time (FIT) specifications in various application domains. In such applications, soft-error mitigation schemes should be employed for both memories and logic. Duplication and comparison such as triple modular redundancy (TMR) and majority voting, or more generally, N-modular redundancy (NMR) proposed by Von Neumann [11] are the most commonly used solutions for reducing SER induced by SEUs and SETs in logic parts. Many researches have investigated increasing the reliability of circuits using redundancy schemes. Their main concern is to increase reliability while minimizing the inevitable overhead of area, power, or time. In this work, we are targeting the reliability issue in the logic based on probabilities of signals and output states. We will make use of this information to build up a reliable version from the original circuit, while maintaining the minimum possible area overhead based on a generalized modular redundancy scheme.

1.2 Problem Statement

Given a combinational logic circuit, our aim is to increase the reliability of this circuit against soft errors using a generalized modular redundancy scheme, while maintaining minimum area overhead.

1.3 Thesis Contributions

The contributions of this work can be summarized as follows:

- Develop a generalized modular redundancy scheme to enhance the reliability of combinational logic circuits against soft errors based on probabilities of states at their outputs.
- Investigate different aspects concerning the application of the generalized modular redundancy scheme. This includes types of redundant modules, complexity of voters and single versus multiple outputs protection.
- Develop a methodology for applying the generalized modular redundancy scheme to increase the reliability of combinational logic circuits with variety of sizes.

1.4 Thesis Organization

The thesis is organized as follows. Chapter 2 starts with a background about different types of faults and their models. Different soft error mitigation methods are also discussed. This includes hardened storage cells, modular redundancy and circuit-level time redundancy.

Chapter 3 introduces the generalized modular redundancy (GMR) concept. In Chapter 4, we start by discussing various aspects concerning the application of the generalized modular redundancy scheme. Based on that, the developed methodology of applying GMR to enhance the reliability of combinational circuits is then presented.

In Chapter 5, we present the work flow which has been followed to improve fault tolerance of combinational circuits. Fault model and reliability evaluation methodology along with simulation environment are also discussed in this chapter. Chapter 6 provides some analyses and evaluations of different aspects about the proposed methodology. Reliability results of various benchmarks are also reported. Finally, Chapter 7 concludes this work and suggests future work.

Chapter 2

Literature Review

2.1 Errors, Faults, and Types of Faults

As stated in [12], we term an internal state of a system an erroneous state when there exist circumstances in which further processing, by the normal algorithms of the system, will lead to a failure. The term "error" is used to designate that part of the state which is "incorrect." An error is thus an item of information; and the terms error, error detection, and error recovery are used as equivalents for erroneous state, erroneous state detection, and erroneous state recovery.

A fault is the mechanical or algorithmic cause of an error, while a potential fault is a mechanical or algorithmic construction within a system such that (under some circumstances within the specification of the use of the system) the construction will cause the system to assume an erroneous state. Faults due to hardware component failures, are often classified by duration, extent, and value. Duration refers to whether the fault is permanent or transient; extent applies to whether the effect of the fault is localized or distributed; and value indicates whether the fault creates fixed or varying erroneous logical values.

2.2 Fault Models

In engineering, models are used to bridge the gap between the physical reality and mathematical abstraction. They allow the development of analytical tools. Therefore, they are essential in the design process. Modeling of faults is highly related to the modeling of the circuit. Generally, the level refers to the degree of abstraction. Thus, the behavioral level has fewer implementation details, and fault models at this level may have no obvious correlation to manufacturing defects. The register-transfer level (RTL) or logic level consists of a netlist of gates. Stuck-at faults at this level are the most popular fault models in digital testing. Other faults at this level are bridging faults and delay faults. Transistor and other lower levels, which referred to as component levels, include stuck-open and stuck-short types of faults. Considering a MOS transistor as an ideal switch, a defect is modeled as the switch being permanently in either the open or the shorted state. This fault model assumes only one transistor to be stuck-open or stuck-short. Stuck-short is also referred to as stuck-on or stuck-closed. These models were proposed by Case [13].

Stuck-at faults are modeled by assigning a fixed (0 or 1) value to a signal line in the circuit. A signal line can be an input or an output of a logic gate or a flip-flop. The most popular forms are the single stuck-at faults, i.e., a line can have two faults: stuck-at-1 and stuck-at-0. There are three assumptions assumed for single stuck-at faults: (1) only one line is faulty, (2) the faulty line is permanently set to either 0 or 1, and (3) the fault can be at an input or output of a gate. In general, several stuck-at faults can be simultaneously present in a circuit. A circuit with n lines can have $(3^n - 1)$ possible stuck line combinations. This is because each line can be in one of the three states: stuck-at-1, stuck-at-0, or fault-free. All combinations except one, having all lines in fault-free states, are counted as faults. It is clearly evident that even a moderate value of n will give an enormously large number of multiple stuck-at faults. For this reason, it is a common practice to model only single stuck-at faults, where an n-line circuit can have at most $2n$ single stuck-at faults [14]. In addition, a large percentage of multiple stuck-at faults are detected by single stuck-at fault tests.

2.3 Fault Avoidance and Fault Tolerance

The traditional approach to achieving reliable computing systems has been based largely on fault avoidance (termed fault intolerance by Avizienis [15]) which includes: utilization of the most reliable components, elimination of expected forms of interference and carrying out of comprehensive testing to eliminate hardware and software design faults. Many recent methods have been proposed to perform fault avoidance through reconfiguration. These methods are classified into two groups: (1) Hardware-oriented methods, where a faulty basic element (BE) is automatically swapped with a spare using additional wires, switches, and controllers, along the replacement policy. (2) Reconfiguration-oriented, where reconfigurability is exploited for fault avoidance with partial mapping modification [16].

An alternative approach to fault avoidance is that of 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 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 [12].

The objective of fault tolerance is either to mask, or to recover from, faults once they have been detected [17]. Many researches on reliable systems are concerned with the detection of faults using error detecting and correcting codes or fault detecting and self repairing circuits. The tolerance itself is achieved using redundancy techniques.

2.4 Soft Errors

When high-energy neutrons (present in terrestrial cosmic radiation) or alpha particles (originating from impurities in the packaging materials) strike a sensitive region in a semiconductor, they generate a dense local track of electron-hole pairs. These electron-hole pairs are then collected at a p-n junction, and create a single event transient (SET). This unexpected current pulse of a short duration, under some conditions, may be misinterpreted by the circuit as a valid signal, and result in an incorrect state or output, thus producing a soft error [3]. As this type of faults does not reflect a permanent failure, hence soft or transient terms are used.

Historically, soft errors have been of great concern in memory cells, that are much more susceptible to particle strikes than combinational logic, where SET are frequently masked before reaching an output or a storage element [18] [19]. However, technological trends such as faster clock rates, smaller device sizes, lower supply voltages, and shallower logic depths are drastically reducing SET masking, and remarkably increasing the occurrence of soft errors in combinational logic [20]. The single event transient may propagate through the combinational logic, and errors may or may not reach a storage element or an output depending on the following factors [21]:

Logical masking: The hazard may not propagate because there is not any sensitized path from the node where the strike happened to any output of the combinational logic circuit.

Temporal masking: As the hazard propagates towards a sequential element, it will form a sort of noise on the data input of that sequential element. This noise may reach the sequential element outside its latching window. Hence, the error will not be latched and there will be no soft error.

Electrical masking: Since all CMOS circuits have limited bandwidths, hazards with bandwidths greater than the cut-off frequency will be attenuated. The amplitude of the hazard pulse will be reduced, and eventually the hazard pulse may disappear. However, since most logic gates are nonlinear circuits with a substantial voltage gain, low-frequency pulses with sufficient initial amplitude will be amplified [22].

As stated earlier, with the increase in clock rates, and the reduction in both feature sizes and supply voltages the effect of those masking is becoming more limited.

The most deceitful form of single event transient errors is silent data corruption (SDC), where a fault causes the system to produce erroneous outputs. To avoid silent data corruption, designers often employ basic error detection mechanisms, such as parity. With the ability to detect a fault but not correct it, we can avoid generating incorrect outputs, but cannot recover when an error happens. That is to say, error detection mechanism does not reduce the overall error rate, but does provide fail-stop behavior and thereby avoids any data corruption. Such errors are categorized as detected unrecoverable errors (DUE) in [23].

There are two techniques that can be used to reduce soft errors impact: (1) error detection and retry and (2) error masking. Error detection and retry involves using on-line error detection circuitry [24] [25] [26] which monitors the outputs of a circuit for the occurrence of an error. If an error is observed, the system recovers through rollback and retry, therefore preventing a fault from happening. Error masking employs circuitry that masks errors using schemes such as: quadded logic [27], interwoven logic [28], and triple modular redundancy (TMR) [26]. For real-time systems, it may not be possible to do error detection and retry, therefore error masking is the only available choice. Sometimes, the cost of implementing error detection and retry may be comparable to that of implementing error masking schemes [3].

2.5 Soft Error Rate

Currently, the industry specifies soft error rates in terms of silent data corruption (SDC) and detected unrecoverable errors (DUE) numbers. Both SDC and DUE rates are typically expressed in FIT (Failure(s) in Time). One FIT expresses one error in a billion (10^9) hours. FIT rates are additive, so we can compute the SDC or DUE FIT rate of a chip or a system by summing the SDC or DUE FIT rates of all its components. The sum of SDC and DUE FIT is usually referred to as the soft error rate (SER) of a chip [23]. The additive property of FIT makes it convenient for calculations. However, mean time to failure (MTTF) is another important indicator. MTTF is inversely related to FIT. A FIT rate of 1000 is equivalent to MTTF of 114 years ($10^9/(1000 \times 24 \times 365)$). Vendors usually set soft error rate budgets for their chips or systems based on target market requirements. For example, IBM targets 114 SDC FIT (1000 yr MTTF), for its Power4 systems [29].

Soft error rate (SER) estimation can be performed at different levels of design hierarchies, through modeling soft errors across the device, logic, and architectural levels. Device level abstracts the soft error impact as a transient pulse waveform. Soft errors are modeled by using nuclear and device physics tools, with an aim of creating a transient current waveform library that captures different process and operating conditions that impact the soft error rate. SPICE simulation is usually used to obtain the estimation at this level of abstraction. At the logic level, estimation of SER is based on attempting to capture electrical, logic, and latch window masking models. This can be used next to compute SET occurrence rate, the error propagation probability (EPP), and the error latching probability. Architectural level solutions may be more effective than circuit or logic level solutions, because the definition of what constitutes an error typically lies in the architecture. For example, in microprocessor architecture, a strike on a branch prediction unit does not result in an error in the microprocessor.

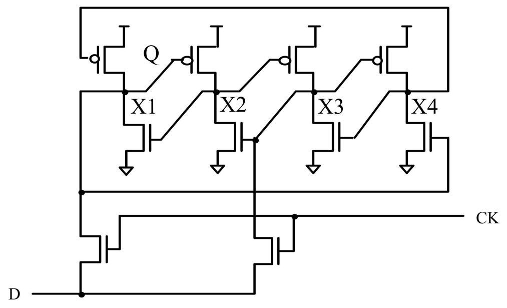


Figure 2.1: Example of hardened storage cells: DICE [1].

2.6 Soft Error Mitigation

Radiation affects logic due to single event transient in combinational logic and single event upset in sequential cells. Solutions for these effects comprise hardened sequential cells, self-checking design, modular redundancy, and circuit-level time redundancy.

2.6.1 Hardened Storage Cells

Hardened storage cells like SRAM cells, latches, and flip-flops preserve their state even if the state of one of their nodes is altered by a transient soft error. Several hardened storage cells have been proposed in the literature. Figure 2.1 shows the DICE cell [1] which is robust against any single event disturbing a single node of the cell.

Another hardening approach is proposed in [30]. In [2] this solution was adapted to implement a new memory cell. This proposed hardening design makes use of 11 CMOS transistors. The proposed hardened memory cell overcomes the problems

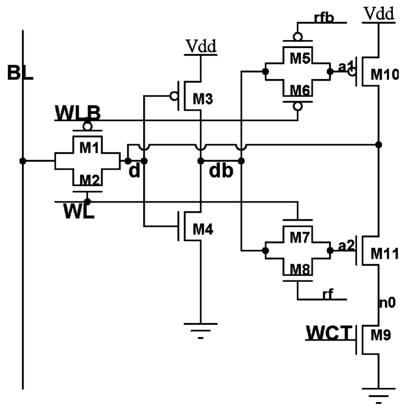


Figure 2.2: 11-transistor nanoscale CMOS memory cell [2].

associated with the previous designs by utilizing novel access and refreshing mechanisms. The cell was shown to provide similar protection against SEUs as DICE while requiring 20% less area and a 55% reduction of speed-power product. Details of several hardened cells can be found in [31].

2.6.2 Modular Redundancy

Protecting combinational logic is a complex task when compared to protecting memories, latches, and flip-flops, and may involve quite high hardware cost. Therefore, it is mandatory to properly select the protection scheme in order to meet design requirements in terms of area, speed, power and reliability. Redundancy techniques (duplication and comparison) such as triple modular redundancy (TMR) [26] are a possible solution for masking errors produced in the logic, despite its high area and power penalty. TMR is a special case of the generalized NMR system. An NMR system is a system that consists of N identical modules which are fed to a majority

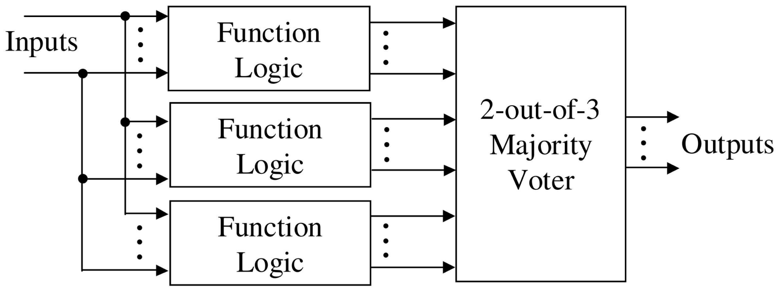


Figure 2.3: Block diagram for TMR-based error masking [3].

voter. TMR is a system where $N = 3$, which consists of three identical modules whose outputs are voted on.

If 2 out of 3 modules produce the same results, this implies that the majority of the modules produce correct results and the system will be functioning correctly. In TMR, all the three identical modules perform the same operation, and a voter accepts outputs from all three modules, producing a majority vote at its output as shown in Figure 2.3. TMR is heavily used in practice whenever the reliability of the circuit is a crucial demand, especially when single faults are needed to be protected. Recently, Xilinx introduced its triple module redundancy technology which is developed to address the special requirements of FPGAs in high-radiation environments. Designed for space applications and proven through numerous mission-critical projects, TMR provides full SEU and SET immunity for any Virtex-4 space-grade FPGA design. This technique automatically builds TMR into Xilinx FPGA designs, providing complete SEU and SET protection [32].

2.6.3 Error Detection using Checkers

Another approach is to extend the function of the combinational blocks to generate error detection code outputs in addition to their regular outputs, and use a code checker to detect errors [33]. The general idea for achieving error detection and correction is to add some redundancy (i.e., some extra bits), which afterward can be used to check consistency of the output bits, and to recover bits determined erroneous. Checkers used this concept to detect if an output codeword is incorrect and correct it if possible [34]. Parity check, cyclic redundancy check (CRC) and Checksum are types of redundancy checks used in data communication for error detection. Forward error correction is an example of error correction. In theory it is possible to correct any error, however error correcting codes are more sophisticated than error detection codes and require more redundancy bits.

2.6.4 Circuit-Level Time Redundancy

A first scheme proposed in [35] [36] uses double sampling to observe an output of a combinational circuit at two different instants. This is done by adding a redundant latch to each circuit output, and driving the redundant latches by using a delayed clock signal, as shown in Figure 2.4. The delayed clock can be produced by adding some delay elements on the normal clock signal. Another more efficient alternative is to use the two edges or the two levels of the clock signal to activate the regular

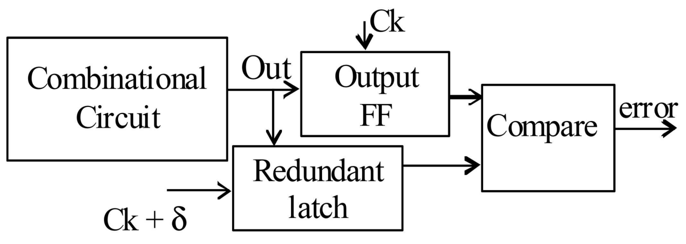


Figure 2.4: A detection technique using a redundant latch and a delayed clock.

and the redundant storage element. Upon error detection, the latest operation has to be repeated (retry). Also, as suggested in [35], to avoid the occurrence of the same error in case of timing faults, the clock frequency is reduced during retry.

In this double sampling scheme, the delay element δ can be added on the combinational circuit output rather than on the clock as in [35] [36]. This creates a second sampling instant that precedes by δ the regular sampling instant. As we can see, for real-time systems, it may not be possible to do error detection and retry, thus error masking and online error detection and correction are the preferable options.

2.7 Related Work

In a work done by Mohanram et al. [3], two soft error rate reduction heuristics were introduced. They are cluster sharing reduction and dominant value reduction. Cluster sharing reduction is based upon two observations. The first is that soft error susceptibility of certain nodes in the logic circuit can be orders of magnitude higher than that of other nodes in the design. The second is that these nodes tend to be clustered together. Thus, the cluster sharing reduction heuristic involves selecting such clusters of nodes (with low soft error susceptibility), so that the clusters' logic can be shared across the three copies used to realize TMR. The clusters are removed from two out of the three copies of the TMR design and they are driven by the cluster from a single copy only. If a particle strike occurs in the non-triplicated portion of the design, it will (in the presence of a sensitized path) propagate to the outputs of all three copies and thus goes undetected. However, any particle strike that occurs on a node in the triplicated logic portion of the circuit will be masked by the 2-out-of-3 voter. Therefore, by carefully selecting the clusters of nodes with low soft error susceptibility, nodes with the highest soft error susceptibility will be in the triplicated logic portion of the circuit, thereby giving a cost-effective reduction in the soft error failure rate.

