UNIVERSITY OF SOUTHAMPTON

Fault Tolerance & Error Monitoring Techniques for Cost Constrained Systems

by

Mauricio Daniel Gutierrez Alcala

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Physical Sciences and Engineering
School of Electronics and Computer Science

September 2017

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Mauricio Daniel Gutierrez Alcala

With technology scaling, the reliability of circuits is becoming a growing concern. The appearance of logic errors in-the-field caused by faults escaping manufacturing testing, single-event upsets, aging, or process variations is increasing. Traditional techniques for online testing and circuit protection often require a high design effort or result in high area overhead and power consumption and are unsuitable for low cost systems. This thesis presents three original contributions in the form of low cost techniques for online error detection and protection in cost constrained systems. The first contribution consists on low cost fault tolerance design technique, that protects the most susceptible workload on the most susceptible logic cones of a circuit, by targeting both timing-independent and timing-dependent errors. The susceptible workload is protected by a partial Triple Modular Redundancy (TMR) scheme. Protecting the 32 most susceptible patterns, an average error coverage improvement of 63.5% and 58.2% against errors induced by stuck-at and transition faults is achieved, respectively, compared an unranked pattern selection and protection. Additionally, this technique produces an average error coverage improvement of 163% and 96% against temporary erroneous output transition and errors induced by bit-flips, respectively. These error coverage improvements incur in an area/power cost in the range of 18.0-54.2%, a 145.8-182.0% reduction compared to TMR. The second contribution proposes a low cost probabilistic online error monitoring technique that produces an alarm signal when systematic erroneous behaviour has occurred over a pre-defined time interval. To detect systematic erroneous behaviour, the collected data is compared on-chip against the signature of error-free behaviour. Results demonstrate on the largest circuits, an average error coverage of 84.4% and 73.1% of errors induced by bit-flips and stuck-at faults, respectively, with an average area cost of 1.66%. The final contribution consists of a circuit approximation technique that can be used for low cost non-intrusive fault tolerance and concurrent error detection, based on finding functionality at the logic level that behaves similarly to single logic gates or constant values. An algorithm is proposed to select the input subsets to approximate. Results show an average coverage of 33.59% of all the input space with an average 7.43% area cost. Using this approximate circuits in a reduced TMR scheme results in significant area cost reductions compared to existing techniques.

Contents

List of Figures

List of Tables

Nomenclature

IC	Integrated Circuits
CMOS	Complementary Metal-Oxide-Semiconductor
CED	Concurrent Error Detection
VDSM	Very Deep Sub-Micron
TMR	Triple Modular Redundancy
IoT	Internet of Things
α	Switching activity
C_L	Load Capacitance
V_{dd}	Supply Voltage
f_{clk}	Operating Frequency
$P_{dynamic}$	Dynamic Power
$P_{leakage}$	Leakage Power
T_{pd}	Transistor Propagation Delay
V_{th}	Threshold Voltage
I_{ds}	Sub-threshold Leakage Current
PV	Process Variation
L_{eff}	Effective Channel Length
μ	Carrier Mobility
NBTI	Negative Bias Temperature Instability
HCI	Hot Carrier Injection
PG	Power Gating
CG	Clock Gating
DVFS	Dynamic Voltage and Frequency Scaling
ATPG	Automatic Test Pattern Generation
SEU	Single-Event Upset
CED	Concurrent Error Detection
BIST	Built-in Self-Test
DWC	Duplications With Comparison
EDC	Error Detecting Codes
TPG	Test Pattern Generator
CUT	Circuit Under Test
ORA	Output Response Analyser

ECC Error Correcting Code

SFT Selective Fault Tolerance

OD Output deviations

PSFT Probabilistic Selective Fault Tolerance

EC Error Coverage

FC Fault Coverage

rp Random Pattern Set

pp Probabilistic Pattern Set

TID Timing-Independent Deviation

TDD Timing-Dependent Deviation

ibSSA Induced by Single Stuck-At

ibBF Induced by Bit-Flip

ibTran Induced by Transition

ibEOT Induced by Temporary Erroneous Output Transitions

SEB Systematic Erroneous Behaviour

EDL Error Detection Latency

M_{sp} Mean Signal Probability

w Signature Window

W_{max} Maximum Signature Window Bound

W_{min} Minimum Signature Window Bound

ALC Approximate Logic Circuit

SER Soft Error Rate

IOB Input-Output Bypass

LC Logic Constant

IC Input Coverage

Acknowledgements

I want to thank first and foremost my mother, whose unconditional support gives me an unbreakable foundation on the important matters. I would like to thank my siblings for their encouragement and support, always on point. I want to thank my supervisors, Bashir Al-Hashimi for the opportunity to pursue a PhD and Tom Kazmierski for providing assistance and helpful commentary throughout. My deep gratitude to examiners Jeff Reeve and Alex Yakovlev for their feedback. I would like to thank my sponsor the National Council of Science and Technology (CONACYT) in Mexico and to the mexican people for their trust and support. I want to express my gratitude to Dr. Daniele Rossi for his valuable insight and to Dr. Vasileios Tenentes for his continued feedback and significant input. I want to all friends and special people that have been with me through the road that is the PhD. My total gratitude for these wonderful experiences.

To the loving memory of my father, whose wisdom guides my present and my future.

Chapter 1

Introduction

As with every piece of machinery, integrated circuits (ICs) are prone to failure. With technology scaling, transistor sizes are reduced to open the way for increased functionality with reduced overall power dissipation, device dimensions and manufacturing costs. However, despite those advantages, the reliability of ICs has been affected [1]. The rising probability of circuit failure caused by increasing device complexity (i.e. transistor density) [2] and the errors caused by increased delay due to temperature rise in CMOS circuits, were well-known reliability concerns in the past. However until recently, achieving a high level of performance was the priority in IC design. With the introduction of very deep submicron (VDSM) CMOS technologies (sub-100nm feature size), the impact on circuit reliability caused by process variation along with other sources of signal instability has been steadily increasing [3]. Transistors in VDSM and nanoscale technologies operate within a small voltage range which makes them more susceptible to electrical noise or radiation induced instability than larger technologies [4]. Additionally, the variations in transistors cause each fabricated IC to possess different characteristics which may produce unknown behaviour and lead to errors under normal operation. Therefore, it is of utmost importance to address the impact of these reliability concerns or else the benefits provided by technology scaling will be severely outweighed by erroneous behaviour appearing in-the-field. However, traditional methodologies used to mitigate the impact of these reliability concerns often result expensive for low cost devices, and are therefore utilised primarily for safety-critical applications. On the other hand, low cost devices must be capable of producing error-free behaviour but the cost required to improve their reliability is constrained. This thesis describes novel low cost techniques to improve the reliability of cost constrained systems.

This chapter is organised as follows. Section 1.1 details the motivations and the scope of this research. Section 1.2 presents an overview of the contributions of each chapter and the organisation of this thesis, followed by the list of publications generated from this research in Section 1.3.

1.1 Motivation and Scope of this Research

The first design strategy used to reduce the impact of process variation was to define the worst-case scenarios to set noise margins or timing margins at which even the longest delay variations comply [5], [6], effectively setting the operating frequency of the IC as the one which will not violate those margins. As the effects of variability increased in VDSM technologies due to reduced feature sizes, the frequency of errors appearing in-the-field increased consistently [3], [7]. Faults escaping manufacturing testing manifesting as errors, single-event upsets caused by radiation induced charge deposition, temperature and voltage supply fluctuations causing gate and interconnect delay increase, and aging related problems increasing threshold voltage, have been observed to affect the functionality and performance of circuits and causing logic errors in-the-field at a higher rate than previous technologies.

The traditional solution for online mitigation of erroneous behaviour and reliable system design is hardware redundancy [8], [9]. It consists on the complete or partial replication. By replicating the circuit, the probability of failure is reduced, as it is highly unlikely an error would occur on every replica at the same time. Triple Modular Redundancy (TMR) is the best-known and most widely applied among the hardware redundancy techniques. TMR replicates original circuit so that there are three replicas whose outputs are passed on to a majority voter [10]. TMR is most widely used for safety-critical applications where robustness and data integrity are the top priority (e.g. aeronautics, medical equipment, transportation). Although TMR systems guarantee high level of reliability, its high area and power overhead, 200% of the original circuit, make it non-viable for cost constrained systems.

In recent years, there has been great interest in the development of cost-effective online testing and error monitoring design techniques to enhance the reliability of cost constrained systems. Most low cost techniques are based on a reliability assessment at the logic level of abstraction [11], [12]. At this abstraction level, low cost techniques are capable of leveraging the trade-off between error detection/circuit protection and the associated hardware cost. Example of cost-effective design techniques include partial replication using approximate logic of a slice of a circuit [13]–[15], enhanced reliability of flip-flops [16], [17] and low cost error detection to detect errors due to intermittent faults [18]. The aforementioned techniques are just a small sample of the methodologies that improve the reliability at a lower cost than traditional error detection and mitigation methods. The following chapters provide a larger review of the existing low cost techniques. However, at the logic abstraction level, there is a plethora of different approaches that can be taken to enhance circuit reliability of cost constrained systems.