Dominant value reduction is the second technique, and it works as follows. First of all, the soft error failure rates at the outputs of the circuit are computed, where each of the primary outputs has either logic 0 or logic 1. For example, let primary output O_i has a logic 1 failure rate that is an order of magnitude higher than its logic 0 failure rate. Then, the authors consider the approach to error masking where just two copies of the logic circuit are used, and the 2-out-of-3 majority voter is replaced by an OR gate as shown in Figure 2.5. Any particle strike that causes a $1 \rightarrow 0 \rightarrow 1$ transient to appear at the output O_i is guaranteed to be masked. However, a $0 \rightarrow 1 \rightarrow 0$ transient will not be masked. Thus, while the logic 1 failure

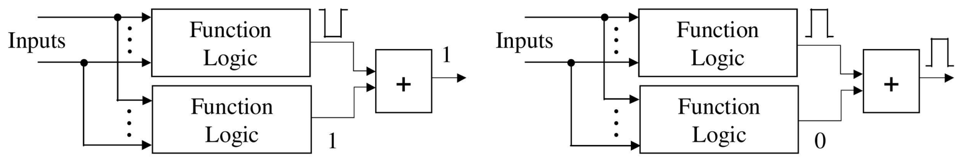


Figure 2.5: Example of dominant value reduction for error masking [3].

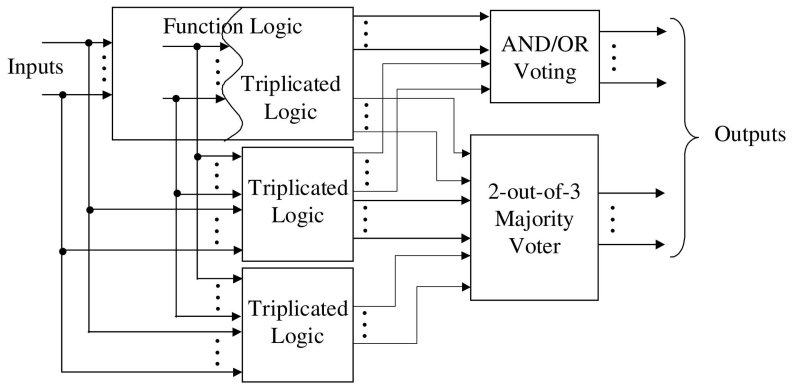


Figure 2.6: Partial error masking scheme [3].

rate of the primary output O_i is reduced to 0, the logic 0 failure rate is not. However, since the logic 1 failure rate is an order of magnitude higher than the logic 0 failure rate, the reduction in failure rate is considerably high.

These two reduction procedures are then combined to form the partial error masking scheme. It starts with a TMR realization for error masking that is first reduced using cluster sharing. The soft error failure rate of the resulting implementation is then estimated along with the area overhead. Dominant value masking is then used to further reduce area overhead. Figure 2.6 depicts partial error masking scheme.

Another similar approach to the dominant value reduction is also proposed by Krishnaswamy et al. in [37]. Their proposed technique increases logic masking at high-impact nodes by exploiting redundancy already present in the circuit as identified by covering relationships among existing nodes. Just like partial TMR, which replicates vulnerable nodes, this technique incurs a smaller area overhead as it increases logic masking through the addition of single gates. In their work, they use the notation $f \subseteq g$ when g covers f, i.e., g is 1 for every input vector that makes $f = 1$. This relation is then generalized to:

$$f\&C(g) \subseteq g\&C(g) \tag{2.1}$$

Here $C(g) =\sim ODC(g)$, is the Boolean function representing the care set of g. In other words, g covers f if and only if g is 1 or a don't-care wherever f is 1. They define node g to be an anti-cover of node f when:

$$g\&C(g) \subseteq f\&C(g) \tag{2.2}$$

Based on the idea that nodes' influence on SER is proportional to the probability that faults arrive at the node, and the probability that those faults are observed as errors at the output, they compute the impact of each gate. Then, for a high-impact node x, they find other nodes that it covers or anti-covers. So, given a candidate node y covered by x, they add redundant logic by transforming node x into $OR(x, y)$ because $y \subseteq x$ implies $OR(x, y) = x$. Similarly, if x is an anti-cover of y, they transform node x into $AND(x, y)$.

In [38], the authors proposed a method for reliability improvement based on modular redundancy schemes. This method selects the best subset among possible redundant architectures. It is built upon the progressive module redundancy technique and the block grading concept. To understand the block grading concept, let us consider a circuit which consists of a certain number K of blocks b_1, b_2, . . ., b_K. The reliability of each block b_i is denoted by q_i. The reliability of such a circuit R depends on the reliabilities of its constituent blocks. By improving the reliability of a single block b_i as in Equation 2.3, the reliability of the circuit will be improved as in Equation 2.4. Δq_i stands for the individual reliability improvement of block b_i. Likewise, ΔR_i expresses the global reliability increase due to reliability improvement of block b_i. ΔR_i scales with the weight (noted w_i) of Δq_i in the new global reliability calculation. The greater w_i is, the greater the ΔR_i will be. The w_i values are integers in the range $[1, K]$.

$$q_i^+ = q_i + \Delta q_i \tag{2.3}$$

$$R_i^+ = R_i + \Delta R_i \tag{2.4}$$

Efficiency is achieved by taking into account grades of blocks with respect to reliability, by adding redundancy progressively and by considering mixed modular redundancy. The idea is to add redundancy first on the blocks that have higher weights, because they contribute to higher reliability improvements, and then on the blocks with lower weights. The proposed method presents a shortcut by avoiding analyses of all the possible redundant architectures exhaustively. This method is not constrained on TMR but extends to 5MR. Compared with previous works such as selective module redundant techniques described in [39], the proposed method does not need to analyze the logic implications of each logic gate, and it could be combined with other soft error mitigation techniques at logic and circuit level.

In [40] the authors present a design technique for hardening circuits mapped onto FPGAs. They detect SEU sensitive gates of a given circuit based on signal probabilities; the signal probability of a line is the probability of this line having a value of 1. After characterizing the input environment, input signals probabilities are then propagated to compute the signal probability of each internal node at the gate level. A gate is identified as a sensitive gate if its SEU sensitive probability is greater than the threshold specified by the user. Triple modular redundancy is then introduced for each sensitive gate. Additionally, a majority voter is introduced between gates depending on the fanout connections of these sensitive gates. The effectiveness of selectively introducing TMR for SEU mitigation is highly dependant on input signal probabilities and the nature of the circuit. The advantage of selective TMR is clear for those circuits wherein the size of the SEU sensitive sub-circuit is much smaller compared to the original circuit.

In [4], the authors introduced the idea of increasing sequential circuit reliability

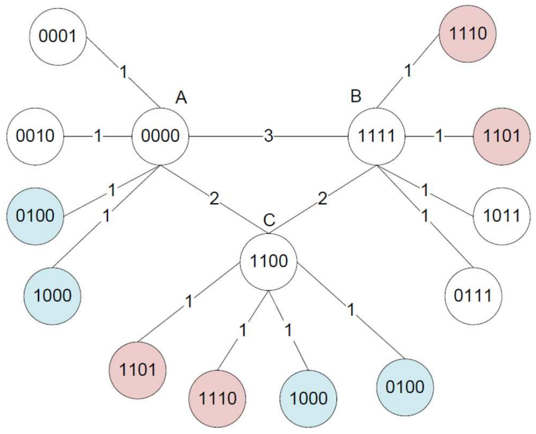


Figure 2.7: Incorrect encoding for 3 states with redundancy [4].

by introducing redundant equivalent states to states with high probability of occurrence in sequential circuit finite state machine. To maintain the same operation of the original FSM, the newly added redundant states have the same input, next state, and output as the original states. Other states with low probability are kept without protection to minimize the area overhead. The authors 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 analysis of the problem suggests that the following requirements have to be met:

- The Hamming distance between protected states codes should be at least 3.
- The Hamming distance between codes of normal states and protected states should be at least 2.

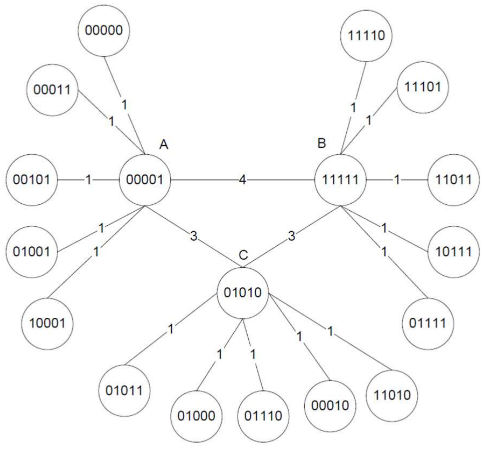


Figure 2.8: Correct encoding for 3 states with redundancy [4].

These requirements are a must in order to be able to add redundant states without code overlapping with protected or normal stats. To illustrate this, let us assume that 3 states need to be protected with no normal states. Thus, using only 4-bits to encode them will result in overlapping between some of the redundant states. Figure 2.7 shows that states A and C have the same redundant states namely, 0100 and 1000. Similarly, states B and C have the same redundant states namely, 1101 and 1110. Therefore, the correct encoding is to use 5-bits instead of 4-bits. One possible encoding is shown in Figure 2.8.

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. He 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. For more details about the work, you can refer to [4].

The authors in [5] suggest that simple replication of micro-architecture modules will no longer suffice, as all replicated modules will have faults. As a result, designers will have to devise a method for operating with defective structures, given that all the structures will have defects. In [41], the authors introduce the concept of history index of correct computation (HICC), where they developed a technique to tolerate the expected flawed nano-chips at run time. The HICC is a measure of a hardware unit's reliability. The HICC module transmits the correct computation based on the history indices of redundant units that implement the identical function. The redundant unit with the highest history index is considered to be the most reliable one, and is selected to transmit its computation. Other redundant units that implement the same function are considered unreliable, and their computation is ignored. The history index of every redundant unit is updated based on majority voting.

To demonstrate how the HICC concept works, consider three copies of an 8-bit ALU with no error correction or detection which are depicted in Figure 2.9 as units A, B and C. A decision unit is located inside the HICC module. The result selector, decision unit, receives results from the three identical units along with their stored history indices, and decides which result is correct, based on the one with the highest history index. The result from the unit with the highest index is transmitted. In this way, computation is based on correct computation history rather than merely on some voting process. The history index is updated based on bitwise majority

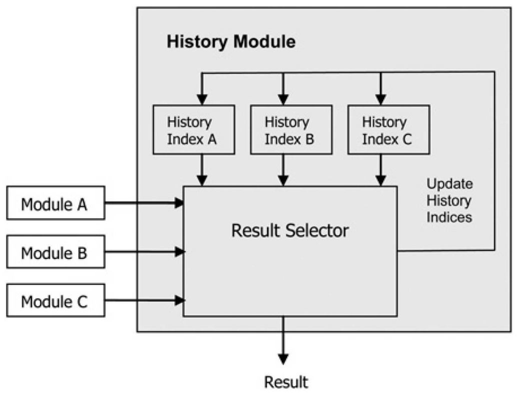


Figure 2.9: HICC module [5].

voting. The history index of each of the redundant units is incremented by 1 when its computation is identical to that of the majority and is decremented by 1 when it is not.

Moreover, maximizing the reliability of a system based on nano-devices may require a combination of techniques [42]. Previous results that used error correcting codes (ECCs) and TMR at the bit and module levels demonstrated that recursive TMR at both levels has the best resilience to noise [43]. 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 Figure 2.10 and Figure 2.11. 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.

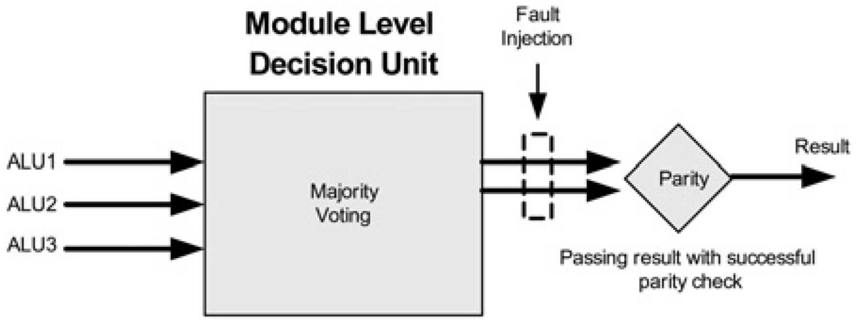


Figure 2.10: Enhanced majority voting with parity checking at the module level [5].

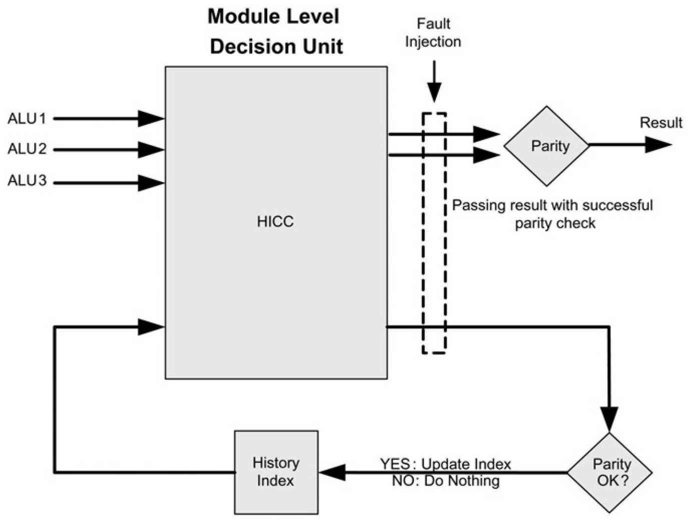


Figure 2.11: Enhanced HICC unit with parity checking at the module level [5].

In [6], the authors investigate defect tolerance based on adding redundancy at the transistor-level for electronic circuits. Their work is focused on transistor stuck-open, stuck-short and bridges between gates of transistors. In order to tolerate single defective transistors, each transistor, A, is replaced by a quadded-transistor structure implementing either the logic function $(A+A)(A+A)$ or the logic function $(AA)+(AA)$. In both of the quadded-transistor structures, any single transistor defect (stuck-open, stuck-short, AND/OR-bridge) will not change the logic behavior, and hence the defect is tolerated.

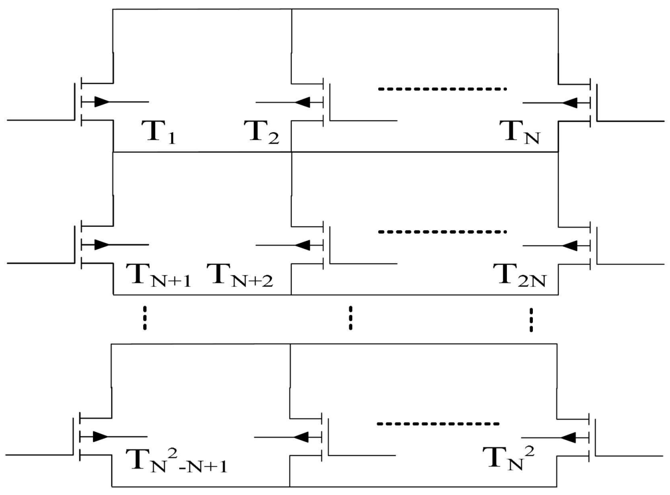


Figure 2.12: Defect-tolerant N^2-transistor structure [6].

The quadded-transistor structure is then generalized to an N^2-transistor structure, where $N \geq 2$. An N^2-transistor structure is composed of N blocks connected in series with each block composed of N parallel transistors, as shown in Figure 2.12. An N^2-transistor structure guarantees defect tolerance of all defects of multiplicity less than or equal to $(N - 1)$ in the structure. Hence, a large number of multiple defects can be tolerated in a circuit implemented based on these structures. As demonstrated by the authors, the investigated technique achieves significantly higher defect tolerance than recently reported nanoelectronics defect-tolerant techniques (even with up to 4 to 5 times more transistor defect probability) and at reduced area overhead.

In [44], two algorithms for the selective assignment of input don't cares (DCs) are proposed to enhance input error resilience. The authors focused on input errors due to propagated failures from previous blocks. Both algorithms determine 0/1 assignments for the most critical DC terms. This work is motivated by the observation that reliability-driven assignment of DCs can improve input error resilience in logic circuits. They showed that selective reliability-driven DC assignment can enhance robustness and avoid the high overheads associated with complete reliability-driven DC assignment.

In [45], the authors introduced the idea of synthesizing combinational circuits to increase their tolerance for soft errors. Their idea is based on extracting sub-circuits from the original multi-level circuit and re-synthesizing each extracted sub-circuit to increase fault masking. After that, the re-synthesized sub-circuits are merged back to the original circuit. As a result, the overall reliability of the original circuit is enhanced. The proposed scheme provides a heuristic that first finds the best irredundant set of cubes to cover an extracted sub-circuit minterms such that fault masking for single fault is maximized especially for minterms with high probability of occurrence. Then, an extra number of cubes can be added as redundant cubes to the cover such that they have a significant effect on maximizing error masking. After that, a technique based on modification of the fast extraction algorithm is proposed to enhance area overhead to the optimized circuits obtained in the previous step. An average failure rate reduction of 26% is obtained compared to the original circuit when the number of injected faults in the new re-synthesized circuit is proportional to area compared to original area. An average area overhead of 61% is added compared to the original circuit.

Chapter 3

Generalized Modular Redundancy

3.1 Introduction

Given a combinational logic circuit with multiple inputs and outputs, we will use "state" notation to denote a given output combination, Figure 3.1. Of course, each state at the output of a circuit has a probability of occurrence. This probability is attributable to the structure of the circuit and input vectors provided by the environment. Based on these probabilities, we would like to increase the reliability of the logic by protecting those states with high probability of occurrence. Protecting only these states will help us to increase the reliability of the circuit while saving some area overhead, by not protecting states with low probability of occurrence. This work is an extension of a previous work for enhancing sequential circuits reliability [4]. The author proposed a technique of protecting states with high probability

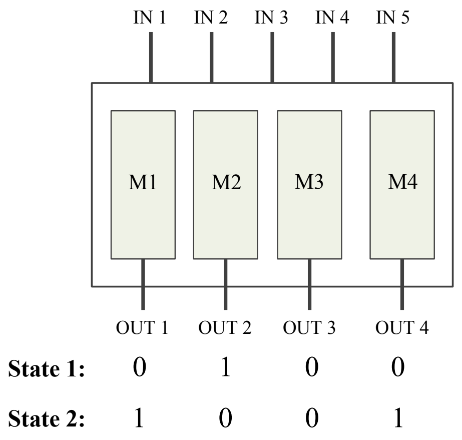


Figure 3.1: States in combinational logic circuits.

of occurrence to enhance fault tolerance for sequential circuits. The protection is done by introducing redundant states to the finite states machine (FSM). 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 protected states.

3.2 Types of States

Based on their probability of occurrence, states at the primary outputs of a combinational logic circuit can be classified into two types: dominant states and states which are not dominant, i.e., with low or moderate probability of occurrence. When the probability of occurrence for a certain state is close to an order of magnitude higher than the probability of occurrence for other states it is considered as a dominant state. Inferior states are states with low probability of occurrence. Therefore, dominant states will be considered for reliability enhancement due to their highly skewed susceptibility to soft errors.

3.3 Protection of Dominant States

Like the introduction of redundant states for sequential circuits [4], redundant modules have to be introduced in combinational circuits. The author devised some requirements that have to be met:

- The Hamming distance for each pair of protected states' codes has to be at least 3.
- The Hamming distance between normal (unprotected) states'codes and protected states' codes has to be at least 2.
- The combinational logic which implements each output has to be not shared with the others.

Protected states, in sequential circuits, include original dominant states; normal states include the remaining states in the finite state machine. In order to detect all single errors for protected states, the Hamming distance between protected state A, and its redundant states should be 1. Similarly for protected state B, the Hamming distance between it and any of its redundant states should be 1. To insure that no two states have the same code, the distance between states A and B must be kept at least 3. For the same reason, the distance between protected and normal states

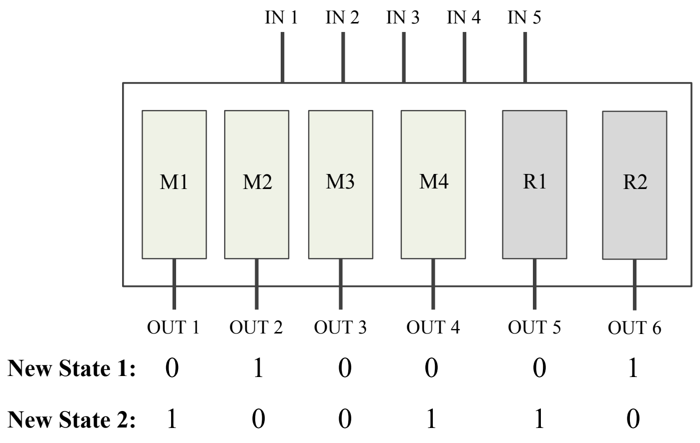


Figure 3.2: New States after introducing redundant modules.

has to be at least 2. Finally, no logic sharing is allowed between different outputs, so that no single error can propagate to more than one output.

As we have stated earlier, a "state" in the combinational logic denotes a given output combination. Dominant state or states are states with high probability of occurrence, while inferior states are states with low probability of occurrence. To increase the reliability of the circuit, extra redundant modules will be introduced to the logic. New states now consist of original outputs and the outputs of redundant modules, as shown in Figure 3.2. Modules *R1* and *R2* are redundant modules. Original outputs plus outputs of redundant modules will be then fed to another logic to produce the protected outputs, as shown in Figure 3.3.

A tool was developed to satisfy the first two requirements for enhancing sequential circuits reliability [4]. In sequential circuits, codes can be assigned to any state; so the developed tool explores the space for an assignment that meets the mentioned requirements with a minimum number of bits. Figure 3.4 shows desired Hamming

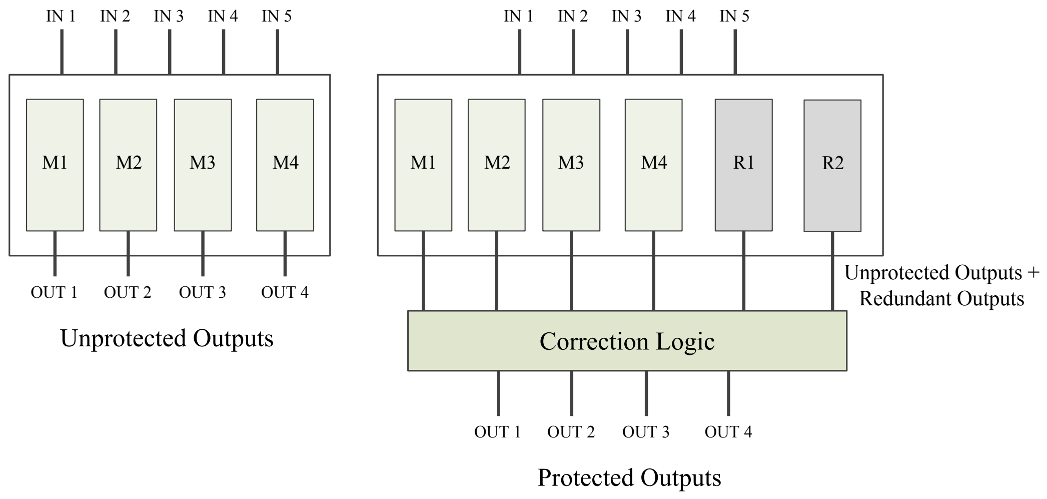


Figure 3.3: Protected outputs after introducing redundant modules and correction logic.

distances between different types of states. However, in combinational logic, a state represents an output combination. Therefore, redundant modules should be carefully selected in order to meet previously stated requirements. Two options are available for adding redundant modules: original module replication and introducing new modules.

- In module replication, a redundant module is essentially a replica of one of the original modules in the circuit. Original modules are replicated and introduced as redundant modules. Hence, a decision has to be made to determine which modules to be replicated and for how many times, in order to satisfy the requirements for combinational logic.

- For newly introduced redundant modules, modules are specially customized so that new states can satisfy the requirements. In this case, we are not limited

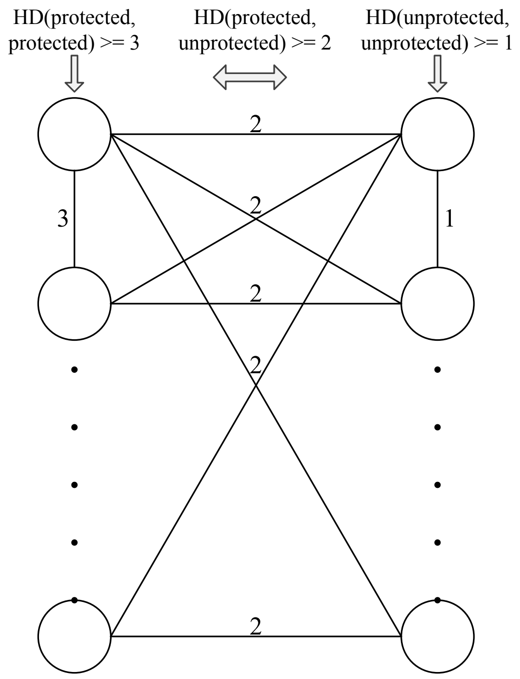

HD(protected, protected) >= 3
HD(protected, unprotected) >= 2
HD(unprotected, unprotected) >= 1
2
3
2
1
2
2
2

Figure 3.4: Desired Hamming distance between different types of states.

by the options offered by modules replication. Instead, we might be able to add fewer redundant modules by customizing them with respect to dominant protected states.

3.4 Protecting Single Output

Let us consider the case of a single output circuit or one module. States at the output are the simple state 0 (logic 0) or state 1 (logic 1). In some cases, the probability of having one state is far more than the probability of having the other. So, the dominant state will be selected for protection, while the other will not. Assume that logic 0 is dominant at the output. New redundant modules have to be introduced to the logic. We can see that by replicating the module one time the first two requirements are satisfied. After adding the extra module, state “0” will become “00” and state “1” will become “11”. There is only one state to protect, so the first requirement is already satisfied. Hamming distance between the protected state and the other state equals to two, so the second requirement is also satisfied. The third requirement is met while synthesizing the circuit.

If an error hits and alters the output, while the circuit is at state “00”, the resulting state will be “01” or “10”. In order to obtain the protected output, both original and redundant outputs will be fed to another logic. In this case, dominance of state 0, it turns out that this logic is an AND gate. Any particle strike that causes a $0 \rightarrow 1 \rightarrow 0$ transient to appear at the output is guaranteed to be masked. However, a $1 \rightarrow 0 \rightarrow 1$ transient will not be masked. Therefore, logic 0 failure rate of the primary output is reduced to 0, while logic 1 failure rate is not. Figure 3.5(a) demonstrates these findings. The same observations can be found when the dominant state is state “1”. However, the logic which will produce the protected output is an OR gate, Figure 3.5(b). The use of these dominant values or states along with simple voters (And, OR) has been discussed earlier in the literature [3].

Let’s consider the case where there is no dominant state at the primary output, i.e., the probability of having state “0” is equal or close to the probability of having state “1”. Obviously, we need to protect the two states. We can see that by replicating the module two times the first two requirements are satisfied. After adding the

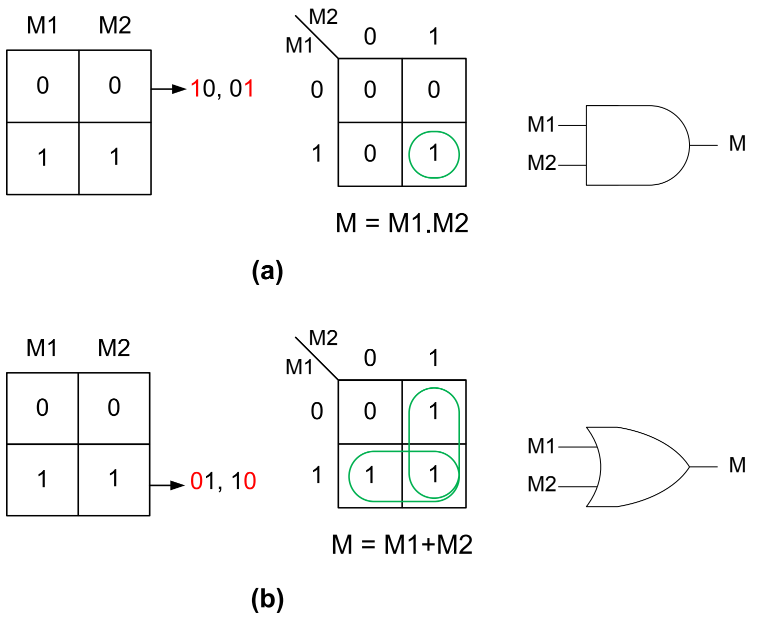


Figure 3.5: Protecting one state in single output circuits.

redundant modules, state "0" will become "000" and state "1" will become "111". Hamming distance between the protected states equals to three, so the first requirement is satisfied. We don't have to worry about the second requirement as all states are considered for protection. Again, to obtain the protected output, all original and redundant outputs will be fed to another logic. Figure 3.6 shows the result of replicating the module and the logic needed to obtain the protected output.

In this case, protecting all states, we had to replicate the original module two times; so we ended up with three replicas. Moreover, the logic that produces the protected output is essentially the 2-out-of-3 majority voter. So, with the purpose of protecting all states at the primary output we ended up applying TMR, where we have three exact copies of the original circuit plus a majority voter.

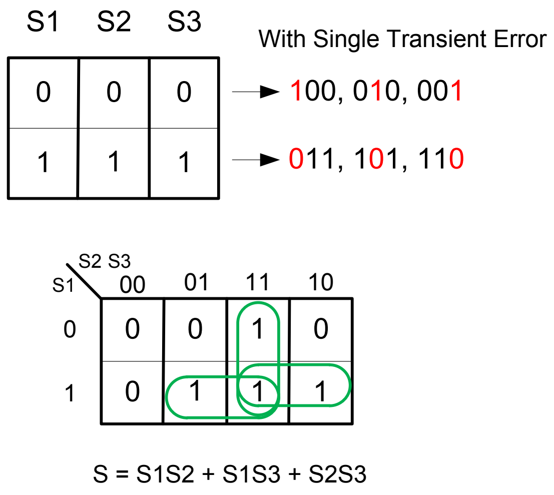


Figure 3.6: Protecting all states in single output circuits.

Equality comparator serves as an excellent example for protecting single output circuits. A 6-bit equality comparator has two inputs, each of which is 6-bit long. It has a single output; the output equals to “1” if and only if the two inputs are equal. Thus, the probability of having a “1” at the output is: $Prob(1) = 2^6/2^{12} = 0.015$, while the probability of having a “0” equals to: $Prob(0) = 1-Prob(1) = 1-0.015 = 0.985$ which is an order of magnitude greater than Prob(1). So, equality comparator is eligible for dominant value protection. Protecting the state which is vulnerable to soft errors the most involves adding one extra module. This will save us some area overhead, another extra module, as opposed to TMR wherein we need two redundant modules.

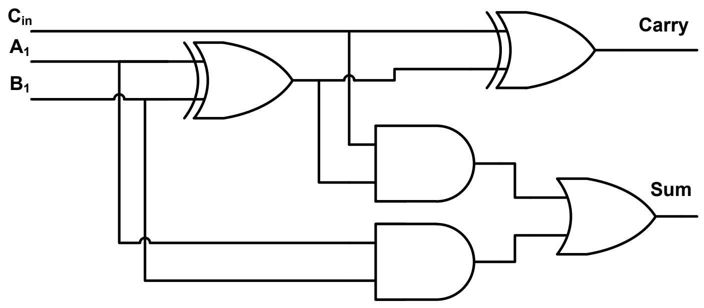


Figure 3.7: Logic diagram of a full adder.

3.5 Protecting Multiple Outputs

In this section, we will discuss the idea of protecting multiple outputs together by using the generalized modular redundancy (GMR) scheme. We will take the full adder as a case study. The full adder consists of two sub-modules: Sum and Carry. Figure 3.7 and Table 3.5 show the full adder and its truth table. By looking at the outputs, we can observe four different output combinations or states, namely: “00”, “01”, “10”, and “11”. It is clearly visible that some states are happening more frequently than others, assuming that all input patterns are equally likely to happen. Probability distribution of states at full adder’s outputs is like this: *P(00) = 1/8*, *P(01) = 3/8*, *P(10) = 3/8*, and *P(11) = 1/8*.

Obviously, there is no one dominant state. However, by protecting more than one state we can apply GMR scheme. Assume that we want to protect the two states with highest probabilities of occurrence (“01” and “10”). Figure 3.8 shows that by replicating each module (Sum and Carry) only one time the requirements mentioned above are satisfied. After replication, states (00, 01, 10, 11) become (0000, 0011,

Cin	A1	B1	Sum	Carry
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 3.1: Truth table of a full adder.

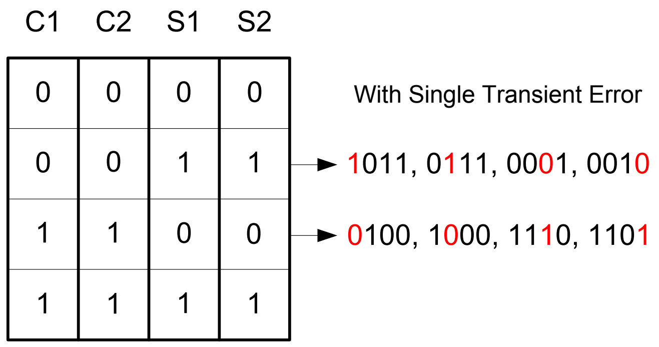


Figure 3.8: Full adder's states after replication (protecting two states).

1100, 1111), respectively. We denote Carry and its replica as C1 and C2. Similarly, S1 and S2 represent Sum and its duplicate.

The equations 3.1 and 3.2, show that the conditions are satisfied, where H is the Hamming distance between two states. As a result, states which will result in from a protected state, when a single transient error hits the logic, are guaranteed to be disjoint from each others and from other unprotected states. The final outputs of the protected version of the full adder can be obtained using Karnaugh-map as shown in Figure 3.9. It is evident that the area of this logic is comparable to the

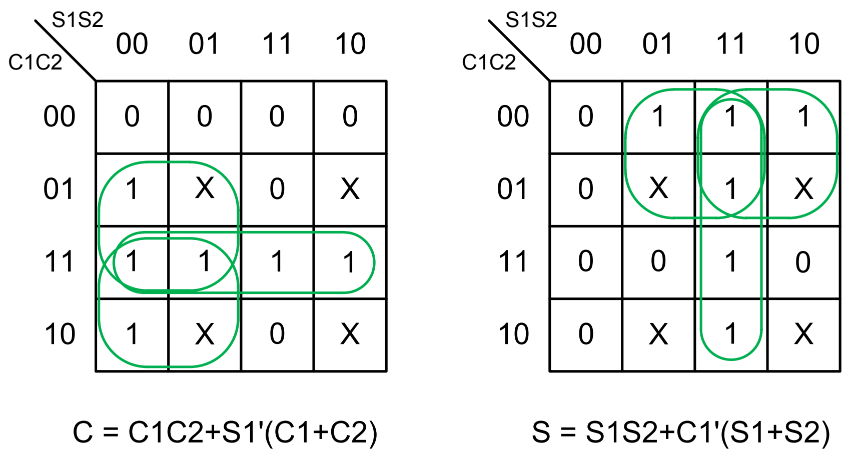


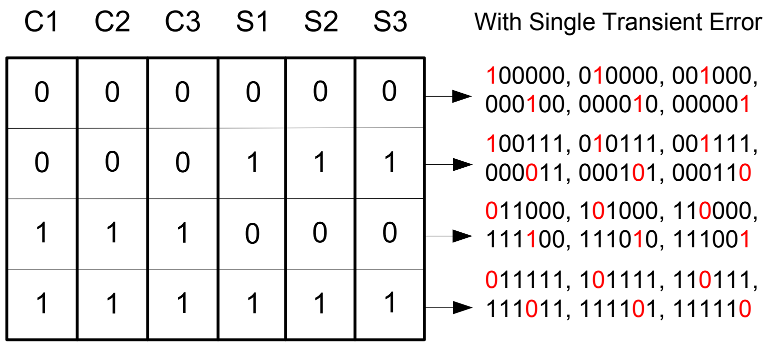
Figure 3.9: Full adder's protected outputs after replication (protecting two states).

area of the majority voter.

$$H(0011, 1100) = 4 > 3. \tag{3.1}$$

$$\begin{aligned} &H(0000, 0011) = 2, H(0000, 1100) = 2, \\ &H(1111, 0011) = 2, H(1111, 1100) = 2. \end{aligned} \tag{3.2}$$

What if we want to apply GMR, generally protecting portion of states, to protect all the states of a full adder? Figure 3.10 shows that in order to protect all states we need to replicate each module two times. This will ensure that states are disjoined. The final protected outputs are also shown in the figure. In this case, protecting all states, we had to triplicate each module; the outputs also are identical to the majority voters used in TMR. Accordingly, we can think of triple modular redundancy (TMR) as a special case of our method, where we protect all states at the output.