Summarizing, the increased reliability concerns caused by process variability in VDSM technologies has a significant impact on the functionality and performance gains obtained by greater integration. Traditional solutions for concurrent error detection (CED) and circuit protection used to enhance the reliability of modern ICs incur in a high cost and are unsuitable for cost constrained systems. Therefore, it is necessary to develop cost-efficient techniques to detect errors and protect circuits where the cost and power consumption are restricted.

1.2 Thesis Contributions and Organisation

This work presents three contributions all aimed to enhance the reliability of cost constrained systems by implementing three novel design techniques applicable at the logic level of abstraction. Aside from Chapter 2 which presents an overview of the theoretical background of this research, Chapters 3, 4 and 5 each present one of the novel reliability enhancing techniques. Finally, Chapter 6 presents the conclusions and the future work directions this research might lead to.

Chapter 3 presents the first contribution in the form of a novel low cost fault tolerance design technique applicable at the logic level, that selects and protects the most susceptible workload on the most susceptible logic cones of a circuit, by targeting both timing-independent and timing-dependent errors. The workload susceptibility is ranked as the likelihood of any error to bypass the inherent logic masking of the circuit and propagate an erroneous response to its outputs when that workload is executed. The susceptible workload is protected by a partial Triple Modular Redundancy (TMR) scheme. For the evaluation of the fault-tolerance ability of this technique, the surrogate error models of timing-independent errors induced by stuck-at faults and transient input bit-flips are considered. Additionally considered are the timing-dependent errors induced by transition faults and temporary erroneous output transitions. This technique can achieve a similar error coverage when protecting the same number of patterns with an average 39.7% area/power cost reduction. Furthermore, it can improve by 95.1% on average the achieved error coverage with a similar area/power cost. Particularly, when protecting only the 32 most susceptible patterns, an average error coverage improvement of 63.5% and 58.2% against errors induced by stuck-at and transition faults is achieved, respectively, compared to the case where the same number of patterns are protected without any ranking. Additionally, this technique produces an average error coverage improvement of 163% and 96% against temporary erroneous output transition and errors induced by bit-flips, respectively. These error coverage improvements incur in an area/power cost in the range of 18.0-54.2%, which corresponds to a 145.8-182.0% reduction compared to TMR.

Chapter 4 presents the second contribution which consists of a novel low cost probabilistic online error monitoring technique that produces an alarm signal when systematic erroneous behaviour has occurred over a pre-defined time interval. Particularly, this technique monitors the signal probabilities at the outputs of the most vulnerable logic cones of a circuit concurrently to its normal operation. To detect systematic erroneous behaviour, the collected data is compared on-chip against the signature of error-free behaviour. This technique is applied to a set of the EPFL and ISCAS benchmarks and results demonstrate that it achieves, on the largest circuits, an average error coverage of 84.4% and 73.1% of errors induced by intermittent bit-flips and intermittent stuck-at faults, respectively, with an average area cost of 1.66% and an error detection latency in the range of [0.01, 3.3] milliseconds.

Chapter 5 presents the last contribution consisting of a circuit approximation technique that can be used for low cost non-intrusive concurrent error detection and for selective fault tolerance. The circuit approximation is based on finding functionality at the logic level that behaves similarly to single logic gates or constant values. An algorithm is proposed to select the input subsets to approximate. This algorithm is applied to arithmetic circuits as a proof of concept. Furthermore, an application of this technique to a set of the LGSynth91 benchmark circuits shows an average input coverage of 33.59% of all the input space with an average 7.43% area cost. Additionally, using the generated approximate logic circuits for selective fault tolerance results in significant area cost reductions compared to existing techniques. In addition to that, it is shown that the generated approximate logic circuits can be used for low cost concurrent error detection.

1.3 Publications

The following peer-reviewed publications have been produced from the research presented in this thesis;

M. D. Gutierrez, V. Tenentes, and T. Kazmierski, *"Susceptible workload driven selective fault tolerance using a probabilistic fault model"* in the 22th IEEE International Symposium on On-Line Testing and Robust System Design 2016. IOLTS. Proceedings, 2016.

M. D. Gutierrez, V. Tenentes, D. Rossi and T. Kazmierski, *"Low Power Probabilistic Online Monitoring of Systematic Erroneous Behaviour"* in the 22th IEEE European Test Symposium 2017. ETS. Proceedings, 2017.

M. D. Gutierrez, V. Tenentes, D. Rossi and T. Kazmierski, *"Susceptible Workload Evaluation and Protection using Selective Fault Tolerance"*. Journal of Electronic Testing: Theory and Applications, August 2017, Vol 33, pp 463-477.

M. D. Gutierrez, V. Tenentes, and T. Kazmierski, *"Low Cost Error Monitoring for Improved Maintainability of IoT Applications"* in the International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems. DFT. Proceedings. 2017.

The following research paper is under preparation:

M. D. Gutierrez and T. Kazmierski, *"Single Gate Approximate Logic Circuits"*

Chapter 2

Background

The appearance of faults in-the-field has been steadily rising with the reduction of technology nodes and increased integration density [19]. Online testing and monitoring techniques have been developed in the past to enhance the reliability of ICs by testing, monitoring and protecting circuits in-the-field. This chapter introduces the basic concepts related to this thesis and presents an overview of the previous research on this field. Additionally, chapters 3, 4 and 5 have their own background placing each of contribution in context of the relevant works in the field. This chapter is organised as follows. Section 2.1 reviews the concept of cost constrained systems. Section 2.2 presents the reliability challenges of cost constrained systems posed by technology scaling and power management techniques and describes the types of faults and errors that may appear as a result. Section 2.3 provides an overview of fault and error detection techniques such as concurrent error detection and built-in self test. Section 2.4 gives an overview of the circuit protection mechanisms most commonly deployed while Section 2.5 briefly describes the state if the art low cost error detection and protection techniques. Finally, Section 2.6 presents the objectives of the research presented in this thesis.

2.1 Cost Constrained Systems

The demand for cheap, mass produced integrated circuits (ICs) has risen significantly over the past years due to the exponential growth of personal devices and those connected to the Internet of Things (IoT). A few examples of such devices are tablets, smartphones, wearables, surveillance cameras, parking, environmental or geo-monitoring sensors. These kind of devices are designed to be cheap to implement, manufacture and operate, thus they are referred to as cost constrained systems.

In order for cost constrained systems to achieve a correct functionality while meeting its cost limitations, transistor sizes have been decreasing due to a process known as technology scaling. Additionally, to reduce operation costs, many low power design techniques have been developed in the past to achieve power consumption reduction. The goal of these techniques to minimize the power consumption as much as possible without having a significant impact on the performance of the system. The power consumption of a digital IC is calculated as follows [20] [21]:

$$P_{total} = \alpha\left(C_L \cdot V_{dd}^2 \cdot f_{clk}\right) + I_{leakage} \cdot V_{dd} \tag{2.1}$$

Which can be written as:

$$P_{total} = P_{dynamic} + P_{leakage} \tag{2.2}$$

Where $P_{dynamic}$ is the dynamic power dissipated by the switching activity (α) of the transistors in the circuit, with load capacitance C_L and operating at frequency f_{clk} with a supply voltage V_{dd}. $P_{leakage}$ is the leakage power which is the dissipated when the transistors are idle.

Low power design techniques aim to reduce either or both the $P_{dynamic}$ and the $P_{leakage}$ components of the overall power consumption in ICs. These techniques deploy a number of different tactics to achieve the targeted power reduction, albeit with a reliability cost that results in non-trivial challenges for modern IC technologies.

2.2 Reliability challenges of Cost Constrained Systems

The lifetime reliability of semiconductor devices in cost constrained systems may be affected by multiple sources [22], [23]. This section provides an overview of the reliability challenges of cost constrained systems posed by the scaling down of technology nodes and by power management techniques.

2.2.1 Technology scaling

The driving reason behind continuous reduction of technology nodes is fabrication costs. The cost of fabricating ICs is dependent primarily on the number of steps required by the technology node, therefore, including as many dies per wafer as possible is desirable. In addition of smaller transistor sizes occupying less physical space, they require less power to operate at the same frequencies, which also implies that a higher performance is possible with the same power consumption. An approximation of transistor propagation delay is given by Equation 2.3 [24]:

$$T_{pd} \propto \frac{C_L \cdot V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \tag{2.3}$$

where T_{pd} is the transistor propagation delay, C_L is the load capacitance, V_{dd} is the supply voltage, V_{th} the threshold voltage and α is used for modelling short channel effects [25]. Both C_L and V_{th} are proportional to the size of the transistor, therefore reducing sizes leads to shorter propagation delay, allowing for greater circuit performance. Incidentally, this behaviour also allows to reduce power consumption without increasing transistor delay by reducing the supply voltage.