C1	C2	C3	S1	S2	S3	With Single Transient Error
0	0	0	0	0	0	100000, 010000, 001000, 000100, 000010, 000001
0	0	0	1	1	1	100111, 010111, 001111, 000011, 000101, 000110
1	1	1	0	0	0	011000, 101000, 110000, 111100, 111010, 111001
1	1	1	1	1	1	011111, 101111, 110111, 111011, 111101, 111110

C = C1C2 + C1C3 + C2C3 S = S1S2 + S1S3 + S2S3

Figure 3.10: Full adder's protected outputs after replication (protecting all states).

A digital comparator serves as an excellent example for protecting multiple output circuits. A 4-bit digital comparator has two inputs A and B each of which is 4-bit long. It has two outputs. The output equals to "00" if and only if the two inputs are equal. It equals to "10" if A is greater than B, and to "01" if B is greater than A. Thus, the probability of having a "00" at the output is: $Prob(00) = 2^4/2^8 = 0.0625$, while the probability of having a "10" or "01" equals to: $Prob(10) = Prob(01) = (1 - Prob(00))/2 = 0.46875$. The protection will be applied to states which are vulnerable to soft errors the most. This includes both states "10" and "01". By protecting these states, more than 93% of soft errors will be masked.

3.6 Why Generalized Modular Redundancy?

We have seen that the dominant value protection concept along with simple voters "AND voter" and "OR voter" are a special case of generalized modular redundancy (GMR) where we protect a single output circuit. Moreover, triple modular redundancy (TMR) can be seen as a special case of this method, where we protect all states at the output no matter how many outputs are protected. Hence, the name generalized modular redundancy is used. What is more important about GMR is the idea of protecting multiple outputs together. To clarify this point, let's go back to our example, the full adder. By looking at each module alone, Sum or Carry, we can see that the probability of having a "0" is equal to the probability of having a "1" for both modules. To protect these modules against transient soft errors, TMR will be used as there is no dominant value. However, if we look at the outputs of the full adder as a block, we can see that there are some dominant states (state "01", and state "10"). Having this in mind, we are able to increase the reliability of the circuit by protecting these dominant states, while maintaining less area overhead. Table 3.6 summarizes these findings.

Circuit	**Num. of Modules**	**Voter**
FA (without protection)	2	-
FA (protecting 2 states)	4	special logic
FA (TMR)	6	majority voter
Eq. comparator (without protection)	1	-
Eq. comparator (protecting 1 state)	2	single gate
Eq. comparator (TMR)	3	majority voter
4-bit comparator (without protection)	2	-
4-bit comparator (protecting 2 states)	4	special logic
4-bit comparator (TMR)	6	majority voter

Table 3.2: Summary of protection methods and their area overhead.

Chapter 4

Using Generalized Modular Redundancy to Enhance Combinational Circuits Reliability

4.1 Introduction

We have seen that generalized modular redundancy can be used to protect states at the output which are highly vulnerable to soft errors. The protection is applied to single or multiple outputs by adding some redundant modules. These redundant modules can be replicas of the original modules or some new customized modules. Correction logic, or voters in some cases, are then needed to obtain the final protected outputs. In this chapter, we will investigate replication against introducing customized redundant modules, complexity of the correction logic or voters, and the protection of single output versus multiple outputs. Based on that, the developed methodology of applying GMR to enhance the reliability of combinational circuits will be presented.

4.2 Module Replication against Customized Redundant Modules

In module replication, a redundant module is essentially a replica of one of the original modules in the circuit. However, newly introduced redundant modules are specially customized modules. In order to protect states at the outputs of combinational circuits, some requirements regarding the Hamming distance between states have to be met. Having this issue in mind, customized redundant modules, in some cases, are considered more flexible than replicated modules.

To explain this point, consider states at the output of a circuit with two outputs. In order to protect states "00", "01", and "10" by module replication, each module has to be replicated two times to satisfy Hamming distance requirements. However, these requirements can be met by adding just three customized modules. Due to the flexibility of customized redundant modules we are able to save some area overhead. Figure 4.1 demonstrates this case, where *M1* and *M2* are original modules; *R1*$\rightarrow$*R4* are replicated modules and *C1*$\rightarrow$*C3* are customized redundant modules.

Area overhead resulting from adding customized modules is most likely to be better or close to the overhead introduced by module replication due to existence of don't care conditions. Nevertheless, adding customized redundant modules requires full description or truth table in order to be able to synthesize these modules properly. Therefore, in this work, customized redundant modules will be used when the truth table has a reasonable size (i.e., number of inputs ≤ 15). Otherwise, module replication will be used when the number of inputs is greater than 15. However, partial truth tables can be used to synthesize customized redundant modules for circuits with large number of inputs, even though this will not guarantee masking of all faults.

For a better understanding of the difference between replicated modules and customized redundant modules, let us take the circuit described in Figure 4.2 which has 6 inputs and 2 outputs. State "01" has a very high probability of occurrence $P(01) = 0.95$. Thus, we will attempt to protect only state "01". In order to meet the Hamming distance requirements at the outputs of the circuit, mentioned earlier in Section 3.3, by using module replication both *M1* and *M2* modules have to be replicated. So, we will end up with two extra modules identical to the modules of the original circuit. However, these requirements can be easily met by adding one customized redundant module, as depicted in Figure 4.3. In this case, we will end up with one extra customized module. Figure 4.4 shows the synthesized version of the circuit after adding the customized redundant module. We can notice that the

M1	R1	R2	M2	R3	R4
0	0	0	0	0	0
0	0	0	1	1	1
1	1	1	0	0	0
1	1	1	1	1	1

(a)

M1	M2	C1	C2	C3
0	0	0	0	-
0	1	1	1	0
1	0	1	1	1
1	1	0	-	-

(b)

Figure 4.1: Protecting states “00”, “01”, and “10” by: a) Module replication. b) Customized redundant modules.

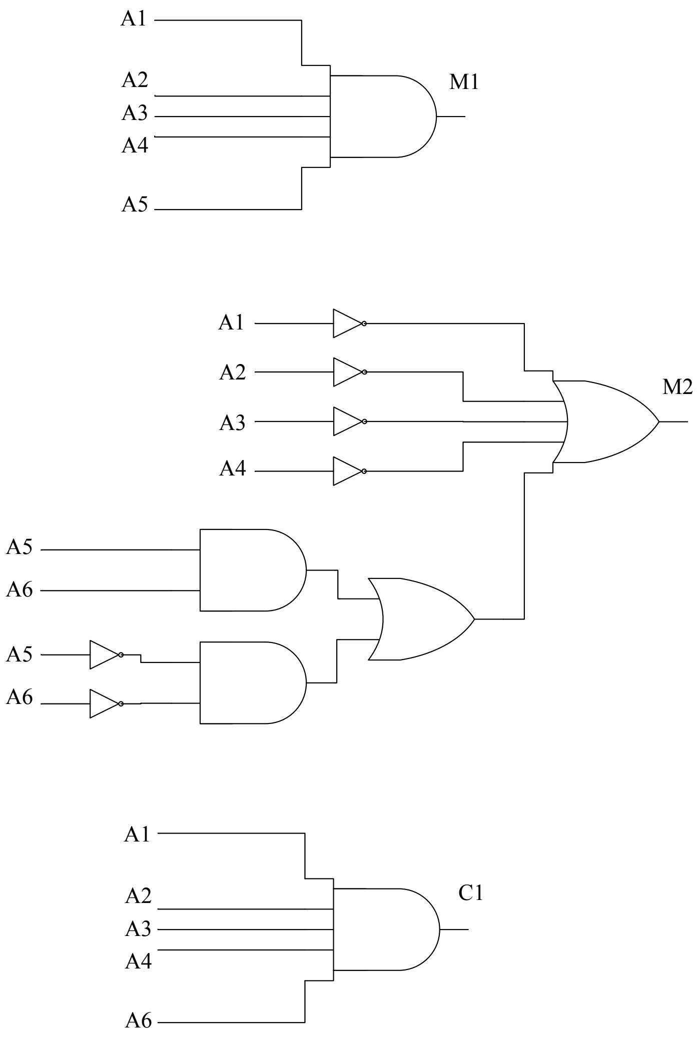
customized redundant module *C1* is smaller in size than original module *M2*, which is one added advantage for this type of modules where customized modules can be smaller in size than the original modules due to existence of don’t cares, as shown in Figure 4.3. After that, the correction logic needed to obtain the protected outputs has to be added to the logic. Figure 4.5 shows the correction logic for modules *M1* and *M2*.

```
.i 6
.o 2
.ilb A1 A2 A3 A4 A5 A6
.ob M1 M2
.p 64
000000 01
000001 01
000010 01
000011 01
.
.
111011 01
111100 01
111101 00
111110 10
111111 11
.e
```

Figure 4.2: Description of example circuit.

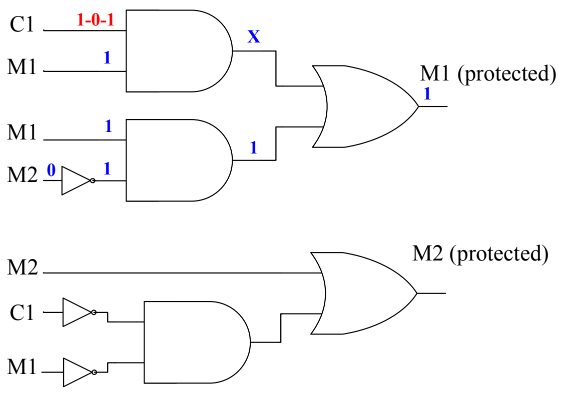
```
.i 6
.o 3
.ilb A1 A2 A3 A4 A5 A6
.ob M1 M2 C1
.p 64
000000 010
000001 010
000010 010
000011 010
.
.
111011 010
111100 010
111101 001
111110 10-
111111 111
.e
```

Figure 4.3: Description of example circuit after adding a customized redundant module.



$$M1 = A1A2A3A4A5$$
$$M2 = A1' + A2' + A3' + A4' + A5A6 + A5'A6'$$
$$C1 = A1A2A3A4A6$$

Figure 4.4: Example circuit after adding a customized redundant module.



M1 (protected) = C1M1 + M1M2'
M2 (protected) = C1'M1' + M2

Figure 4.5: Correction logic for modules *M1* and *M2* in example circuit.

4.3 Complexity of Correction Logic

The idea behind GMR is to protect susceptible states against soft errors while saving some area overhead. Two sources of overhead exist in this method: redundant modules and correction logic or voters to obtain the protected outputs. Having fewer redundant modules is the key to saving area overhead. However, if the correction logic or voters are complex enough, they will cause to override savings in area overhead. For this reason, voters must be minimal in terms of area. We have seen that only one gate (AND or OR) is needed to protect one state at a single output circuit. A reasonable logic size is needed to protect states for circuits with two outputs. 2-out-of-3 majority voter is used when all states at the output are protected. However, experiments show that protecting more than two outputs together demands larger logic to obtain the protected outputs. This may reduce the savings in area achieved by protecting portion of the states at the output. For this reason, we will limit the use of GMR in this work to protect single or pair of outputs. Figure 4.6 shows the majority voter and an example of the correction logic needed in pair protection for the cases of protecting one, two and three states. M_i stands for original modules, R_i stands for replicated modules and C_i stands for customized redundant modules. Two-level logic minimization has been done to these examples by using "espresso" tool. The size of the correction logic when protecting two states is similar to that of the majority voter; it is smaller when only one state is protected. However, the size becomes larger when three states are protected. This is acceptable, especially when the size of the modules is large enough. The size of the correction logic can be reduced further by using multilevel minimization techniques. Table 4.1 reports the size of the correction logic, for all possible scenarios, in number of literals in the sum of product form after using fast extraction "**fx**" and common cube extraction "**gcx**" for multilevel minimization which are implemented in the sequential interactive synthesis (SIS) package [46]. We can see that, in case of protecting one state, the size of the correction logic is smaller than that of the majority voter. The size of the correction logic when protecting two states is similar to the size of the majority voter. However, the size of the correction logic when protecting three states is slightly more than double the size of the majority voter. This introduces. a disadvantage to this technique, unless the size of circuit's modules is large enough where area savings obtained by GMR can compensate the extra overhead introduced by

```
Majority voters:
{P_M1} = M1 R1 + M1 R2 + R1 R2
{P_M2} = M2 R3 + M2 R4 + R3 R4

Correction logic (protecting one state):
{P_M1} = C1 M1 + M1 M2
{P_M2} = C1 M2 + M1 M2

Correction logic (protecting two states):
{P_M1} = C1 C2 M2' + C1 M1 + C2 M1 M2' + C2' M1 M2
{P_M2} = C2 M1' M2 + C2' M1 M2

Correction logic (protecting three states):
{P_M1} = C1 C2 C3 M2' + C1 C2 M1 + C1 C3 M1 M2' + C2 C3 M1 M2' + C3' M1 M2
{P_M2} = C1' C2 C3 M1' + C1' C2 M2 + C1' C3 M1' M2 + C2 C3 M1' M2 + C3' M1 M2
```

Figure 4.6: Majority voter and correction logic for pair protection (with two-level minimization).

the correction logic.

In regard to pair protection, the number of redundant modules is proportional to the number of protected states at the outputs. For example, if we want to protect one state at two outputs circuit or sub-circuit, one customized redundant module has to be added to the logic. If we are to protect two states, two customized redundant modules have to be introduced to meet Hamming distance requirements. Protecting three states will involve the addition of three customized redundant modules. Figures 4.7 and 4.8 depict all possible protection scenarios, where states above the dotted line are the protected states; notations *Mi* and *Ci* are used to designate original modules and customized redundant modules, respectively. It is clearly visible that in all these cases area overhead savings will be achieved. Protecting one state

Scenario	**Lits(sop)**
Protecting state “00”	8
Protecting state “01”	7
Protecting state “10”	7
Protecting state “11”	6
Protecting states “00” & “01”	15
Protecting states “00” & “10”	15
Protecting states “00” & “11”	14
Protecting states “01” & “10”	13
Protecting states “01” & “11”	16
Protecting states “10” & “11”	16
Protecting states “00” & “01” & “10”	28
Protecting states “00” & “01” & “11”	26
Protecting states “00” & “10” & “11”	26
Protecting states “01” & “10” & “11”	26
Majority voter for two outputs	12

Table 4.1: Size of correction logic after multilevel minimization for all possible scenarios of pair protection.

saves three modules and protecting two states saves two modules in comparison to TMR. Even when protecting three states out of four, we are still able to save a module.

4.3.1 Masking of Correction Logic

One more aspect about correction logic is its ability to mask errors originating in the modules. In the first place, correction logic is designed to mask single errors. However, the majority voter for a certain output is independent from any other output and depends only on the module producing this output in addition to its redundant modules. While in pair protection, the correction logic of both outputs depends on all original and redundant modules. So, for a pair of outputs in TMR, the majority voters have a masking probability of $(6+9)/(2^6-1) = 0.238$, where a single error might happen within any of the six modules (2 original + 4 redundant modules) or two errors each occurring in one of the modules belonging to each output (1 original + 2 redundant modules), as they are independent. For a pair of outputs in GMR single-output protection, the correction logic has a masking probability of $(4+4)/(2^4-1) = 0.53$, where a single error might happen within any of the four modules (2 original + 2 redundant modules) or two errors each occurring in one of the modules belonging to each output (1 original + 1 redundant module), as they are independent. For GMR output pair protection, the correction logic when protecting one state has a masking probability of $3/(2^3-1) = 0.429$, where

	M1	M2	C1
00	**0**	**0**	**0**
01	0	1	1
10	1	0	1
11	1	1	-

	M1	M2	C1
01	**0**	**1**	**0**
00	0	0	1
10	1	0	-
11	1	1	1

	M1	M2	C1
10	**1**	**0**	**0**
00	0	0	1
01	0	1	-
11	1	1	1

	M1	M2	C1
11	**1**	**1**	**1**
00	0	0	-
01	0	1	0
10	1	0	0

(a)

	M1	M2	C1	C2
00	**0**	**0**	**0**	**0**
01	**0**	**1**	**1**	**1**
10	1	0	1	-
11	1	1	0	-

	M1	M2	C1	C2
00	**0**	**0**	**0**	**0**
10	**1**	**0**	**1**	**1**
01	0	1	1	-
11	1	1	0	-

	M1	M2	C1	C2
00	**0**	**0**	**0**	**-**
11	**1**	**1**	**1**	**1**
01	0	1	1	0
10	1	0	1	0

	M1	M2	C1	C2
01	**0**	**1**	**0**	**-**
10	**1**	**0**	**1**	**1**
00	0	0	1	0
11	1	1	1	0

	M1	M2	C1	C2
01	**0**	**1**	**0**	**0**
11	**1**	**1**	**1**	**1**
00	0	0	1	-
10	1	0	0	-

	M1	M2	C1	C2
10	**1**	**0**	**0**	**0**
11	**1**	**1**	**1**	**1**
00	0	0	1	-
01	0	1	0	-

(b)

Figure 4.7: a) One customized redundant module to protect one state. b) Two customized redundant modules to protect two states.

	M1	M2	C1	C2	C3
00	**0**	**0**	**0**	**0**	**-**
01	**0**	**1**	**1**	**1**	**0**
10	**1**	**0**	**1**	**1**	**1**
11	1	1	0	-	-

	M1	M2	C1	C2	C3
00	**0**	**0**	**0**	**0**	**-**
01	**0**	**1**	**1**	**1**	**0**
11	**1**	**1**	**0**	**1**	**1**
10	1	0	1	-	-

	M1	M2	C1	C2	C3
01	**0**	**1**	**0**	**0**	**-**
10	**1**	**0**	**0**	**1**	**0**
11	**1**	**1**	**1**	**1**	**1**
00	0	0	1	-	-

	M1	M2	C1	C2	C3
00	**0**	**0**	**0**	**0**	**-**
10	**1**	**0**	**1**	**1**	**1**
11	**1**	**1**	**0**	**1**	**0**
01	0	1	1	-	-

Figure 4.8: Three customized redundant modules to protect three states.

a single error might happen in one of the modules (2 original + 1 redundant). The probability of masking when protecting two states equals to $4/(2^4 - 1) = 0.267$, as there are 4 modules (2 original + 2 redundant). Finally, the probability of masking when protecting three states equals to $5/(2^5 - 1) = 0.161$, as there are 5 modules (2 original + 3 redundant). As a result, this analysis adds more advantage to GMR single-output protection in addition to pair protection when one or two states are protected. On the contrary, the probability of masking for the correction logic is less than that of the majority voter when three states are protected in GMR pair protection.