On the other hand, reducing transistor sizes lowers the threshold voltage, which produces a significant increase in the leakage power. The primary contributor of leakage power is the sub-threshold leakage current that may be approximated as Equation 2.4 [24]:

$$I_{ds} \propto e^{\frac{V_{gs} - V_{th}}{n \cdot V_T}} \tag{2.4}$$

where V_{gs} is the gate-to-source voltage, V_{th} is the threshold voltage and V_T is the thermal voltage. The sub-threshold leakage current increases exponentially as the V_{th} is reduced. As a result, the leakage power also increases exponentially.

Scaling down of technology nodes has also had an impact on the reliability of cost constrained systems. As transistors get smaller, they become prone to all sources of instability. Additionally, the higher integration levels allowed by smaller transistors increase system complexity and power density, resulting in a higher probability of producing erroneous behaviour [7]. The main contributors to the impact of technology scaling on the reliability of cost constrained systems are process variation and circuit wearout. [3], [22], [26].

2.2.1.1 Process Variation

The IC fabrication process is not exact and may produce variations in the characteristics of transistors within the same technology node. This problem is known as Process Variation (PV) [1], [27], [28]. Transistor sizes in very deep submicron (VDSM) technologies are significantly smaller than the light wavelength used in the lithography process to imprint the circuits on the silicon wafer which poses a challenge to modern manufacturing processes [27]. The imprinting can be imprecise leading to uneven surfaces on transistor channels thus creating differences in the electrical properties throughout the chip. In addition to that, sub-wavelength lithography suffers from printability issues like variations on the focus, pitch and light source exposure, whose effects can make the lines imprinted on the silicon wafer of a size different than the one designed on the mask [29]. These variations cause different threshold voltages in transistors on the same die, which produces uncertainty in circuit delay of critical timing paths, and may lead to performance degradation.

Figure 2.1 illustrates the distribution of the extracted sub-threshold voltage V_{th}, effective channel length L_{eff}, and carrier mobility μ of a 65nm chip [30]. The variation of the parameters of the transistors throughout the circuit follows a gaussian distribution whose standard deviations from the mean where of 5%, 4% and 21% respectively. These variations are expected to keep increasing as technology nodes scale down.

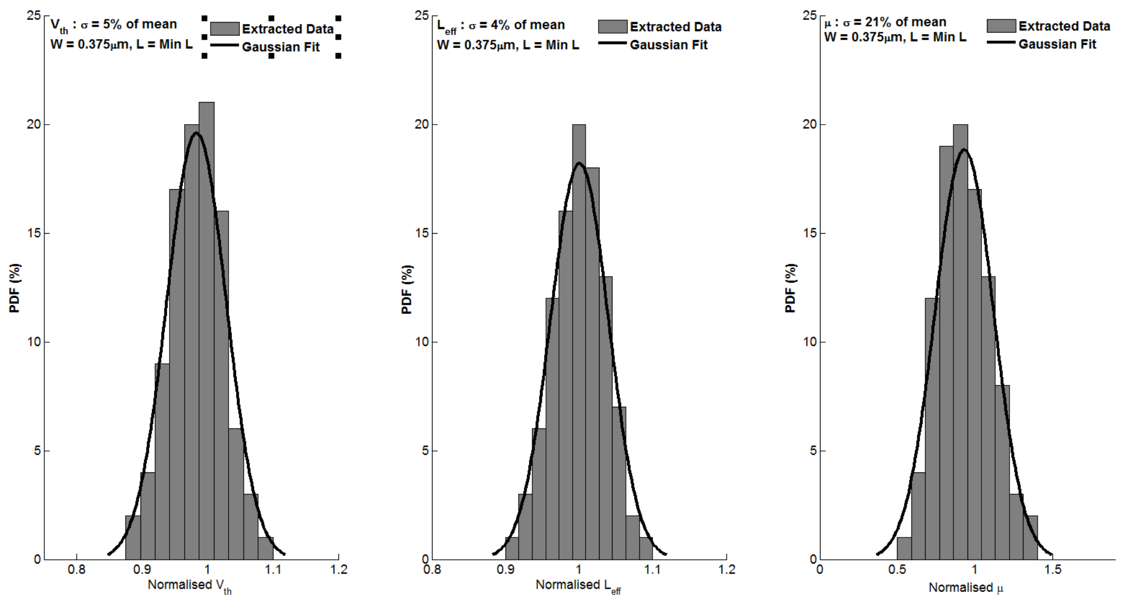


FIGURE 2.1: Variations of V_{th}, L_{eff} & μ [30]

2.2.1.2 Circuit Wearout

Semiconductor circuits degrade gradually due to normal operation. However, technology scaling allows a greater integration increasing power density which leads to accelerated circuit wearout and reduced lifetime of devices [31], [32]. Among the circuit wearout effects that have an impact on the reliability of a circuit are Negative bias temperature instability (NBTI) and Hot Carrier Injection (HCI)[33]. NBTI is generated by positive charge accumulating at SiO_2 interface and the oxide while under negative bias, causing the threshold voltage to change, which decreases channel mobility thereby increasing the intrinsic delay of the transistor [31]. HCI is caused by carriers in the transistor channel (electrons or holes) gaining enough energy to tunnel through the oxide, manifesting as leakage current and changing transistor parameters gradually damaging the circuit [34].

2.2.2 Power Management Techniques

As shown by Equation (2.1), there are two components in the total power consumption: dynamic and leakage power. Various power management techniques have been developed to reduce power consumption, either dynamic or leakage in cost constrained systems [35]–[37] . Following is an overview of the most commonly used techniques.

2.2.2.1 Power Gating

Leakage power has become a major concern regarding the overall power dissipation of nanoscale technologies [38]. One of the most commonly used leakage power reduction techniques is Power Gating (PG) [36]. This technique consists of completely turning off sections of the circuit that are expected to be idle for a certain amount of time, effectively preventing any leakage power consumption on the disabled sections. A drawback of this technique is that re-enabling the power-gated sections may take a long time period, in addition to the spike in power consumption that a re-initialization of the circuit may cause.

PG may cause a problem known as ground bounce noise, which is generated by the fluctuations on the power rails [39]. Figure 2.2 shows a power gated logic circuit affected by ground bounce noise. When the transition from sleep/off mode to active/on mode causes the sleep transistor (located at the bottom) to turn ON to saturation, a sudden discharge creates a current surge at the moment of the mode change, which results in voltage fluctuations across the power supply rails. These fluctuations may change the state of sequential elements in the circuit, potentially leading to unknown behaviour.

2.2.2.2 Clock Gating

In digital circuits, the high switching activity of the clock tree dissipates the most dynamic power compared to the rest of the system [40]. Clock gating (CG) techniques were developed to prevent the switching of the clock of sequential elements that are not required to update every clock cycle. The basic implementation requires the insertion of a clock gating module consisting of a latch and an AND gate, which controls the the switching of the clock signal. Similar to power gating, a section of the circuit that is clock-gated may not power up in time, generating glitches and affecting timing which lead to erroneous behaviour [38].

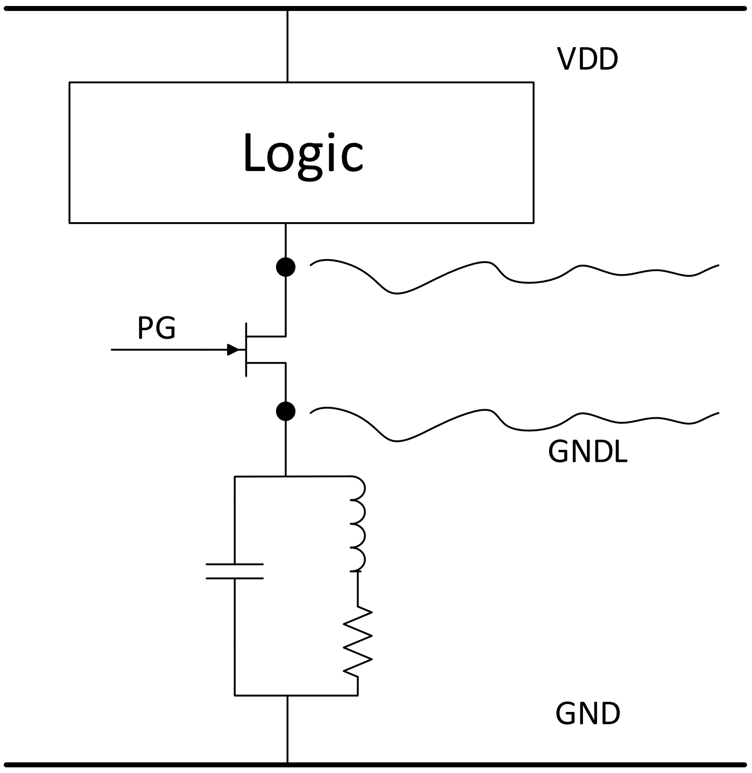


FIGURE 2.2: Ground bounce noise in a power gated structure [39]

2.2.2.3 Dynamic Voltage and Frequency Scaling

As denoted by Equations (2.1) and (2.2), the dynamic power consumption is given as:

$$P_{dynamic} = \alpha \left(C_L \cdot V_{dd}^2 \cdot f_{clk} \right) \tag{2.5}$$

where α is the switching activity, C_L is the load capacitance of the transistors Vdd is the supply voltage and f_{clk} is the frequency of the clock.

Notice from Equation (2.5) that V_{dd} has a quadratic relation to $P_{dynamic}$, therefore reducing it is the most effective means to decrease $P_{dynamic}$. Dynamic Voltage and Frequency Scaling (DVFS) is a power management technique which consists of reducing the supply voltage, to reduce dynamic power consumption. On the other hand, lowering V_{dd} increases the intrinsic transistor delay, which restricts the clock frequency at which the circuit can operate. This forces the reduction of the f_{clk} to prevent erroneous behaviour due to timing-related errors, which may occur if a task execution deadline of a time-constrained system is violated due to the slower clock frequency [41], [42]. Therefore, this technique may be be applied whenever there is slack time on the computation that allows a timely execution of the task with a slower f_{clk} [43]–[45].

2.2.3 Faults in Cost Constrained Systems

As discussed previously, technology scaling and low power design techniques may impose reliability challenges on cost constrained systems. The reliability of these systems may be affected by faults appearing in-the-field during normal operation. These faults can be classified into three categories according to their duration [46]: permanent, transient and intermittent faults.

2.2.3.1 Permanent Faults and Fault Models

Permanent faults may appear due to defects escaping manufacturing testing, or by aging and circuit wearout effects caused by the gradual degradation of devices under typical operating conditions. These faults manifest continuously therefore having a lasting impact the system reliability.

The IC fabrication process is not perfect which may result in defective ICs being imprinted onto the silicon wafer. This process imperfection is exacerbated by the increased fabrication complexity as technology scales down. In order to verify that an IC was fabricated correctly, it is tested against manufacturing defects. However, due the diversity of possible defects in VDSM technologies, the generation of tests for real defects poses a significant challenge. Therefore, fault models were developed to emulate the effects that the different defects may have on the operation of the IC. These fault models should reflect the behaviour of defects with accuracy, and be computationally efficient. Various fault models have been proposed [47], of which the most widely used is the the stuck-at fault model.

Under the stuck-at fault model, a stuck-at fault is a logic-level (logic 0 or logic 1) defect which affects the logic value of the nets in a circuit. These nodes may belong to the primary inputs or outputs as well as any internal node in any gate. This type of faults force the value of the affected node to a constant logic 0 or logic 1, referred to as stuck-at-0 or stuck-at-1, respectively. Transistor-level stuck-at is another fault model [48] which targets transistor defects such as stuck-open (missing connection to ground) or a stuck-short (a conducting path between the power source and ground).

Other fault models target delay defects that affect the timing of the circuit. A delay fault produces an extra propagation delay in the affected path in such a way that the total delay is larger than the time limit required for a fault-free operation. Popular delay fault models include the gate-delay fault model, which models the gate interconnect and pin-to-pin delay, the path-delay fault model, which considers the delay distributed over a group of interconnected gates, and the transition fault model. The transition fault model is the most widely used delay fault model [49], [50]. The transition fault model requires the propagation of transitions in the circuit while operating at the desired frequency.

To propagate a transition, it is necessary to set apply two patterns at the inputs of the circuit, one for initialization of the relevant signals to known and controllable values, and one to produce the desired transition. This fault model works under the assumption that only a single gate is affected by a delay fault, and that the extra delay caused by the fault prohibits the correct transition from reaching a primary output or sequential element.

Fault models are then used by the Automatic Test Pattern Generation (ATPG) tools to generate test patterns of digital circuits [47]. ATPG tools define a list of faults depending on the fault model targeted an find those patterns that detect such faults. Then, they collect fault coverage statistics (the percentage of faults that may be detected) by deploying fault simulation campaigns. A fault simulation campaign consists of injecting modelled faults in a simulation environment and applying test patterns. The outputs of the circuit tested are compared against the values expected in the presence of the targeted faults to determine whether the fault is detected.

2.2.3.2 Transient Faults

Transient faults have a temporary impact on the reliability of ICs. These faults are caused by single-event upsets (SEU) or by other sources of instability and noise in modern ICs [51]. A SEU are produced by cosmic and alpha particle strikes. When a particle hits a node in the circuit, it may generate a short current pulse with enough magnitude that propagates through the node and reaches a storage element. If the current pulse has enough energy, the state of the storage element flips, effectively causing a bit-flip. The transistor sensitivity to SEUs is dependant on the transistor capacitance and supply voltage [7], which are both affected by technology scaling. The SEUs were known to cause bit-flips in storage elements for years, however, as technology scaled into VDSM technologies, they have also become a cause of temporary gate failures in combinational logic. This is due to the reduced threshold voltages of transistors in smaller technologies.

Another source of transient faults is the intrinsic noise in modern ICs. This noise is caused by inductive effects generated from the fast switching speed and high transistor density of modern ICs [52]. Process variation can also contribute to the generation of transient faults, due to the uneven threshold voltages on the transistors through out the chip.

2.2.3.3 Intermittent Faults

Intermittent faults may appear and disappear repeatedly. These faults are activated in-the-field by defects that escaped manufacturing testing, by the systematic of power or thermal constrains, or by aging. Intermittent faults cause bursts of errors that repeat

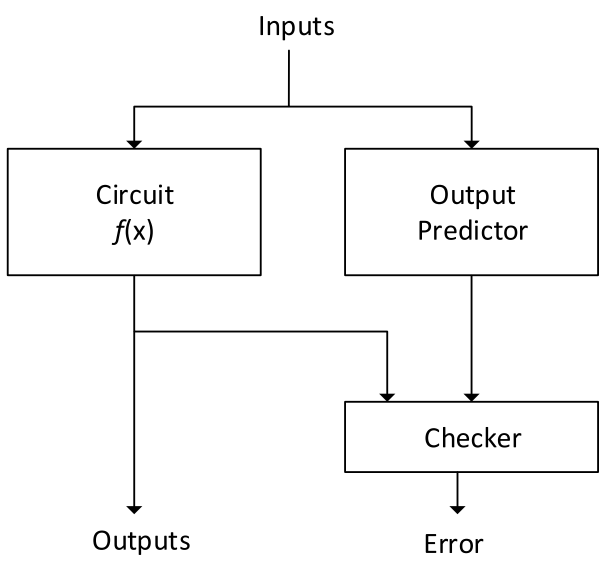


Figure 2.3: General architecture of a CED mechanism [55]

periodically on the same places under specific operating conditions [53]. They manifest as multiple bit-flips on sequential or combinational logic, or exhibit a behaviour similar to permanent faults for a finite period of time.

Intermittent fault models were introduced as a variation of existing fault models [54], which include Intermittent stuck-at, Intermittent bit-flips and Intermittent delay. These fault models are based on the permanent fault models described previously, with the difference that the activation time of the intermittent faults is finite.

2.3 Fault and error detection techniques

Faults appearing in-the-field may impact the circuit reliability, producing an erroneous output. Therefore, it is of great interest to develop techniques in order to detect when errors, caused by permanent, transient or intermittent faults, have occurred.

The existing fault and error detection techniques can be classified into two categories: Concurrent Error Detection (CED) and Built-in Self-Test (BIST). CED targets the detection of logic errors occurring during normal operation. On the other hand, BIST mechanisms target the identification and diagnosis of faults in the circuit. An overview of both CED and BIST techniques is presented as follows.

2.3.1 Concurrent error detection (CED)

Concurrent error detection (CED) techniques are used to enhance system reliability [56]–[60]. The general architecture of a CED mechanism is shown in Figure 2.3 [55]. CED techniques operate according to the following principle: Consider a circuit that performs a function f on an input pattern $x = (x_0, x_1, \cdots, x_n,)$. The CED mechanism consist of another circuit which concurrently predicts the output of the original circuit or a specific characteristic corresponding to such output for a given input pattern x. Finally, a checker determines whether the predicted output or its specific characteristic corresponds to the one produced by the original circuit given an input pattern x. If a mismatch occurs, an *error* signal is asserted indicating an error has occurred in the original circuit, the output predictor or in the checker.