The probability of masking for the case of GMR has to be multiplied by the probability of state protection. So, the overall probability of masking for TMR is

$1 - P_e + P_e \times P_m$. While, it is $1 - P_e + P_e \times P_s \times P_m$ for GMR. Where, P_e is the probability of having an error in the circuit represented by its modules, P_m is the probability of masking and P_s is the probability of state protection. The probability of having an error in the module of an output differs between TMR and GMR. It should be less in GMR due to the use of customized redundant modules where the area is probably less; thanks to existing don't cares.

Having don't care conditions in customized redundant modules, as shown in Figures 4.7 and 4.8, can add more fault tolerance in the circuit. When there is a don't care in an unprotected state, we don't have to care about the value of the redundant module, as the Hamming distance requirements are already met. So, even if a transient error appeared at this extra module, we can still have a correct result at the output of the correction logic. This also is shown in Figure 4.5. If the don't care is present in a state that is not protected this makes us tolerate errors in the customized module. However, if the don't care is in one of the protected states, then this will allow us to tolerate double errors by design. One error on the customized module that has the don't care and one error on any other module.

4.4 Single Output Versus Pair Protection

As we have restricted the use of GMR in this work to deal with a single or pair of outputs, an investigation has to be made in order to verify the effectiveness of each option. To achieve this, the circuit which is described in Figure 4.2 will be used as an example. This circuit has 6 inputs and 2 outputs. It has a fixed output “01” for all input patterns except three where the states are “00”, “10”, and “11”. We perform two protection procedures. In the first one, single output protection is applied to each output, while in the other the two outputs are protected as a pair. In single output protection we need to add two redundant modules. For pair protection, we protect for two states as they require the same area overhead, in terms of redundant modules, when applying single output protection. The percentage of protected states at the outputs for both cases is set to be the same. Later, correction logic or voters are added to obtain the protected outputs. Figures 4.9 and 4.10 show the resulting circuits after applying single and pair protection. *M1* and *M2* are original modules, *R1* and *R2* are replicated modules, *C1* and *C2* are customized redundant modules. By analyzing the reliability of each method against soft errors, it is evident that single output protection is more advantageous, as shown in Figure 4.11. The size of correction logic, which is significantly larger in pair protection, together with probability of masking play an important role in this regards. As this logic has no protection against soft errors, reliability will be degraded as this logic gets bigger. Also, the reliability of single output protection is better than pair protection when protecting only one state. This is shown in Figure 4.12. These findings are consistent for larger circuits. Thus, we will favor single output protection whenever it is possible.

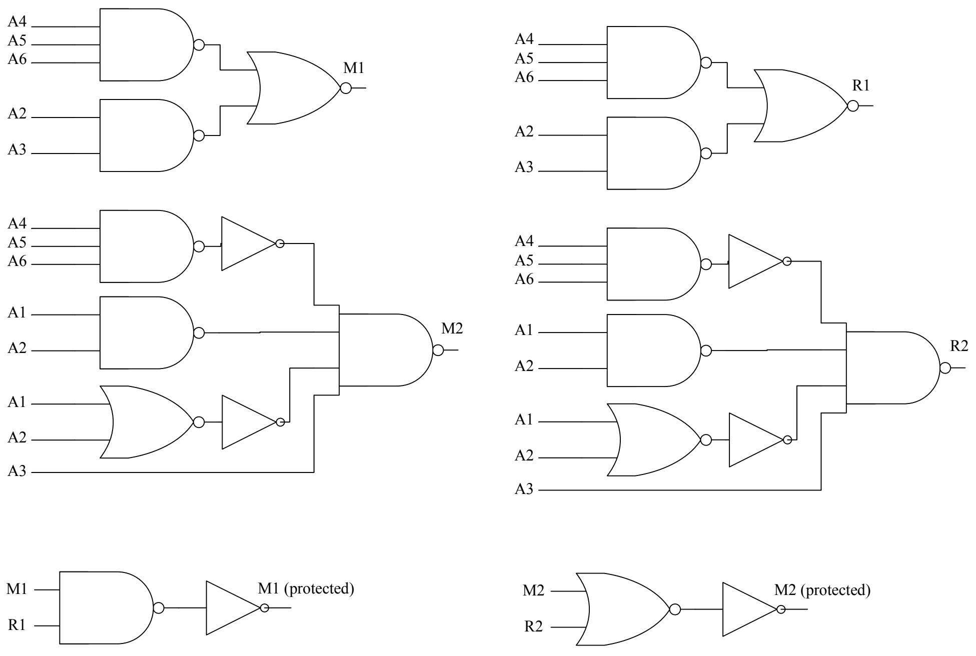


Figure 4.9: Original and replicated modules for single protection of the example circuit along with correction logic.

4.5 Methodology of Applying GMR

From the discussion above, we were able to reach some important conclusions about critical issues for applying the generalized modular redundancy scheme to increase combinational circuits' fault tolerance. These conclusions comprise the following:

- Customized redundant modules will be used when the truth table has a reasonable size (i.e., number of inputs ≤ 15) for both single and pair protection. Otherwise, module replication will be used for circuits with more than 15 inputs.

- Due to the complexity of the correction logic needed to obtain protected out-

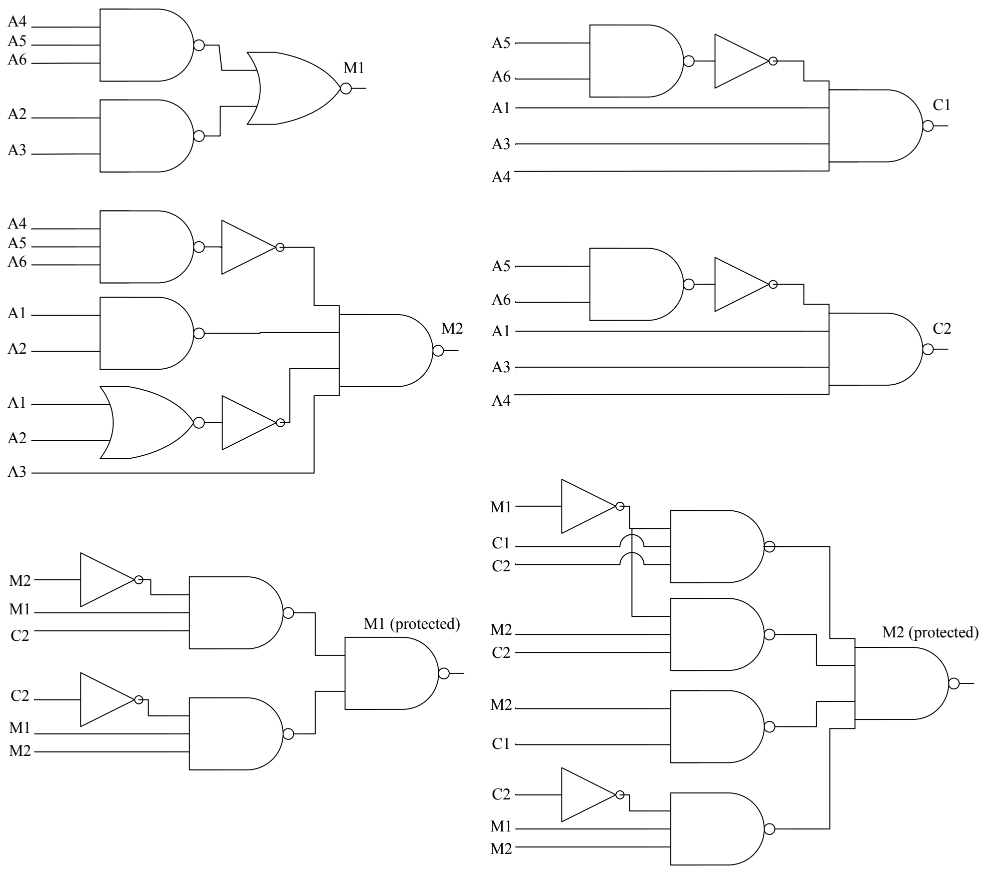


Figure 4.10: Original and customized modules for pair protection of the example circuit along with correction logic.

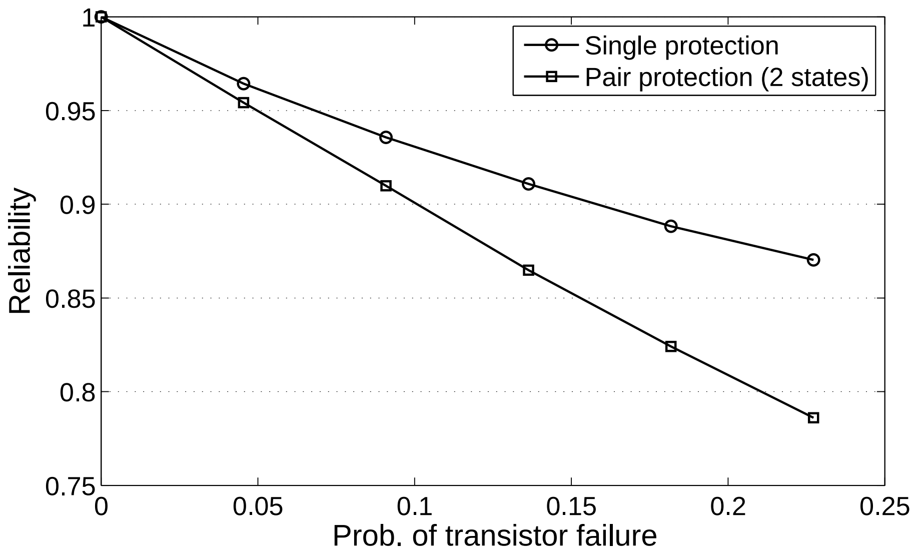


Figure 4.11: Reliability of single protection against pair protection (protecting 2 states).

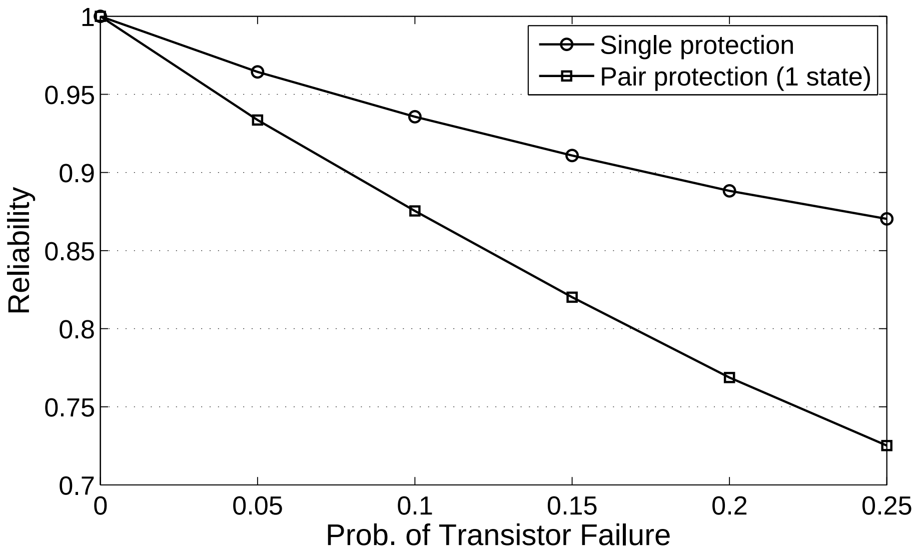


Figure 4.12: Reliability of single protection against pair protection (protecting 1 state).

puts, applying GMR protection will be limited to sub-circuits with one or two outputs.

- We will favor single output protection whenever it is possible. If not, pair protection will be used. Otherwise, triple modular redundancy will be applied.

Based on these conclusions, the methodology depicted in Figure 4.13 will be used to apply GMR for enhancing combinational circuits reliability.

In the initialization step, a protection threshold for identifying dominant states at the outputs, *thr*, has to be specified. For a single output, a state will be considered dominant if the probability of occurrence for this state is greater than the threshold. However, for pair protection, meeting this threshold might require protecting more than one state. States with the highest probabilities of occurrence will be protected such that the sum of their probabilities must exceed protection threshold. A truth table of the circuit has to be provided also. This table will be used to calculate probability of occurrence for states at the outputs. For circuits with large number of inputs, the table will be unacceptably large. Thus, partial truth tables can be used to estimate probabilities of states at the outputs by simulation.

During evaluation, the probability of occurrence for the dominant state $P(D_i)$ at output i is calculated. Then, output i is paired with other available outputs j and the probability of occurrence for the dominant state(s) $P(D_{ij})$ is calculated. As we may protect more than one state, we keep track of the number of protected states

thr	:	Protection threshold for identifying dominant states.
T/PT	:	Truth table/Partial truth table of the circuit.
$P_i(S)$	:	Probability of occurrence of state S at output $\boldsymbol{i}$.
$P_{ij}(S)$	:	Probability of occurrence of state S when pairing outputs $\boldsymbol{i}$ and $\boldsymbol{j}$.
$P(D_i)$	:	Probability of the dominant state at output $\boldsymbol{i}$.
$P(D_{ij})$	:	Probability of the dominant state(s) when pairing outputs $\boldsymbol{i}$ and $\boldsymbol{j}$.
$N_{P_{ij}}$	:	Number of protected states when pairing outputs $\boldsymbol{i}$ and $\boldsymbol{j}$.

```
    Begin
1.  Initialization:
2.  Set protection threshold thr.
3.  Provide T/PT.
4.  Evaluation:
5.  ForEach output i Do
6.      Evaluate probability of occurrence $P_i(0)$, $P_i(1)$.
7.      For all other outputs j.
8.          Evaluate probability of occurrence $P_{ij}(00)$, $P_{ij}(01)$, $P_{ij}(10)$, $P_{ij}(11)$.
9.      Choose best candidate j where thr has been met with minimum $N_{P_{ij}}$.
10.     Decision:
11.     If $P(D_i) > thr$ Then
12.         If $N_{P_{ij}} > 2$ Then
13.             Apply single protection.
14.         Else
15.             If $P(D_i) \geq P(D_{ij})$ Then
16.                 Apply single protection.
17.             Else
18.                 Apply pair protection.
19.             EndIf
20.         EndIf
21.     Else If $N_{P_{ij}} \leq 3$ Then
22.         Apply pair protection.
23.     Else
24.         Apply TMR protection.
25.     EndIf
26. EndFor
    End.
```

Figure 4.13: Methodology of applying GMR for enhancing combinational circuits reliability.

$N_{P_{ij}}$ when pairing output i with output j. After that, we choose the best candidate j for pairing with i such that $P(D_{ij})$ is greater than *thr* with minimum value for $N_{P_{ij}}$.

Next, a decision on the protection scheme has to be made. This includes single protection, pair protection and TMR. The cost of applying single protection equals the cost of applying pair protection when the number of protected states $N_{P_{ij}}$ is two. For this reason, the scheme which provides maximum protection will be used, line 15 in the methodology. If both single protection and pair protection are not applicable, then TMR will be used to protect the output.

4.6 Illustrative Example

In a standard floating point representation a real number is represented using three values: a sign bit, mantissa and exponent; the value of the real number is:

$r = (-1)^{sign} \times 1.mantissa \times 2^{exponent}$

In this representation, the most significant bit (MSB) of the real number (which is always 1) is hidden. To add or subtract two floating point numbers, first their exponents must be aligned by shifting (1.mantissa) of one of the two numbers to the right. Then the shifted/unshifted (1.mantissa) of the two numbers are added or subtracted. Since the result of this addition/subtraction may not be of the form (1.xxx), it is necessary to shift the result to the left until the MSB is 1. This process is called Normalization. In order to find the proper shift amount, a leading zero detector (LZD) is required. The LZD can also detect the cases when the result of addition is all zeros. So, for example a 24-bit leading zero detector circuit will have 24 inputs and 5 outputs. In this case, the state "00000" has the highest probability of occurrence followed by "00001" then "00010" and so on.

Let the outputs of this circuit be: *out0*, *out1*, *out2*, *out3* and *out4* from the left to the right. To enhance the reliability of this circuit by using the generalized modular redundancy scheme, we will use the methodology discussed above. At first, we will set protection threshold, *thr*, to 0.9 and provide the truth table shown in Figure 4.14. The algorithm will start with the evaluation step. The probability of having a "1" at *out0* equals to $P_{out0}(1) = 256/2^{24}$, and the probability of having a "0" at this output equals to $P_{out0}(0) = 1 - P_{out0}(1) = 0.99998$. We can see that $P_{out0}(0) > thr$, hence "0" is the dominant state at *out0* and $P(D_{out0}) = P_{out0}(0)$. After that, output *out0* is paired with other available outputs (*out1*, *out2*, *out3* and *out4*) and the probability of occurrence for the dominant states $P(D_{ij})$ is calculated. The results are as follows: $P(D_{out0\_out1}) = 0.99609$, $P(D_{out0\_out2}) = 0.94116$, $P(D_{out0\_out3}) = 0.79998$ and $P(D_{out0\_out4}) = 0.66665$. We can see that output *out1* is the best candidate for pairing with *out0* such that $P(D_{ij})$ is greater than *thr* with minimum value for $N_{P_{ij}}$, where $P(D_{ij}) = P(D_{out0\_out1})$ with $N_{P_{out0\_out1}} = 1$.

At the decision step we have the following: $P(D_{out0}) > thr$, $N_{P_{out0\_out1}} < 2$ and $P(D_{out0}) > P(D_{out0\_out1})$. As a result, *out0* will be protected as a single output.

The same process will be repeated for *out1* and *out2*. They also will be protected as single outputs.

At this point, we are left out with *out3* and *out4*. The evaluation step for *out3* results in the following: $P_{out3}(1) = 0.2$, $P_{out3}(0) = 1 - P_{out3}(1) = 0.8$ and $P_{out3}(0) < thr$. There is no dominant state at *out3* as we have set the *thr* to 0.9. When pairing *out3* with the only output left *out4*, the resulting dominant state $P(D_{out3\_out4})$ equals to 0.933 with the number of protected states, $N_{P_{out3\_out4}}$, equals to 3. At the decision step we have: $P(D_{out3}) < thr$, $P(D_{out3\_out4}) > thr$ and $N_{P_{out3\_out4}} = 3$. As a result, *out3* will be protected as a pair along with *out4*. By using GMR scheme, we are able to reduce area overhead by 4 modules; *out0*, *out1* and *out2* require one extra module for each as they are protected as single outputs; *out3* and *out4* are protected as pair and they require 3 extra modules as the number of protected states is 3. So, we ended up with 11 modules. However, protecting this circuit by TMR will require $5 \times 3 = 15$ modules. Detailed reliability figures will be discussed later in Chapter 6.

```
.i 24
.o 5
.ilb A23 A22 A21 A20 A19 A18 A17 A16 A15 A14 A13
     A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
.ob out0 out1 out2 out3 out4
.p 25
1----------------------- 00000
01---------------------- 00001
001--------------------- 00010
0001-------------------- 00011
00001------------------- 00100
000001------------------ 00101
0000001----------------- 00110
00000001---------------- 00111
000000001--------------- 01000
0000000001-------------- 01001
00000000001------------- 01010
000000000001------------ 01011
0000000000001----------- 01100
00000000000001---------- 01101
000000000000001--------- 01110
0000000000000001-------- 01111
00000000000000001------- 10000
000000000000000001------ 10001
0000000000000000001----- 10010
00000000000000000001---- 10011
000000000000000000001--- 10100
0000000000000000000001-- 10101
00000000000000000000001- 10110
000000000000000000000001 10111
000000000000000000000000 11000
.e
```

Figure 4.14: Actual file describing the truth table of the leading zero detector circuit.

Chapter 5

Experimental Setup & Framework

5.1 Benchmarks

In this work, LGSynth91[1] benchmarks circuits are used. These contain a set of circuits with various sizes, in terms of size of the logic, and number of inputs and outputs, see Table 5.1.

5.2 Fault Model and Injection Mechanism

In our work, we assume a stuck-open and stuck-short fault models at the transistor level. Faults can be injected at any transistor; stuck-open means that the transistor is stuck at the OFF state, while stuck-short means that it is stuck at the ON

[1]http://www.cbl.ncsu.edu:16080/benchmarks/LGSynth91/

Benchmark	**Number of inputs**	**Number of outputs**
5xp1	7	10
alu4	14	8
apex1	45	43
apex2	39	3
apex3	54	50
apex4	9	19
b12	15	9
clip	9	5
cordic	23	2
duke2	22	29
ex5p	8	63
misex2	25	18
misex3	14	14
sao2	10	4
seq	41	35
table3	14	14
table5	17	15
vg2	25	8
z5xp1	7	10

Table 5.1: Benchmarks circuits.

state. The transistor level is used as soft errors usually alter the state of individual transistors.

In case of a stuck-open fault, the gate of the transistor at which the fault is injected is connected to GND for NMOS transistors and to VDD for PMOS transistors. For stuck-short faults, the gate of the transistor at which the fault is injected is connected to VDD for NMOS transistors and to GND for PMOS transistors. This is shown in Figure 5.1. In each simulation iteration, a single or multiple faults are injected randomly, stuck-open or stuck-short is randomly applied to these faulty transistors.

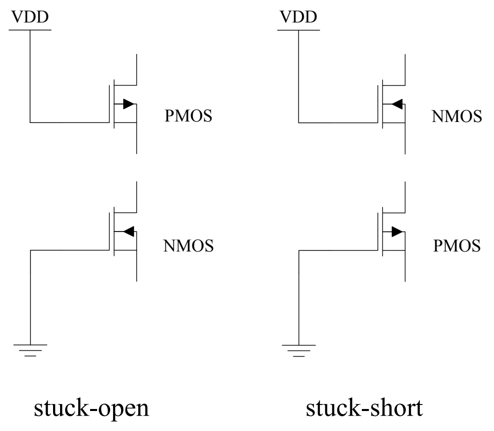


Figure 5.1: Stuck-open and stuck-short fault models.

5.3 Measuring Reliability of Circuits

For evaluating circuit failure probability and reliability, we adopt the simulation-based reliability model used in [47]. We compare circuit reliability based on the generalized modular redundancy scheme with TMR.

To compute the circuit failure probability, F_m, resulting from injecting m defective transistors, we use the following procedure:

1. Set the number of iterations to be performed, I, to 10000 and the number of failed simulations, K, to 0.
2. Simulate the fault-free circuit by applying a random test vector T.
3. Randomly inject m transistor defects.
4. Simulate the faulty circuit by applying the test vector T.
5. If the outputs of the fault-free and faulty circuits are different, increment K by 1.
6. Decrement I by 1 and if I is not 0 goto step 3.
7. Failure Rate $F_m = K/10000$.

Assuming that every transistor has the same defect probability, P, and that defects are randomly and independently distributed, the probability of having a number of m defective transistors in a circuit with N transistors follows the binomial distribution [47] as shown in Eq. 5.1.

$$P(m) = \binom{N}{m} P^m \times (1-P)^{N-m} \tag{5.1}$$

Assuming the number of transistor defects, m, as a random variable and using the circuit failure probability F_m as a failure distribution in m, the probability of circuit failure, F, and circuit reliability, R, are computed as in Eq. 5.2 and Eq. 5.3.

$$F = \sum_{m=0}^{N} F_m \times P(m) \tag{5.2}$$

$$R = 1 - F = 1 - \sum_{m=0}^{N} F_m \times P(m) \tag{5.3}$$

Reliability estimation of combinational circuits can be achieved by measuring their failure rates. Failure rate is the percentage of which a circuit will produce faulty output when a fault is injected in the logic. This way, reliability of a circuit is reciprocally proportional to its failure rate.

5.4 Tools

Many tools are used for different purposes. They comprise:

Espresso: For logic minimization [48].

SIS: For logic minimization and synthesis [46]. The following commands are used for both modules and correction logic:

1. **espresso**.

2. **fx**.
3. **read_library** “Library”[2].
4. **map**.

ModelSim: Used for simulation at the transistor level [49]. Benchmarks are translated from Bench format to Verilog. Then, ModelSim is used to evaluate the reliability of both unprotected and protected versions of the benchmarks.

5.5 Work Flow

We start by providing the protection threshold at the outputs and a truth table or partial truth table for circuits with large number of inputs (more than 16 in this work). The truth table is introduced in espresso *pla* file format. After that, we evaluate probability of occurrence for states at the outputs. Based on this, a decision will be made on how to protect the output under processing. Evaluation and decision steps will be repeated until all primary outputs have been processed. Next, redundant modules followed by voters to obtain the protected outputs will be introduced to the logic. Eventually, we synthesize the logic such that no logic sharing is allowed among different modules. This is achieved by synthesizing each module alone. Then, we combine these synthesized modules together along with the

[2]It includes an *inverter* along with *nand* and *nor* gates with 2, 3 and 4 inputs.

voters. The final protected synthesized version of circuit will be in *bench* format. Figure 5.2 depicts the flow for applying the generalized modular redundancy scheme.

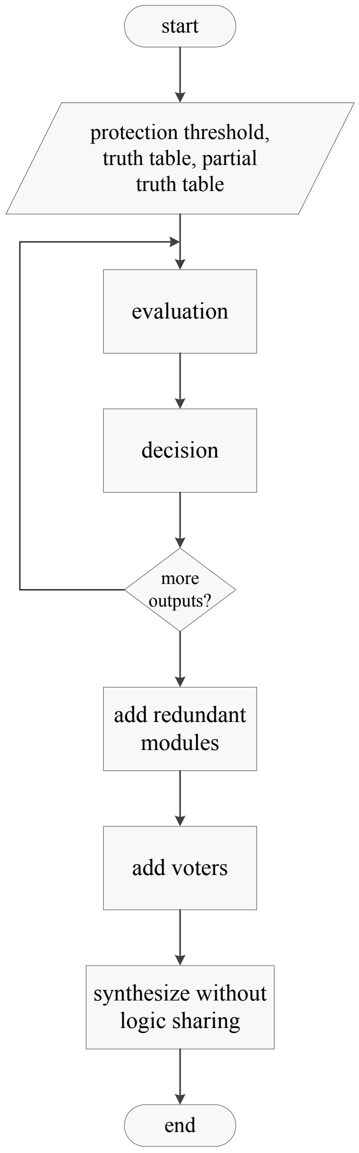

start
protection threshold, truth table, partial truth table
evaluation
decision
more outputs?
add redundant modules
add voters
synthesize without logic sharing
end

Figure 5.2: Work flow for applying the generalized modular redundancy scheme.

Chapter 6

Experimental Results & Discussion

6.1 Analyzing Benchmarks

We have seen the methodology of applying GMR to enhance combinational circuits' reliability in Chapter 4. We have also seen cases where the use of GMR is particularly beneficial (general comparator and equality comparator) in Chapter 3. In order to investigate the applicability of the proposed methodology, it will be applied on the set of benchmarks presented in Chapter 5. We examine different values of protection threshold *thr*. Then, for each value, we evaluate the number of protected modules in the following categories: single output protection, pair protection and triple modular redundancy. We also list the number of protected states for outputs that are protected as pairs. Tables 6.1, 6.2 and 6.3 summarize this evaluation.

Protection threshold *thr* can take values in the range [0, 1]. A value of zero means that no protection is applied. A value of one means that all states at the primary outputs are protected, i.e., TMR will be applied as TMR is the special case of GMR where all states at the primary outputs are protected. As the value of *thr* drops below one, the number of modules protected by TMR will also decrease. These modules will be protected by GMR as singles or pairs. This is clearly visible in "apex4" benchmark where the number of modules protected by TMR has dropped from 10 to 1.

In most of the cases, the number of modules which are protected by TMR is considerably low. In fact, a good number of cases like: "misex2", "misex3", "table3", "table5", "duke2" and "ex5p" have almost no modules protected as TMR. This enables us to exploit the advantage of GMR where decent reliability figures are achieved while saving area overhead. For few benchmarks like "clip" and "5xp1" TMR is used despite the reduction of *thr* down to 0.75. For such cases no or limited area savings can be achieved.

From this analysis, we can see that the proposed methodology of applying GMR to increase fault tolerance of combinational circuits is very encouraging. In the next section, reliability that corresponds to different protection thresholds will be inspected for some benchmarks.

Benchmark	5xp1				alu4				
Protection threshold	0.95	0.9	0.85	0.8 - 0.75	0.95	0.9	0.85	0.8	0.75
No. of outputs	10				8				
Single protection	0	0	0	0	0	0	0	0	0
TMR protection	8	8	8	6	4	4	4	4	0
Pair protection	2	2	2	4	4	4	4	4	8
No. of protected states in each pair	3	3	3	2	3	2	2	2	2
				3	3	3	3	3	3
									3
									3

Benchmark	apex2		apex4				
Protection threshold	0.95 - 0.9 - 0.85 - 0.8	0.75	0.95	0.9	0.85	0.8	0.75
No. of outputs	3		19				
Single protection	2	3	1	1	2	2	2
TMR protection	1	0	10	2	1	1	1
Pair protection	0	0	8	16	16	16	16
No. of protected states in each pair			3	3	3	2	2
			3	3	3	2	2
			3	3	3	2	2
			3	3	3	3	2
				3	3	3	3
				3	3	3	3
				3	3	3	3
				3	3	3	3

Benchmark	b12				clip			cordic	
Protection threshold	0.95	0.9	0.85 - 0.8	0.75	0.95 - 0.9 - 0.85	0.8	0.75	0.95	0.9 - 0.85 - 0.8 - 0.75
No. of outputs	9				5			2	
Single protection	0	0	0	3	0	0	0	0	2
TMR protection	1	1	1	0	5	5	1	0	0
Pair protection	8	8	8	6	0	0	4	2	0
No. of protected states in each pair	2	2	2	2			3	2	
	3	2	2	2			3		
	3	3	2	2					
	3	3	2						

Table 6.1: Analyzing benchmarks at different protection thresholds (1).

Benchmark	duke2				misex2			misex3	
Protection threshold	0.95	0.9	0.85 - 0.8	0.75	0.95	0.9 - 0.85	0.8 - 0.75	0.95	0.9 - 0.75
No. of outputs	29				18			14	
Single protection	12	21	23	25	10	13	15	0	13
TMR protection	1	0	0	0	0	1	1	0	1
Pair protection	16	8	6	4	8	4	2	14	0
No. of protected states in each pair	2	2	2	2	2	2	2	2	
	2	2	2	2	2	2		2	
	2	2	2		2			2	
	2	2			3			2	
	2							3	
	2							3	
	2							3	
	2								

Benchmark	sao2		table3			table5			
Protection threshold	0.95 - 0.9 - 0.85 - 0.8	0.75	0.95	0.9	0.85 - 0.8 - 0.75	0.95	0.9	0.85	0.8 - 0.75
No. of outputs	4		14			15			
Single protection	2	2	1	10	14	2	11	13	15
TMR protection	0	0	1	0	0	1	0	0	0
Pair protection	2	2	12	4	0	12	4	2	0
No. of protected states in each pair	3	2	2	2		2	2	2	
			2	2		2	2		
			2			2			
			2			2			
			2			2			
			3			3			

Table 6.2: Analyzing benchmarks at different protection thresholds (2).

Benchmark	**ex5p**					**vg2**		**Z5xp1**	
Protection threshold	0.95	0.9	0.85	0.8	0.75	0.95 - 0.9 - 0.85	0.8 - 0.75	0.95 - 0.9 - 0.85	0.8 - 0.75
No. of outputs	63					8		10	
Single protection	26	36	51	57	63	4	4	0	1
TMR protection	1	1	0	0	0	4	2	8	7
Pair protection	36	26	12	6	0	0	2	2	2
No. of protected states in each pair	2	2	2	2			2	3	3
	2	2	2	2					
	2	2	2	2					
	2	2	2						
	2	2	2						
	2	2	2						
	2	2							
	2	2							
	2	2							
	2	2							
	2	2							
	2	2							
	2	2							
	2								
	3								
	3								
	3								
	3								

Benchmark	**apex1**					**apex3**			**seq**	
Protection threshold	0.95	0.9	0.85	0.8	0.75	0.95	0.9 - 0.85 - 0.8	0.75	0.95	0.9 - 0.85 - 0.8 - 0.75
No. of outputs	43					50			35	
Single protection	11	11	13	25	31	30	34	38	11	13
TMR protection	6	4	2	0	0	0	0	0	2	2
Pair protection	26	28	28	18	12	20	16	12	22	20
No. of protected states in each pair	2	2	2	2	2	2	2	2	2	2
	2	2	2	2	2	2	2	2	2	2
	2	2	2	2	2	2	2	2	2	2
	2	2	2	2	2	2	2	2	2	2
	2	2	2	2	2	2	2	2	2	2
	2	2	2	2	2	2	2	2	2	2
	2	2	2	2		2	2		2	2
	2	2	2	2		2	2		2	2
	2	2	2	2		2			2	2
	2	2	2			2			2	2
	2	2	2						2	
	2	2	2							
	2	2	2							
		2	2							

Table 6.3: Analyzing benchmarks at different protection thresholds (3).

6.2 Evaluating Reliability at Different Protection Thresholds

We have seen how changing protection threshold, *thr*, can affect protection scheme at the outputs. However, as the value of *thr* goes smaller it affects the overall reliability of the protected version of the circuit. Thus, the value of *thr* has to stay at high values to insure that the reliability of the protected circuits stays at acceptable rate. Figure 6.1 depicts changing in overall reliability of "apex4" benchmark in response to different protection thresholds. It is evident that the circuit with the highest protection threshold (0.95) has the best reliability. At *thr* equals to 0.9, decent reliability figures are achieved accompanied by remarkable overhead savings. By decreasing the value of *thr* we can notice the corresponding reduction in reliability. Table 6.4 shows the reliability of the original circuit "apex4", protected versions at different protection thresholds using GMR, and protected version using TMR along with their area overhead. It is evident that at thresholds higher than 0.8, GMR is able to achieve higher reliability figures than that of TMR along with decent savings in area overhead.

At protection threshold equals to 0.95, 8 outputs are protected as 4 pairs and 1 output is protected as a single output. In each pair of outputs, we protect 3 out of 4 states which requires 3 redundant modules. The total savings in area overhead are 5 modules (4 in pair protection + 1 in single protection). Moreover, the existence of don't cares in the customized redundant modules (which is used in pair protection) increases fault tolerance of protected circuits. If we don't care about the output of a redundant module, this means that, in this particular case, we can mask the error without referring to that redundant module. So, no matter if an error hits that module or not, we still can obtain a correct output. For example, in one of the customized modules within this circuit, more than 68% of the minterms are don't cares. These don't cares can significantly boost the reliability of the protected circuit. In addition to its higher area overhead, TMR protection lacks the advantage of the don't cares within the redundant modules. For these reasons, TMR falls short behind GMR even at *thr* equals to 0.8, where 20% of states at the outputs are not protected. At protection threshold equals to 0.9, we are able to save 4 more modules. When *thr* equals to 0.8, the total of 13 modules are saved as against TMR protection. However, decreasing protection threshold results in the use of pair protection to protect four pair of outputs, where three states are protected, rather than TMR (as shown in Table 6.1). Away form savings in area overhead, this will have a negative effect as the correction logic when protecting three states is larger than the majority voters, and probability of masking for this logic is less. Therefore, due to the sacrifice made when protection threshold is reduced along with previously stated reasons, the reliability at 0.95 is better than that at 0.9 and lower.

Figure 6.2 and Table 6.5 show reliability results for “ex5p” benchmark. By analyzing these results, the same findings from previous example can be extracted.

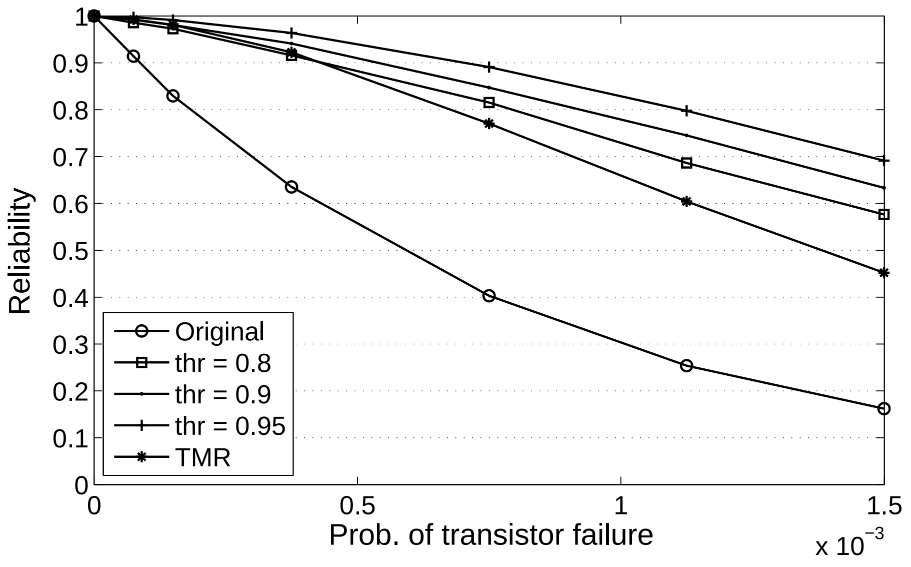


Figure 6.1: Reliability of “apex4” benchmark at different protection thresholds.