The specific characteristic predicted by the output predictor circuits may consist of: the full or partial output itself, its parity or the number of 1's or 0's. Following is an overview of a number of CED techniques.

2.3.1.1 Hardware duplication

CED using hardware duplication with comparison (DWC), shown in Figure 2.4, is a technique in which the output predictor consists of a full or partial replica of the original circuit [61]. This replica performs the same logic function as the original circuit and it is expected to produce the same outputs. If the output predictor is only a partial replica, it is only capable of detecting errors on the outputs that are fully replicated [62]. A comparator is used to check if the outputs of both circuits match. In the case of a mismatch, the system indicates that an error has occurred.

A DWC system ensures an error-free operation only when both the original and the replica circuits do not generate identical errors, assuming that the comparator is also error-free. The most basic comparator consists of an array of XOR gates whose output is asserted when its inputs are different. However, since the comparator is essential to the error-free operation of the DWC system, self-checking comparators have been developed to guarantee an error-free comparison [63], [64].

2.3.1.2 Error Detecting Codes

CED techniques based on error detecting codes (EDCs) rely on encoding a characteristic of the output of a circuit along with the output itself. These characteristic may be the parity of the output, the number of logic 0's or 1's in the output, among others.

Of the EDCs, parity prediction is the most widely used CED technique. The parity of a set of binary digits is the number of logic 1's and it can be even or odd. Figure

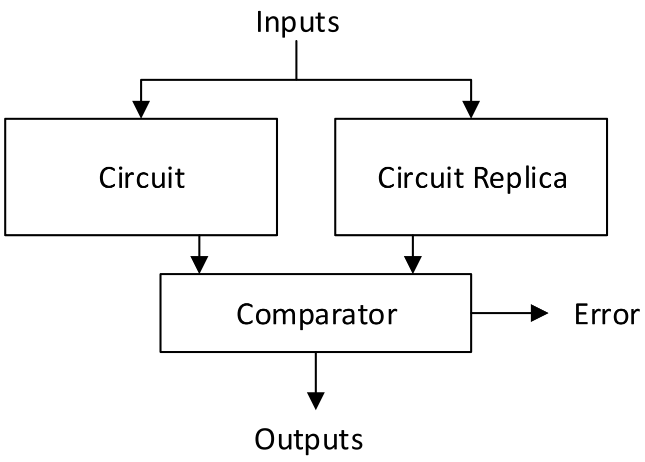


FIGURE 2.4: Duplication with comparison CED [61]

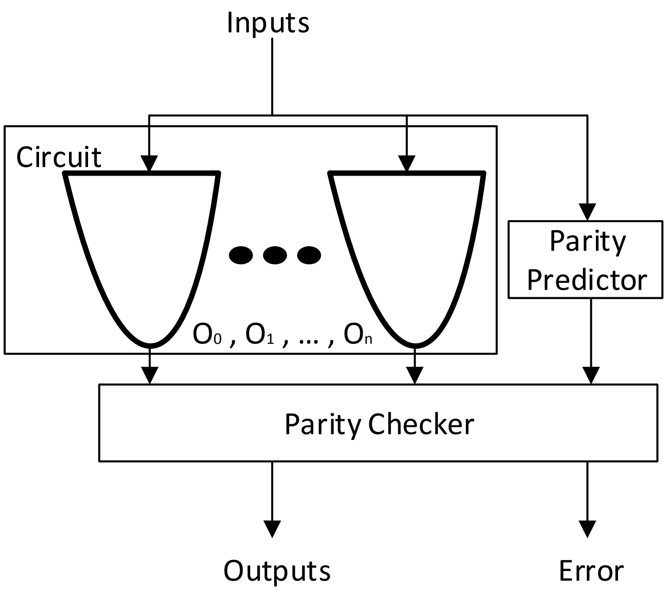


FIGURE 2.5: Parity prediction with one parity bit [64]

2.5 presents the architecture of a single parity bit CED technique. The circuit with n outputs must be designed such that there is no logic sharing between its logic cones. Therefore, single faults can only affect one output at a given time. The parity of the outputs is predicted and passed on to the parity checker along with the outputs to check whether the actual parity of the outputs agrees with that of the predicted parity bit [64]–[66].

A major drawback of parity-based error detection is that it imposes the restriction on the

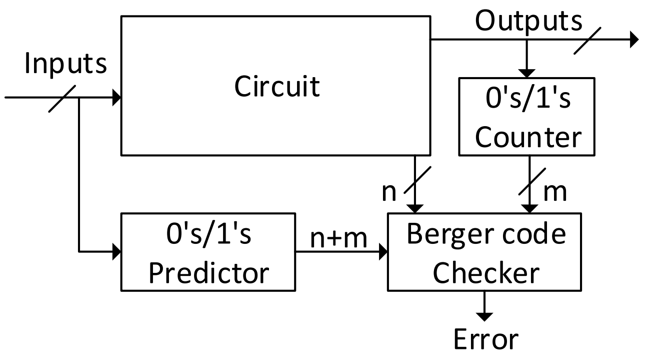


Figure 2.6: Concurrent error detection with Berger codes [67]

circuit that no logic sharing between the cones is allowed. This restriction often results in a high area cost. In order to reduce the area cost, multiple parity bits may be generated by partitioning the outputs into different parity groups, each with an associated parity bit. Logic sharing is only allowed between logic cones belonging to different groups, thus a single fault can affect at most one parity group.

Other CED techniques using error detecting codes have been proposed that rely on unidirectional codes. Unidirectional codes are deployed under the assumption that only unidirectional logic errors may occur; i.e, either only 0→1 or 1→0, but not both. The most widely known unidirectional cone is the Berger code [67], shown in Figure 2.6.

Using a Berger code, the encoded output is formed by the output of the circuit and *checkbits* containing the number of 0's (or 1's) in the output itself. Therefore, the length of the encoded output is calculated as $n+m$ where n is the number of bits in the circuit output and m is the length of the checkbits, which is calculated is follows:

$$m = \lceil log_2(n) \rceil \tag{2.6}$$

This results in the minimum number of bits required to represent the number of 0's (or 1's) in the output.

2.3.2 Built-in Self Test (BIST)

Built-in Self-Test (BIST) techniques have been proposed in the past to test a circuit in-the-field [68], [69]. BIST mechanisms, illustrated in Figure 2.7, work under the following principle: A test pattern generator (TPG), typically implemented using a linear feedback shift-register, and an output response analyser (ORA) are integrated on-chip along with the circuit under test (CUT). The TPG generates test patterns in order to sensitize the

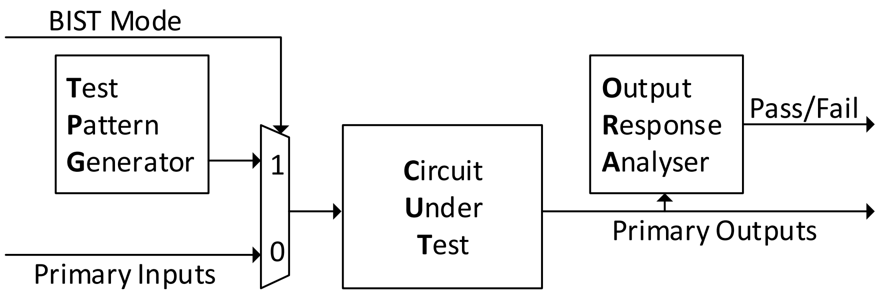


Figure 2.7: Basic BIST architecture [68]

CUT. This process is similar to the test-patterns generated by the ATPG and input into the circuit by the Automatic Test Equipment (ATE) deployed during manufacturing testing. The outputs of the CUT are examined by the ORA to detect if the circuit has become faulty. BIST mechanisms are tailored to a targeted fault model (stuck-at, transition, etc...) and thus are only capable of detecting specific types of faults. The cost of these BIST mechanism is often small, however, the design of BIST mechanisms is dependant on the nature of the logic of the CUT and the targeted fault model, which makes automating these mechanisms a difficult task. It is important to note that BIST mechanisms, are not designed to detect logic errors like CED techniques. These mechanisms are designed to test the circuit for faults, with no consideration if a fault is causing errors, or may cause an error in the future.

BIST mechanisms were originally developed to assist manufacturing testing by moving the test pattern generation and the analysis of the output response on-chip [70]–[73]. However, with the increased reliability concerns of modern VDSM technologies, BIST mechanisms have been adapted to perform tests online, after manufacturing testing and while the circuit is in-the-field. The efforts to use BIST mechanisms in-the-field can be categorized in two main directions: Online BIST and Concurrent BIST.

2.3.2.1 Online BIST

Traditional BIST mechanisms were mainly used for quality assessment after the manufacturing process or prior shipping the circuit to a vendor [70]. However, the need to ensure system integrity during the lifetime of ICs, particularly for safety-critical applications, resulted in the design of BIST mechanisms capable of performing tests online and in-the-field.

Online BIST mechanisms, allow the detection of faults during the lifetime of digital systems [32], [46], [74]. The basic principle of offline BIST is preserved for the implementation of online BIST solutions (Figure 2.7). Online solutions however, apply periodically to the CUT a heavily compressed deterministic test-set and checks the response to detect the presence of faults. The frequency at which this periodic test is applied may depend on an automatic process built in the circuit, the system software, or by an user input [75]–[77]. In order to apply the test, the state of the circuit must change to BIST (or test) mode, therefore being unable to perform concurrently to normal operation.

2.3.2.2 Concurrent BIST

Concurrent BIST mechanisms were introduced to allow testing for faults in the circuit concurrently to normal operation [78]–[81]. These mechanisms use input patterns and responses generated off-line using ATPG, which are compacted and stored in on-chip memory. When one of the precomputed patterns is present at the primary inputs of the CUT during normal operation, the circuit response is checked against the stored expected response. These kinds of tests are non-intrusive, as they have no impact on circuit performance, however storing the sufficient patterns and responses may incur in a significant area and power cost. Additionally, the test times are unpredictable as in order to complete the test, all the required input patterns of the test must appear at the inputs, which may result in a long fault-detection latency.

2.4 Fault and Error protection techniques

To enhance the reliability of cost constrained systems, techniques that protect circuits against faults and errors have been developed in the past. In contrast to CED and BIST techniques which target only detection, these techniques have different targets: Preventing the appearance of faults or errors, or mitigating their effects to ensure correct functionality despite their presence. Protecting circuits indicates that some form of *Fault Tolerance* is included. Particularly, the protection against faults appearing in-the-field is commonly referred to as *Hardening.* Many hardening techniques developed in the past rely on providing extra resources to add some form of redundancy to the system. This redundancy may come from replicated hardware, intrinsic information redundancy in the outputs of the circuit or timing redundancy in re-execution methods and software recovery. Following is an overview of the most widely used techniques to protect a circuit against faults or errors.

2.4.1 Hardware Redundancy

Hardware redundancy consists of fully or partially replicating a circuit. One of the classic hardware redundancy schemes is Triple Modular Redundancy (TMR) [8], [82], [83].

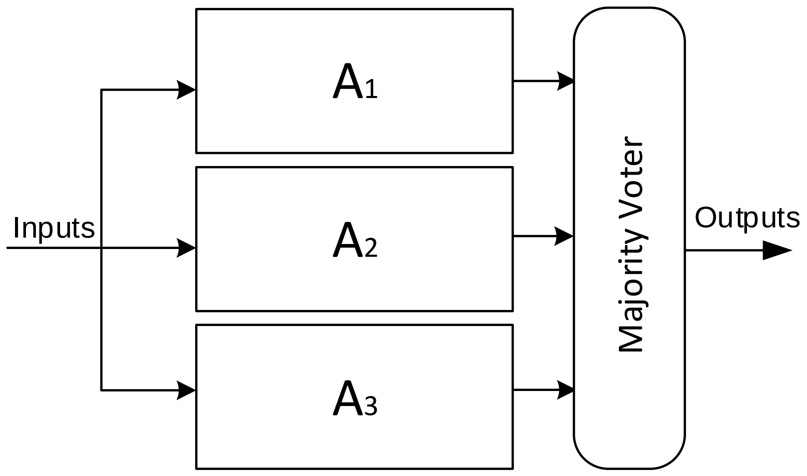


Figure 2.8: Basic TMR scheme [8]

TMR triplicates the system whose outputs are passed onto a majority voter circuits to compare them, selecting the majority values. TMR schemes may be applied to triplicate whole systems, blocks or even cells. The basic TMR scheme is illustrated in Figure 2.8.

2.4.2 Error Correcting Codes

Error correcting codes (ECC) rely on introducing information redundancy in systems or using the one intrinsic to some applications in order to correct errors as they occur [84]. If an error appears, ECCs are designed in such a way that there is enough information at the encoded output to reconstruct the correct data from the faulty data. Many ECCs have been proposed as an extension of error detecting codes (section 2.3.1.2). Parity based techniques have been reported [85], [86] applied primarily in communication channels or storage elements. Hamming Codes extend the concept of parity checks to allow for single error correction and double error detection [87].

2.4.3 Timing Redundancy and Software re-execution

Adding timing redundancy to a system consists of re-executing an instruction or re-applying an input when an error is detected [88]–[90]. This process often involves using the system software to issue a re-execution command which recomputes the last erroneous data [91]. The main advantage of implementing error correction using software is that it has no impact on design and area cost, although it imposes a high performance penalty.

2.5 Low Cost Error Detection and Correction

Traditional CED and Hardening techniques are often expensive in terms of area or power for cost constrained systems. Therefore they have been used in circuits operating in harsh conditions or in safety-critical applications where data integrity is the top priority, such as aerospace electronics or medical devices. For instance, duplication with comparison CED often incurs in over 100% area cost [55] including the comparator. Furthermore, TMR entails an area and power cost greater than 200% of the original circuit. These costs result prohibitive for systems where low cost and low power consumption are the priority.

Low cost and low power techniques have been proposed to reduce the cost by trading off error detection and correction capabilities for a reduced area and power overhead. The main premise is the following: If providing CED or hardening a whole circuit is too expensive, then only the parts of the circuit or the application that are the most vulnerable to faults or errors must be checked or protected [92], [93]. Some methodologies to identify the vulnerable parts (soft spots) have been proposed [74], [94], [95], which allow low cost techniques to focus their efforts on those parts.

Selective Hardening techniques protect only the vulnerable parts of a circuit [96]. This protection may be done by localized TMR on the soft spots [97]. Other techniques require the analysis of specific workloads to find the sequential elements to harden [98], [99]. Additionally, selective hardening may also be applied at the gate level by duplicating [100] or resizing [101] vulnerable gates.

Instead of protecting particular parts of a circuit, Selective Fault Tolerance (SFT) has been proposed to reduce the cost of TMR by protecting only a specific subset of the input space in combinatorial circuits [102], [103]. SFT relies in an heuristic to arbitrarily select the input subset to protect, guaranteeing TMR-levels of protection for only those input vectors. This results in a reduced area cost compared to TMR.

2.6 Research Objectives

It is evident from the theoretical background presented in this chapter that low cost fault and error detection and correction techniques are and will continue to be a necessity. The reliability of modern ICs continues to be further aggravated by the consistent trend of scaling down technologies into the nanoscale. As discussed before, cost constrained systems are designed to be cheap to produce and often operate under strict power constrains. Many applications of cost constrained systems are not constrained by strict data integrity requirements, allowing a trade-off between cost and fault tolerance. Furthermore, some of the low cost solutions previously developed are tailored to a specific type of faults (permanent or transient), or are dependent on the type of circuit they are targeting (datapath, control, sequential elements).

The aim of this thesis is to develop low cost and low power online testing & monitoring techniques that enhance the reliability of ICs in cost constrained systems. The techniques proposed in this thesis leverage the trade-off between fault and error coverage vs area overhead and power consumption in order to enable different levels of protection at various costs. Furthermore, the proposed techniques are applicable at the logic level to any circuit and are developed to be compatible with the standard electronic design automation (EDA) flow. This compatibility is of utmost importance to reduce the design effort associated with the implementation and inclusion of these techniques in existing and future designs. The techniques are compared qualitatively and whenever possible, quantitative, to existing concurrent error detection, selective hardening and selective fault tolerance techniques.

2.7 Concluding Remarks

This chapter presented the basics of cost constrained systems. The reliability challenges posed by the design, design and operation of cost constrained systems were detailed. The types of faults that may appear during the normal operation of a cost constrained system, such as permanent, transient or intermittent faults, have been identified. The necessity of techniques that detect faults and errors in-the-field has been discussed and some of those techniques have been briefly described. Additionally, an overview of the various circuit protection techniques has been presented. Finally, the research objectives were laid out.

Chapter 3

Probabilistic Selective Fault Tolerance

Section 2.2 of Chapter 2 discusses the reliability challenges of cost constrained systems. These challenges indicate that circuits require protection against faults and errors appearing in-the-field. A number of fault and error protection techniques have been developed in the past to tackle this problem, as presented in Section 2.4. However traditional fault tolerance techniques are often expensive, thus low cost techniques were introduced to reduce the cost of circuit protection (Section 2.5). These techniques trade reliability to reduce the area cost and the power consumption of integrated circuits by protecting only a subset of their workload or their most vulnerable parts. The selection of the workload to protect is done randomly or arbitrarily [102], [103], without consideration if the patterns in the workload are the most susceptible to errors. Automatic Test Pattern Generation (ATPG) may be used to select the input patterns to protect (Section 2.2.3.1), however, the traditional fault models are biased towards the one specific type of fault they model (Section 2.2.3). Additionally, faults appearing in-the-field may not propagate errors, thus they will not affect the overall functionality of the system. Therefore, in the presence of faults not all workloads are equally susceptible to errors [104]. This motivates the need to develop low cost fault tolerance techniques that protect the workload against errors caused by any type of faults.