Prob. of transistor failure	**Reliability**				
	Original	***thr* = 0.8**	***thr* = 0.9**	***thr* = 0.95**	**TMR**
7.499E-05	0.9141	0.9855	0.9917	0.9973	0.9924
1.500E-04	0.8294	0.9725	0.9807	0.9913	0.9809
3.749E-04	0.6351	0.9157	0.9411	0.9636	0.9225
7.499E-04	0.4029	0.8152	0.847	0.8907	0.7702
1.125E-03	0.2536	0.6861	0.7449	0.7976	0.6041
1.500E-03	0.1621	0.5763	0.6332	0.6912	0.452
Overhead	100.00%	253.18%	259.67%	298.41%	303.70%

Table 6.4: Reliability of “apex4” benchmark at different protection thresholds.

When *thr* equals to 0.95, reliability is at its best; however, overhead savings is not significant. At lower threshold rates, major overhead savings is achieved on the price of sacrificing reliability. This circuit has a large number of outputs (63). As a result, the number of voters and their area overhead will highly affect the reliability of this circuit as the logic of these voters is not protected. Moreover, module sizes in this circuit are considerably small. This will reduce the overall reliability of the protected version of the circuit. Due to the previously discussed reasons, we can see that even after using TMR to protect the circuit, the reliability of the circuit is worse than its reliability without protection. However, the case is better for GMR at high protection thresholds. So, a trade off has to be made between reliability and overhead saving. In the rest of this work, a *thr* equals to 0.9 will be used. Table 6.6 presents number of module savings at *thr* equals to 0.9 when compared to TMR. A maximum overhead reduction of 33% can be achieved when we duplicate all modules instead of triplicating them. This corresponds to the analysis performed earlier in Section 6.1. For some benchmarks with large number of inputs, like “apex1”, “apex3” and “seq”, area overhead savings are limited due to the use of module replication solely. For instance, we are able to save 25 modules instead of 33 in “seq” circuit.

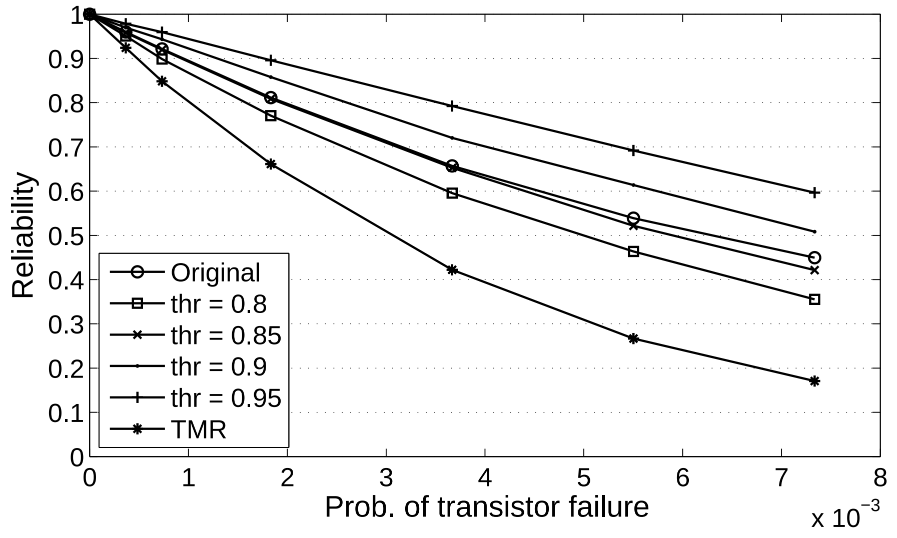


Figure 6.2: Reliability of “ex5p” benchmark at different protection thresholds.

Prob. of transistor failure	**Reliability**					
	Original	***thr* = 0.8**	***thr* = 0.85**	***thr* = 0.9**	***thr* = 0.95**	**TMR**
3.668E-04	0.9597	0.9507	0.958	0.9696	0.9785	0.9237
7.337E-04	0.9215	0.8984	0.9196	0.9433	0.9594	0.8484
1.834E-03	0.8113	0.7704	0.8086	0.8579	0.8956	0.6612
3.668E-03	0.6571	0.5956	0.6529	0.7204	0.7923	0.4222
5.503E-03	0.5389	0.4636	0.5218	0.6138	0.6917	0.2668
7.337E-03	0.4496	0.3554	0.4215	0.5081	0.5964	0.1707
Overhead	100.00%	225.09%	235.73%	262.22%	285.99%	360.09%

Table 6.5: Reliability of “ex5p” benchmark at different protection thresholds.

Benchmark	**Module savings**	**Total num. of modules (TMR)**	**Reduction**
5xp1	1	30	3%
alu4	3	24	13%
apex1	25	129	19%
apex2	2	9	22%
apex3	42	150	28%
apex4	9	57	16%
b12	6	27	22%
clip	0	15	0%
cordic	2	6	33%
duke2	29	87	33%
ex5p	62	189	33%
misex2	17	54	31%
misex3	13	42	31%
sao2	3	12	25%
seq	25	105	24%
table3	14	42	33%
table5	15	45	33%
vg2	4	24	17%
z5xp1	1	30	3%

Table 6.6: Total module savings & reduction percentages compared to TMR at *thr* equals to 0.9.

6.3 Effectiveness of Single Output Protection

We have seen in Section 3.4 that equality comparator serves as an excellent example for protecting single output circuits. The probability of having a "1" at its output is: $Prob(1) = 0.015$, while the probability of having a "0" equals to: $Prob(0) = 0.985$. Protecting the state which is vulnerable to soft errors the most involves adding one extra module, as opposed to TMR wherein we need two redundant modules. Despite protecting only one state at the output, the reliability of the circuit is greater than its reliability when protected using TMR. This is clearly evident in Figure 6.3. Reliability figures and area overhead are shown in Table 6.7.

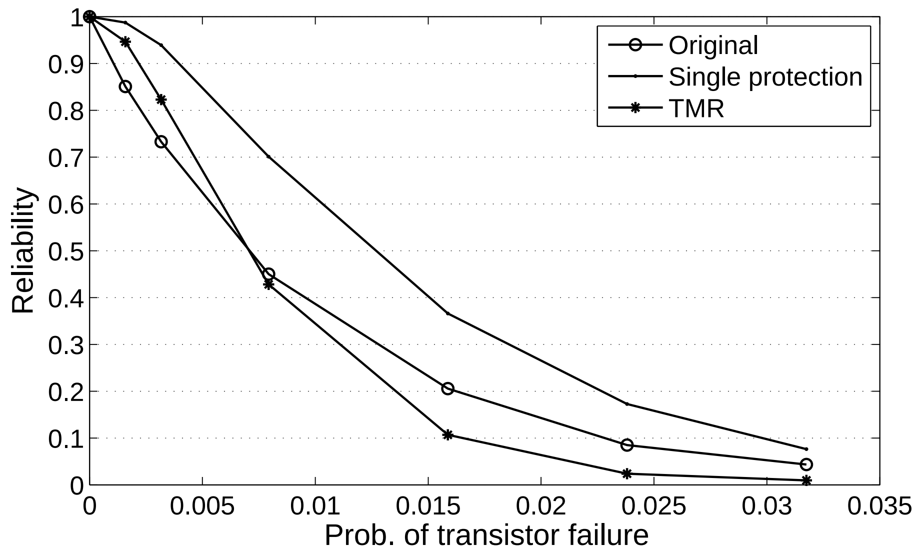


Figure 6.3: Reliability of 6-bit equality comparator with single output protection.

Prob. of transistor failure	Reliability		
	Original	**Single protection**	**TMR**
1.587E-03	0.8508	0.9871	0.9464
3.175E-03	0.7327	0.9391	0.8226
7.937E-03	0.45	0.7007	0.4282
1.587E-02	0.2054	0.3658	0.1068
2.381E-02	0.0849	0.1725	0.0238
3.175E-02	0.0433	0.0762	0.0094
Overhead	100.00%	200.95%	304.13%

Table 6.7: Reliability of 6-bit equality comparator with single output protection.

6.4 Effectiveness of Pair Protection

We have seen in section 3.5 that the digital comparator serves as an excellent example for pair protection. The protection will be applied to states which are vulnerable to soft errors the most, namely states “10” and “01”. By protecting these states, more than 93% of soft errors will be masked. Since only three states appear at the outputs of the comparator, protecting these three states provides full protection for this circuit. In contrast to TMR, where all four possible states are protected using module replication, protecting the three states that might actually happen by adding customized redundant modules can achieve the same protection level provided by TMR which is a 100%. However, due to the remarkable savings in area overhead, the reliability of pair protection is considerably greater than the reliability of TMR, as the failure rate is proportional to the area of the circuit. Figure 6.4 shows the reliability of the digital comparator when protecting two and three states using GMR, and when protecting all states using TMR. It is clear from Table 6.8 that even protecting only 2 out of 3 states is better than TMR thanks to savings in area overhead.

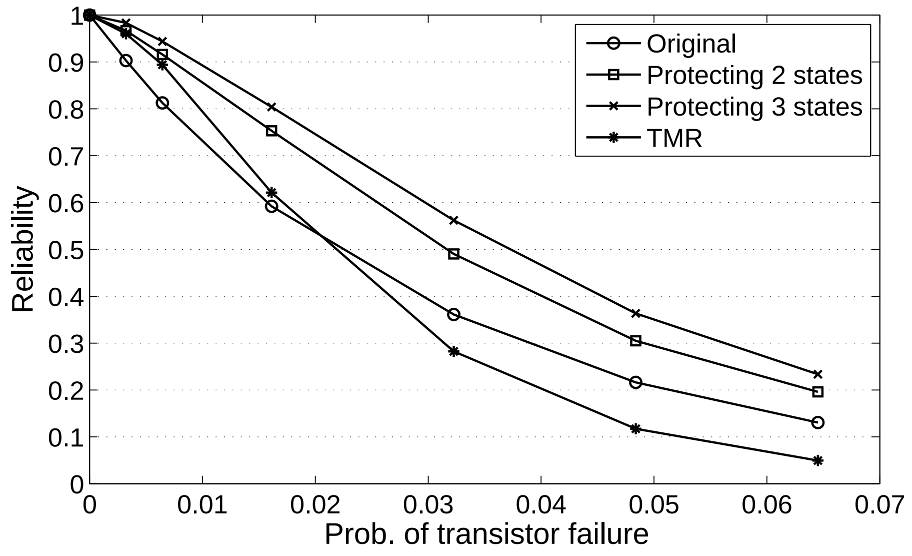


Figure 6.4: Reliability of 4-bit general comparator with pair protection.

Prob. of transistor failure	**Reliability**			
	Original	**Protecting 2 states**	**Protecting 3 states**	**TMR**
3.226E-03	0.9028	0.9666	0.9832	0.96
6.452E-03	0.8124	0.9158	0.9438	0.8939
1.613E-02	0.5918	0.753	0.8037	0.6211
3.226E-02	0.361	0.4903	0.562	0.2822
4.839E-02	0.2161	0.3048	0.3632	0.1172
6.452E-02	0.1305	0.1959	0.2333	0.0493
Overhead	100.00%	200.65%	222.58%	316.77%

Table 6.8: Reliability of 4-bit general comparator with pair protection.

6.5 Effectiveness of GMR Protection

In Section 4.6, we have seen a complete example for applying the GMR protection scheme to enhance the reliability of the leading zero detection circuit. Protecting this circuit using GMR exploits its advantages where both single and pair protection are applied. At protection threshold equals to 0.9, three out of five outputs are protected as single outputs, while the remaining two are protected as pair. A remarkable savings in area overhead is achieved, and reliability of GMR is greater than that of TMR. Figure 6.5 and Table 6.9 demonstrate this.

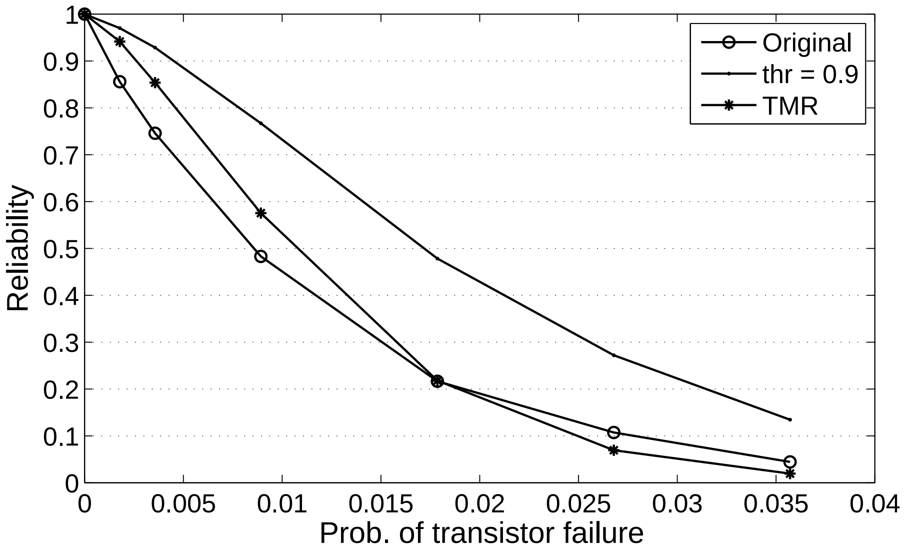


Figure 6.5: Reliability of 24-bit leading zero detector.

Prob. of transistor failure	Reliability		
	Original	***thr* = 0.9**	**TMR**
1.786E-03	0.8558	0.97	0.9413
3.571E-03	0.7457	0.9284	0.8536
8.929E-03	0.483	0.767	0.5752
1.786E-02	0.2164	0.4783	0.2177
2.679E-02	0.107	0.2718	0.0693
3.571E-02	0.0443	0.1345	0.0196
Overhead	100.00%	227.50%	323.21%

Table 6.9: Reliability of 24-bit leading zero detector.

6.6 Reliability of Other Benchmarks at *thr* Equals to 0.9

In this section, we report reliability results of applying GMR protection, with *thr* equals to 0.9, to the set of benchmarks used in this work. We compare these results with the reliability of applying TMR protection. In Table 6.10, we report the reliability results obtained based on the simulation procedure outlined in Section 5.3 for the generalized modular redundancy scheme for several transistor defect probabilities based on stuck-open and stuck-short defects. In Table 6.11, we report the reliability results for the triple modular redundancy scheme. The effectiveness of the generalized modular redundancy scheme is clearly demonstrated by the results as it achieves higher circuit reliability when compared to that of triple modular redundancy. This is in addition to the observation that the GMR scheme requires less area overhead as indicated in the tables. These reliability figures have been achieved

Benchmarks	Generalized Modular Redundancy							
	#Trans.	**0.0001**	**0.0002**	**0.0005**	**0.001**	**0.002**	**0.005**	**Overhead**
alu4	17538	0.999	0.992	0.957	0.866	0.659	0.230	243.85%
apex1	38852	0.991	0.968	0.871	0.704	0.356	0.030	287.62%
apex2	23590	0.992	0.968	0.761	0.426	0.141	0.015	245.22%
apex3	25878	0.992	0.974	0.909	0.757	0.464	0.046	278.08%
apex4	34630	0.988	0.973	0.913	0.779	0.447	0.020	259.67%
b12	1818	0.999	0.997	0.993	0.984	0.960	0.866	394.35%
cordic	14204	0.987	0.955	0.809	0.541	0.199	0.013	200.17%
duke2	8410	0.997	0.993	0.981	0.958	0.898	0.688	240.29%
ex5p	7148	0.991	0.983	0.960	0.924	0.845	0.641	262.22%
misex2	1404	0.997	0.994	0.986	0.972	0.945	0.867	244.60%
misex3	30086	0.994	0.982	0.907	0.729	0.316	0.008	208.87%
sao2	1870	0.999	0.998	0.995	0.986	0.963	0.868	201.51%
seq	47000	0.991	0.974	0.882	0.646	0.282	0.018	241.05%
table3	18884	0.994	0.988	0.962	0.892	0.698	0.188	207.43%
table5	20424	0.994	0.990	0.961	0.882	0.703	0.188	209.82%
vg2	3692	0.999	0.999	0.996	0.980	0.933	0.737	265.61%

Table 6.10: Reliability and area overhead of benchmarks for the GMR scheme with 0.9 protection.

thanks to reduction in overall area overhead due to reduction in total number of required redundant modules and the use of customized redundant modules.

For benchmarks with minimal savings in area overhead and where the size of modules is considerably small, the addition of voters to obtain protected outputs may cancel the savings. Sometimes, the total overhead of using GMR with such benchmarks may exceed the overhead of using TMR especially when pair protection is used which requires larger voters.

“b12” benchmark is an example of such circuits. From Table 6.10 and Table 6.11 we can see that area overhead when using GMR is greater than the overhead when TMR is used. Despite the increase of total overhead, reliability of GMR is still

Benchmarks	**Triple Modular Redundancy**							
	#Trans.	**0.0001**	**0.0002**	**0.0005**	**0.001**	**0.002**	**0.005**	**Overhead**
alu4	21784	0.995	0.981	0.902	0.716	0.387	0.119	302.89%
apex1	41642	0.982	0.943	0.821	0.624	0.271	0.013	308.28%
apex2	28938	0.991	0.938	0.650	0.286	0.055	0.007	300.81%
apex3	29218	0.978	0.934	0.815	0.597	0.278	0.017	313.97%
apex4	40502	0.989	0.971	0.877	0.659	0.302	0.008	303.70%
b12	1614	0.997	0.995	0.986	0.971	0.939	0.832	350.87%
cordic	21340	0.984	0.917	0.650	0.321	0.107	0.013	300.73%
duke2	11254	0.978	0.960	0.916	0.830	0.650	0.276	321.54%
ex5p	9816	0.980	0.959	0.896	0.798	0.636	0.302	360.09%
misex2	2190	0.991	0.982	0.956	0.912	0.827	0.605	381.53%
misex3	43576	0.985	0.953	0.776	0.423	0.077	0.001	302.53%
sao2	2888	0.998	0.996	0.987	0.969	0.920	0.727	311.21%
seq	59404	0.980	0.953	0.766	0.454	0.095	0.000	304.67%
table3	27676	0.990	0.973	0.919	0.784	0.485	0.040	304.00%
table5	29592	0.989	0.974	0.918	0.777	0.472	0.047	304.01%
vg2	4378	0.995	0.989	0.969	0.930	0.858	0.683	314.96%

Table 6.11: Reliability and area overhead of benchmarks for the TMR scheme.

better than that of TMR. This can be attributed to the existence of don't cares in the customized redundant modules.