This chapter presents a novel low cost fault tolerance design technique that selects and protects the most susceptible workload on the most susceptible parts of the circuit by targeting faults that are both timing-independent (i.e. stuck-at) and timing-dependent (i.e transition) errors. The workload susceptibility is ranked as the likelihood of any error to bypass the inherent logic masking of the circuit and propagate an erroneous response to its outputs when that workload is executed. The susceptible workload is protected by a partial Triple Modular Redundancy (TMR) scheme.

This chapter is organized as follows. Section 3.2 presents an overview of previous works on Selective Fault Tolerance, reviews the probabilistic fault model of output deviations, introduces the concepts of uncorrelated and application-specific workloads and presents a motivational example. Section 3.3 describes the proposed probabilistic selective fault tolerance design technique and output deviation-based ranking metrics used for pattern ranking. Simulation results from the application of the proposed technique to a set of the LGSynth'91 and ISCAS'85 benchmarks are discussed in Section 3.4, Finally, the concluding remarks are given in Section 3.5.

3.1 Background

Reliability of devices has been affected by technology scaling despite its advantages [19]. Devices manufactured using 32nm technologies and below are more prone to errors due to all sources of instability and noise [22] due to the elevated cost of mitigating process variability [105] and to the escalation of aging mechanisms [31]. As a result, transient and permanent faults can appear and generate errors in-the-field, therefore techniques to make Integrated Circuits (ICs) fault tolerant are required.

Fault tolerant IC design techniques such as hardware redundancy [106] and error correcting codes are utilized for enhancing circuit reliability. Hardware redundancy consists of the complete or partial replication of a circuit in order to ensure correct functionality. By replicating the circuit, the reliability is increased as it is highly unlikely that an error would occur on every replica at the same time. Triple Modular Redundancy (TMR) utilizes two replicas of the original circuit, whose outputs are passed on to a majority voter [106]. TMR has been widely used for safety-critical applications where robustness and data integrity are the top priority. Although TMR achieves a high level of reliability, it imposes a high area and power overhead, 200% of the original circuit plus the voter circuits, thus it is not viable for low power applications. Selective Fault Tolerance (SFT) and Selective hardening have been proposed to reduce area overhead and power consumption, by protecting only a subset of the workload of a circuit or its most vulnerable parts [102], [107].

Selective hardening aims to protect the most vulnerable parts of a circuit against soft errors [107]. This has been achieved in microprocessors [98], [99] through the identification of the architectural vulnerability factor of state elements which often requires stuck-at fault injection campaigns to calculate. Moreover, selective hardening has also been used in combinational logic to identify and protect vulnerable gates or nodes. This is achieved by propagating signal probabilities at the RT-Level to estimate the likelihood of an erroneous output caused by soft errors [62], [101], [107], [108]. A recent selective hardening technique uses a lightweight algorithm to rank the soft error susceptibility of logic cones in combinational logic according to their size, which is given by the sum of the fan-in and fan-out of its cells [109]. The highest ranked cones are duplicated and compared to detect if an error has occurred, in which case, shadow latches at the input enable a roll-back mechanism to recover from the error, but incurring in a time penalty.

On the other hand, SFT as introduced by [102], [103], ensures functional protection of a pre-defined set of input patterns, which are referred to as workload, by using a partial TMR scheme. However, in the presence of a fault, not all input patterns are equally susceptible to it. Some patterns are less protected by the inherent logic masking of a circuit. When such patterns are executed in the presence of faults, the probability that the logic masking will be bypassed and an erroneous response will be generated is higher. Such patterns are defined as the susceptible workload. Previous works on SFT rely on randomly selecting the workload to protect without examining the susceptibility of that workload to faults.

Probabilistic fault models were developed for ranking test patterns according to their ability to sensitize the logic cones of a circuit that are more likely to propagate an erroneous response. Probabilistic fault models are known for improving both the modelled and the un-modelled defect coverage of tests, while not being biased towards any particular type of faults. Output deviations (OD) were introduced in [110] as an RT-Level fault model calibrated through technology failure information that stems from technology reliability characterization, such as inductive fault analysis [111]. This model is utilized for selecting the input patterns that maximize the probability of propagating an erroneous response to the primary outputs. The input patterns with the highest output deviations have a greater ability to bypass the inherent logic masking of the circuit. In [112] is shown that selecting input patterns with high output deviations tends to provide more effective error detection capabilities than traditional fault models. In [113], a test set enrichment technique for the selection of test patterns is proposed. Output deviations have also been used for enriching the unmodeled defect coverage of tests during x-filling [114] and linear [115]–[117] and statistical [118] compression. The output deviation-based metric proposed in [116], was shown to increase the unmodeled defect coverage of test vectors by considering both timing-independent defects, such as stuck-at faults, and timing-dependent defects, such as transition faults which require two patterns.

3.2 Motivation

In this section, the concept of Selective Fault Tolerance and the probabilistic fault model of output deviations (OD) are reviewed and the different types of workloads considered in this chapter are briefly introduced. The effectiveness of OD in detecting errors induced by multiple types of faults compared to the random selection of input patterns is presented as a motivational example.

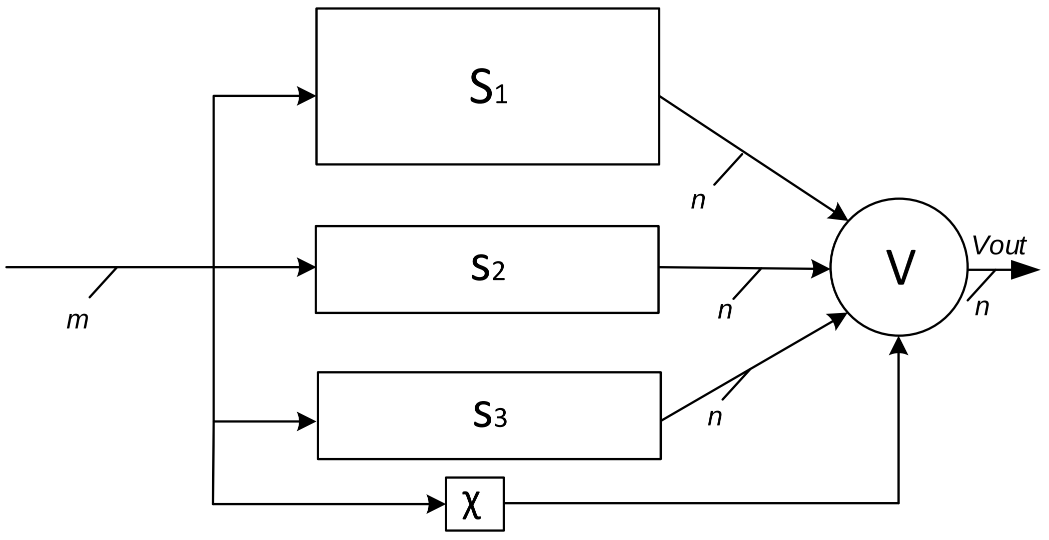


FIGURE 3.1: Previous Selective Fault Tolerance design

3.2.1 Selective Fault Tolerance

Selective Fault Tolerance (SFT) was proposed as a modification of TMR. SFT reduces TMR cost by protecting the functionality of the circuit for only a subset of input patterns [102]. This input pattern subset X_1 is selected randomly by the designer. The input patterns within the subset are ensured to be protected with the same level of reliability of TMR, while the rest are not guaranteed protection.