The only weak spot of a protected circuit is the voter which is added to obtain the protected outputs. As the size of this vulnerable part increases, reliability will degrade due to errors in this part of the circuit which is not protected. So, it is favorable to have voters which are as small as possible; like the voter in single output protection which is only one gate.

The existence of don't cares in the customized redundant modules increases fault tolerance of protected circuits. If we don't care about the output of a redundant module, this means that, in this particular case, we can mask the error without referring to that redundant module. So, no matter if an error hits that module or not, we still can obtain a correct output.

6.7 Protecting Voters

In order to increase the fault tolerance of the error-prone voters different techniques have been proposed. A cascade NMR or TMR scheme is offered using redundant voters to reduce the probability of circuit failure in the voter [42] [50]. The TMR process can be repeated by combining three of the TMR units with another majority voter to form a second-order TMR unit with even higher reliability. Another technique that adds redundancy at the transistor level and provides built-in immunity to stuck-open, stuck-short and bridges defects has also been proposed by El-Maleh et al. [6]. This technique is based on replacing each transistor by N^2-transistor structure ($N \geq 2$) that guarantees defect tolerance of all $N-1$ defects. It provides significantly less circuit failure probability and higher reliability than other techniques based on gate level (quadded logic) and unit level (TMR).

To investigate the effect of protecting voters on the overall reliability of the circuit the second technique will be used [6]. Figure 6.6 shows the advantage of protecting voters in both cases where GMR and TMR were used for a 4-bit general comparator. We can clearly notice the benefit of protecting these voters and how they affect the overall reliability of the circuit, see Table 6.12. The overall area overhead of applying

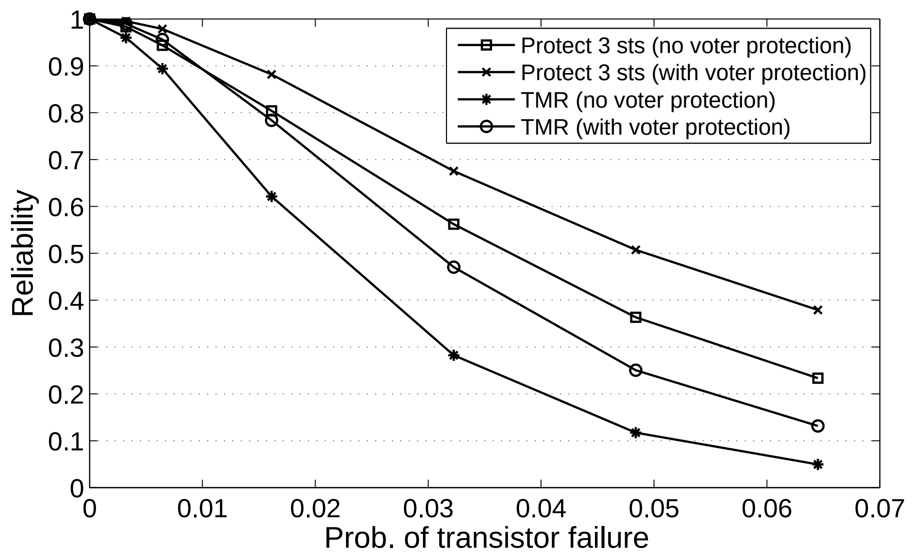


Figure 6.6: Reliability of 4-bit general comparator with/without voter protection.

GMR and protecting voters for a digital comparator equals to the area overhead of applying TMR to that circuit. However, there is a huge improvement in reliability between these two cases. So, by utilizing the savings in area overhead achieved by GMR in protecting the voters, we can make the most of the generalized modular redundancy in terms of both reliability and area overhead.

Figure 6.7 shows reliability of "ex5p" benchmark before and after protecting voters in both cases where GMR and TMR were used. We have seen earlier, in Section 6.2, that this circuit has a large number of outputs (63) and its module sizes are considerably small. As a result, the number of voters and their area overhead will highly affect the reliability of this circuit as the logic of these voters is not protected. Even after using TMR to protect the circuit, the reliability became worse than its

Prob. of transistor failure	**Reliability**			
	Protect 3 states (no voter prot.)	**Protect 3 states (voter prot.)**	**TMR (no voter prot.)**	**TMR (voter prot.)**
3.226E-03	0.9832	0.9951	0.96	0.9897
6.452E-03	0.9438	0.9785	0.8939	0.956
1.613E-02	0.8037	0.8815	0.6211	0.7837
3.226E-02	0.562	0.6753	0.2822	0.470
4 4.839E-02	0.3632	0.5072	0.1172	0.2504
6.452E-02	0.2333	0.3789	0.0493	0.1313
Overhead	222.58%	316.77%	317.42%	367.10%

Table 6.12: Reliability of 4-bit general comparator with/without voter protection.

reliability without protection. However, we can notice the improvement in reliability for TMR after protecting the correction logic. Despite the fact that reliability has improved dramatically, area overhead has also increased to a great extent. This dramatic increase in reliability for TMR can be attributed to the following reasons: (1) large number of voters which corresponds to the large number of outputs in the circuit, (2) relatively large size of majority voters, (3) lower masking probability as opposed to the correction logic for pair protection when only two states are protected for all pairs. When *thr* equals to 0.9, the increase in reliability when protecting the correction logic is not as much as that of TMR. This is due to smaller size and better masking probability of the correction logic as outputs are protected as singles or pairs with 2 states to protect. From Table 6.13, we can see that the overall area overhead of applying GMR and protecting correction logic for the "ex5p" benchmark equals to the area overhead of applying TMR. However, there is a substantial improvement in reliability when the GMR scheme is used.

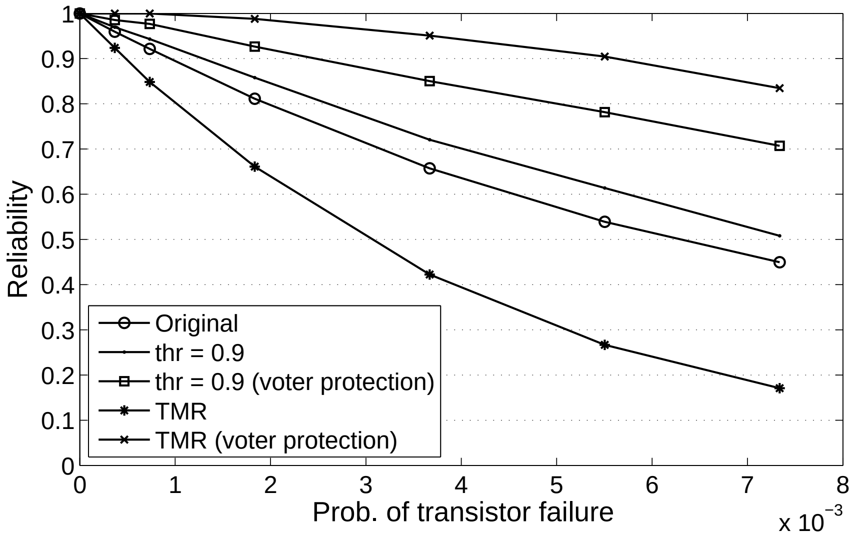


Figure 6.7: Reliability of "ex5p" benchmark with/without voter protection.

Prob. of transistor failure	Reliability			
	***thr* = 0.9**	***thr* = 0.9 (voter protection)**	**TMR**	**TMR (voter protection)**
3.668E-04	0.9696	0.9854	0.9237	0.9999
7.337E-04	0.9433	0.9767	0.8484	0.9997
1.834E-03	0.8579	0.9267	0.6612	0.9880
3.668E-03	0.7204	0.8504	0.4222	0.9510
5.503E-03	0.6138	0.7814	0.2668	0.9047
7.337E-03	0.5081	0.7070	0.1707	0.8347
Overhead	262.22%	365.88%	360.09%	540.35%

Table 6.13: Reliability of "ex5p" benchmark with/without voter protection.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

In this work, a generalized modular redundancy scheme to enhance the reliability of combinational logic circuits against soft errors has been proposed. It is based on probability of occurrence for states at the outputs of these circuits. An investigation on different aspects regarding the application of the generalized modular redundancy scheme has been done. This includes types of redundant modules, complexity of voters and single versus multiple outputs protection.

Furthermore, a methodology for applying the generalized modular redundancy scheme to increase the reliability of combinational logic circuits has been developed. Reliability analysis for various benchmark circuits shows that the proposed methodology can achieve reliability figures higher than that of triple modular redundancy.

Generally, remarkable overhead savings are also accomplished in addition to that superior reliability. Moreover, reliability of the correction logic can be increased by utilizing the attained overhead savings. This way, the overall reliability of the circuit can be further promoted to higher levels while maintaining low area overhead.

7.2 Future Work

As a future work, as investigation regarding the incorporation of the following techniques could be done:

- Incorporation of synthesis techniques which target maximizing the masking property in the logic. This can be applied to synthesize individual modules in a circuit. By maximizing the masking property in the logic, we are increasing the reliability of the modules themselves against soft errors. After that, the generalized modular redundancy scheme can be applied to take care of non-maskable errors.
- Synthesis of customized redundant modules based on partial truth tables for circuits with large number of inputs, even though this will not guarantee masking of all faults.
- Combining the generalized modular redundancy scheme with cluster sharing reduction.

Bibliography

[1] T. Calin, M. Nicolaidis, and R. Velazco. Upset hardened memory design for sub-micron CMOS technology. *IEEE Transactions on Nuclear Science,*, 43(6):2874 –2878, Dec 1996.

[2] Sheng Lin, Yong-Bin 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,*, 19(5):900 –904, May 2011.

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

[4] Ayed Saad Al-Qahtani. Fault tolerance tchniques for sequential circuits: A design level approach. Master's thesis, King Fahd University of Petroleum & Minerals, June 2010.

[5] Y. Dotan, N. Levison, and D. Lilja. Fault tolerance for nanotechnology devices at the bit and module levels with history index of correct computation. *Computers Digital Techniques, IET*, 5(4):221 –230, July 2011.

[6] A.H. El-Maleh, B.M. Al-Hashimi, A. Melouki, and F. Khan. Defect-tolerant N^2-transistor structure for reliable nanoelectronic designs. *Computers Digital Techniques, IET*, 3(6):570 –580, November 2009.

[7] Hammerstrom-D. Colwell B. Bourianoff G. Zhirnov V. Carruthers, J. Computer architectures for nanoscale devices. ITRS 2004. http://public.itrs.net.

[8] James R. Heath, Philip J. Kuekes, Gregory S. Snider, and R. Stanley Williams. A defect-tolerant computer architecture: Opportunities for nanotechnology. *Science*, 280(5370):1716–1721, 1998.

[9] N. Cohen, T.S. 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 Technical Digest.*, pages 315 –318, 1999.

[10] M. Nicolaidis. Design techniques for soft-error mitigation. In *IEEE International Conference on IC Design and Technology (ICICDT)*, pages 208 –214, June 2010.

[11] J. von Neumann. Probabilistic logics and synthesis of reliable organisms from unreliable components. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 43–98. Princeton University Press, 1956.

[12] T. Anderson and B. Randell. *Computing systems reliability.* Cambridge University Press, 1979.

[13] Glenn R. Case. Analysis of actual fault mechanisms in CMOS logic gates. In *Proceedings of the 13th Design Automation Conference*, DAC '76, pages 265–270, New York, NY, USA, 1976. ACM.

[14] M.L. Bushnell and V.D. Agrawal. *Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits.* Frontiers in electronic testing. Kluwer Academic, 2000.

[15] Algirdas Avižienis. Design of fault-tolerant computers. In *Proceedings of the November 14-16, 1967, fall joint computer conference*, AFIPS '67 (Fall), pages 733–743, New York, NY, USA, 1967. ACM.

[16] H. Konoura, Y. Mitsuyama, M. Hashimoto, and T. Onoye. Implications of reliability enhancement achieved by fault avoidance on dynamically reconfigurable architectures. In *International Conference on Field Programmable Logic and Applications (FPL)*, pages 189 –194, Sept. 2011.

[17] P.K. Lala. *Self-checking and fault-tolerant digital design.* The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann, 2001.

[18] II Lantz, L. Soft errors induced by alpha particles. *IEEE Transactions on Reliability,*, 45(2):174 –179, Jun 1996.

[19] P. Liden, P. Dahlgren, R. Johansson, and J. Karlsson. On latching probability of particle induced transients in combinational networks. In *Digest of Papers., Twenty-Fourth International Symposium on Fault-Tolerant Computing. FTCS-24.*, pages 340 –349, Jun 1994.

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

[21] T. Karnik and P. Hazucha. Characterization of soft errors caused by single event upsets in CMOS processes. *IEEE Transactions on Dependable and Secure Computing,*, 1(2):128 – 143, April-June 2004.

[22] S. Mitra, T. Karnik, N. Seifert, and Ming Zhang. Logic soft errors in sub-65nm technologies design and CAD challenges. In *Proceedings 42nd Design Automation Conference.*, pages 2 – 4, June 2005.

[23] S.S. Mukherjee, J. Emer, and S.K. Reinhardt. The soft error problem: an architectural perspective. In *11th International Symposium on High-Performance Computer Architecture. HPCA-11.*, pages 243 – 247, Feb. 2005.

[24] M. Gössel and S. Graf. *Error detection circuits.* McGraw-Hill, 1993.

[25] M. Nicolaidis and Y. Zorian. *On-line testing for VLSI a compendium of approaches*, pages 7–20. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[26] Daniel P. Siewiorek and Robert S. Swarz. *Reliable Computer Systems (3rd ed.): Design and Evaluation.* A. K. Peters, Ltd., Natick, MA, USA, 1998.

[27] J.G. Tryon. *Quadded Logic, Redundancy Techniques for Computing Systems.* Spartan Books, Washington, 1962.

[28] W. H. Pierce. *Failure-Tolerant Computer Design.* Academic Press, New York, 1965.

[29] D.C. Bossen. CMOS soft errors and server design. In *Workshop on Radiation Induced Soft Errors*. IEEE Pres, 2002.

[30] M. Diaz, J.C. Geffroy, and M. Courvoisier. On-set realization of fail-safe sequential machines. *IEEE Transactions on Computers,*, C-23(2):133 – 138, Feb. 1974.

[31] M. Nicolaidis. *Soft Errors in Modern Electronic Systems*. Frontiers in Electronic Testing. Springer, 2010.

[32] Xilinx. http://www.xilinx.com/ise/optional_prod/tmrtool.htm.

[33] Algirdas Avizienis. Arithmetic algorithms for error-coded operands. *IEEE Transactions on Computers,*, C-22(6):567 –572, June 1973.

[34] W. Stallings. *Data And Computer Communications*. Alternative eText Formats Series. Pearson/Prentice Hall, 2007.

[35] M. Nicolaidis. Time redundancy based soft-error tolerance to rescue nanometer technologies. In *Proceedings. 17th IEEE VLSI Test Symposium.*, pages 86–94, 1999.

[36] L. Anghel and M. Nicolaidis. Cost reduction and evaluation of a temporary faults detecting technique. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition.*, pages 591–598, 2000.

[37] S. Krishnaswamy, S.M. Plaza, I.L. Markov, and J.P. Hayes. Enhancing design robustness with reliability-aware resynthesis and logic simulation. In *IEEE/ACM International Conference on Computer-Aided Design. ICCAD 2007.*, pages 149 –154, Nov. 2007.

[38] Tian Ban and Lirida Naviner. Progressive module redundancy for fault-tolerant designs in nanoelectronics. *Microelectronics Reliability*, 51(9-11):1489 – 1492,

2011. Proceedings of the 22th European Symposium on the Reliability of Electron Devices, Failure Physics and Analysis.

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

[40] Xiaoxuan She and P.K. Samudrala. Selective triple modular redundancy for single event upset (SEU) mitigation. In *NASA/ESA Conference on Adaptive Hardware and Systems. AHS 2009.*, pages 344 –350, Aug. 2009.

[41] Y. Dotan, N. Levison, R. Avidan, and D.J. Lilja. History index of correct computation for fault-tolerant nano-computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,*, 17(7):943 –952, July 2009.

[42] S.K. Shukla and R.I. Bahar. *Nano, quantum and molecular computing: implications to high level design and validation.* Solid Mechanics and Its Applications Series. Kluwer Academic Publishers, 2004.

[43] A. Klein Osowski, 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,*, 5(5):575 –586, Sept. 2006.

[44] A. Zukoski, M.R. Choudhury, and K. Mohanram. Reliability-driven don't care assignment for logic synthesis. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1 –6, March 2011.

[45] Khaled A.K. Daud. Synthesis of soft error tolerant combinational circuits. Master's thesis, King Fahd University of Petroleum & Minerals, December 2011.

[46] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. S. Vincentelli. SIS: A System for Sequential Circuit Synthesis. *Electronics Research Laboratory Memorandum*, (UCB/ERL M92/41), May 1992.

[47] Yan Qi Pieter Jonker Jie Han, Jianbo Gao and Jos A.B. Fortes. Toward hardware-redundant, fault-tolerant logic for nanoelectronics. *IEEE Design and Test of Computers*, pages 328–339, July-August 2005.

[48] Robert King Brayton, Alberto L. Sangiovanni-Vincentelli, Curtis T. McMullen, and Gary D. Hachtel. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, Norwell, MA, USA, 1984.

[49] ModelSim. http://www.model.com/.

[50] K Nikolic, A Sadek, and M Forshaw. Fault-tolerant techniques for nanocomputers. *Nanotechnology*, 13(3):357, 2002.

Vitae

- Feras Chikh Oughali
- Born in Damascus, Syria, September 9^{th} 1986.
- Received B.S. degree in Computer Engineering from Damascus University, Damascus, Syria in 2009.
- Received M.S. degree in Computer Engineering from KFUPM, Dhahran, Saudi Arabia in 2012.
- Publications:
  - Sadiq M. Sait, Feras Chikh Oughali, Abdalrahman Arafeh, FSM State-Encoding for Area and Power Minimization Using Simulated Evolution Algorithm, Journal of Applied Research and Technology, December 2012.
  - Abdalrahman Arafeh, Feras Chikh Oughali, Tarek Sheltami, Ashraf Mahmoud, A Contention Free Multi-Channel MAC Protocol With Improved Negotiation Efficiency for Wireless Adhoc Networks, Proceedings of the International Conference on Ambient Systems, Networks and Technologies, Paris, France, 2010.

- Contact Information:
  - Current Address: Dhahran, Saudi Arabia.
  - Permanent Address: Damascus, Syria.
  - Mobile: +966-569077842
  - Email: oughali.feras@gmail.com