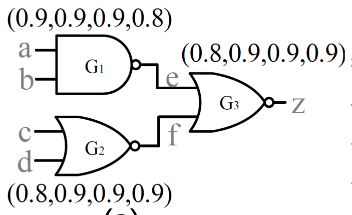
Figure 3.1 depicts the existing technique of SFT design. For a circuit S_1 to be protected using SFT, two smaller circuits s_2 and s_3 are generated. The behaviour of circuits s_2 and s_3 with a protected set X_1 is described as follows:

$$S_1(x) = s_2(x) = s_3(x), \qquad x \in X_1 \tag{3.1}$$

$$(S_1(x) = s_2(x)) \vee (S_1(x) = s_3(x)),\ x \notin X_1 \tag{3.2}$$

To determine if the input x falls within the protected set X_1, the characteristic function $\chi(x)$ must be defined [103]. The output of this function is passed on to a modified majority voter as shown in Figure 3.1, where the outputs of s_2 and s_3 are considered if and only if $\chi(x) = 1$, otherwise the output of the protected design is the output of circuit S_1.

$$\chi(x) = \begin{cases} 1 & x \in X_1 \implies S = V_{out} \\ 0 & x \notin X_1 \implies S = S_1 \end{cases} \tag{3.3}$$



(a)

Inputs, z	$p_{e,0}$	$p_{e,1}$	$p_{f,0}$	$p_{f,1}$	$p_{z,0}$	$p_{z,1}$
0000, 0	0.1	0.9	0.2	0.8	0.886	**0.114**
0101, 0	0.1	0.9	0.9	0.1	0.837	**0.163**
1111, 1	0.8	0.2	0.9	0.1	**0.396**	0.604

(b)

FIGURE 3.2: Output deviations example [110]: (a) simple circuit with confidence level vectors and (b) propagated output deviations

A simplified method for SFT was proposed in [119], where the circuits s_2 and s_3 are replaced by identical circuits. These circuits are the minimal combinational circuits that for an input pattern within the protected set X_1, exhibit the same output as the original circuit S_1. The protected design can be described in the following form: (*dont care* indicates undefined values).

$$S_1(x) = s_2(x) = s_3(x), \qquad x \in X_1 \tag{3.4}$$

$$(s_2(x) = \textit{dont care}) \wedge (s_3(x) = \textit{dont care}), \; x \notin X_1 \tag{3.5}$$

3.2.2 Probabilistic Fault Model: Output Deviations

The selection of input patterns with high output deviations tends to provide higher error detection capabilities than traditional fault models [112]. The Output deviations are used to rank patterns according to their likelihood of propagating a logic error. There are a few requirements to compute the output deviations of an input pattern. First, a confidence level vector is assigned to each gate in the circuit. The confidence level R_k of a gate G_k with N inputs and one output is a vector with 2^N elements such as: $R_k = (r_k^{00...00}, r_k^{00...01}, r_k^{00...10}, \cdots, r_k^{11...11})$ where each $r_k^{xx...xx}$ denotes the probability that G_k's output is correct for the corresponding input pattern. The actual probability values of the confidence level vectors can be generated from various sources, e.g., inductive fault analysis, layout information or transistor-level failure probabilities. In this work, the probability values obtained by inductive fault analysis shown in [111] are used. However, as it is discussed in [110], deviation-based test patterns may be generated using various sets of confidence level vectors.

Propagation of signal probabilities in the circuit follows the principle shown in [120], with no consideration for signal correlation to reduce computation complexity. The signal probabilities $p_{k,0}$ and $p_{k,1}$ are associated to each net k in the circuit. In the case of a NAND gate G_k with inputs a, b and output z, the propagation of signal probabilities with the confidence level vector r is as follows:

$$p_{z,0} = p_{a,0}p_{b,0}\left(1 - r_k^{00}\right) + p_{a,0}p_{b,1}\left(1 - r_k^{01}\right) + p_{a,1}p_{b,0}\left(1 - r_k^{10}\right) + p_{a,1}p_{b,1}r_k^{11} \quad (3.6)$$

$$p_{z,1} = p_{a,0}p_{b,0}r_k^{00} + p_{a,0}p_{b,1}r_k^{01} + p_{a,1}p_{b,0}r_k^{10} + p_{a,1}p_{b,1}\left(1 - r_k^{11}\right) \quad (3.7)$$

The same principle is applied to compute the signal probabilities for all gate types. For a gate G, let its fault-free output value for the input pattern t_j be d, $d \in (0, 1)$. The output deviation Δ_{Gj} of G for input pattern t_j is defined as $p_{G\bar{d}}$, where $\bar{d}$ is the complement of d ($\bar{d} = 1 - d$). In other words, the output deviation of an input pattern is a measure, unbiased towards any fault model, of the likelihood that the output is incorrect due to a fault in the circuit [110].

Example: Figure 3.2 shows a circuit with a confidence level vector associated with each gate. The table presents three input patterns and their output deviations. The first column shows the input pattern (a, b, c, d), along with the expected fault-free output value z. The next columns show the signal probabilities for both logic '0' and '1' of the two internal nets and the primary output (e, f, z). The output deviation of a pattern is the likelihood that an incorrect value is observed at the output z. Therefore the output deviations (the erroneous behaviour in G_3) for the presented input patterns are: $\Delta_{3,0000} = p_{z,1}$, $\Delta_{3,0101} = p_{z,1}$, $\Delta_{3,1111} = p_{z,0}$. In this example, the input pattern 1111 has the greatest output deviation, with a probability of observing a 0 (the erroneous value) at z of $p_{z,0}$=0.396, thus offering the highest likelihood of detecting an error. •

3.2.3 Workload types

This work considers two workload types based on whether the application is known during design:

- *Uncorrelated workload:* The application of the IC is not known during the design time, such as general purpose processors, and, hence, the in-the-field workload can only be considered uncorrelated. Only protection against the timing-independent errors, such as those induced by stuck-at faults or input bit-flips, can be targeted.
- *Application-specific workload:* The application of the IC is known during the design time, hence some information related to the in-the-field workload might also be available. Therefore, the workload can be protected against both timing-independent and timing-dependent errors, because the input patterns of the IC might be correlated allowing the identification of consecutive susceptible patterns within the workload.

3.2.4 Motivational example

Table 3.1 presents the results of a motivational example that shows how different patterns in a workload may exhibit different susceptibility to errors. Two sets of patterns are selected. The first set is random patterns (rp) and the second set is selected based on the probabilistic fault model of output deviations (pp) [110] for the combinational circuit *pdc* from the LGSynth'91 benchmarks (Table 3.2) . For this experiment, rp and pp sets are generated by gradually increasing the size of the sets from 8 to 1024 patterns. The values of the random patterns presented (*rp* columns), are obtained using the average results of 30 different sets. The first column shows the size of the rp and pp sets. The next columns show the fault coverage (FC) of the rp and the pp, respectively, obtained by fault simulating the circuit against the permanent fault models of stuck-at (SA) and transition faults. These results represent the ratio of faults which affect the operation of the circuit for the examined input patterns. The pp set used to compute the stuck-at fault coverage was obtained by considering an uncorrelated workload, while the pp set used to compute the transition fault coverage was obtained considering an application-specific workload consisting of 20000 patterns. Columns 'Impr.' show the improvement of the pp over the rp. As expected, the pp set present a higher susceptibility of stuck-at faults, which is in the range [51.9% 311.1%], and for transition faults in the range of [69.1% 300%]. The next columns present results from the susceptibility evaluation of the rp and pp sets of patterns against errors induced by input bit-flips and temporary erroneous output transitions. The pp set used to evaluate the error coverage induced by bit-flips was obtained considering an uncorrelated workload, while the pp set used to compute the temporary erroneous output transitions was obtained by considering an application-specific workload. Particularly, input bit-flips are conducted by flipping a single bit at every input pattern of an uncorrelated workload. The values shown are the percentage of bit-flips at the primary inputs that bypassed the logic masking and propagated through the circuit to reach the output. For such errors, the pp set exhibit higher susceptibility [12.0% 216.0%] compared to the rp set. In the case of temporary erroneous output transitions, which are errors that manifest as sporadic missing transitions at the outputs when applying consecutive pattern pairs within an application-specific workload, the pp set exhibits an error coverage improvement in the range of [55.2% 293.8%], compared to the rp set.

patterns	SA FC (%)		Impr. (%)	Tran. FC (%)		Impr. (%)	Bit-flips Susc. (%)		Impr. (%)	EOT Susc. (%)		Impr. (%)
	rp	pp		rp	pp		rp	pp		rp	pp	
8	4.9	14.2	189.4	1.7	7	300.0	0.6	1.9	216.0	0.3	1.0	287.5
16	5.3	19.6	268.6	3.4	11.5	240.4	1.6	3.3	106.3	0.5	2.0	293.8
32	6.5	26.6	311.1	6.3	19.0	202.2	2.6	4.8	84.7	1.4	3.6	161.6
64	9.7	33.7	247.9	10.5	24.6	133.9	5.1	9.4	84.3	2.4	6.9	195.3
128	21.6	42.1	94.7	12.4	32.6	163.2	10.1	16.3	61.4	4.7	14.0	195.0
256	22.4	49.8	122.1	18.5	41.0	121.5	20.8	29.6	42.3	12.3	28.1	127.4
512	37.0	56.2	51.9	24.8	49.0	97.9	45.3	57.3	26.5	23.1	49.7	114.8
1024	39.0	61.8	58.5	33.9	57.3	69.1	83.3	93.5	12.0	53.0	82.2	55.2

Table 3.1: Measured SA fault coverage, transition fault coverage, susceptibility to bit-flips and temporary erroneous output transition between probabilistic and random patterns, for circuit *pdc*

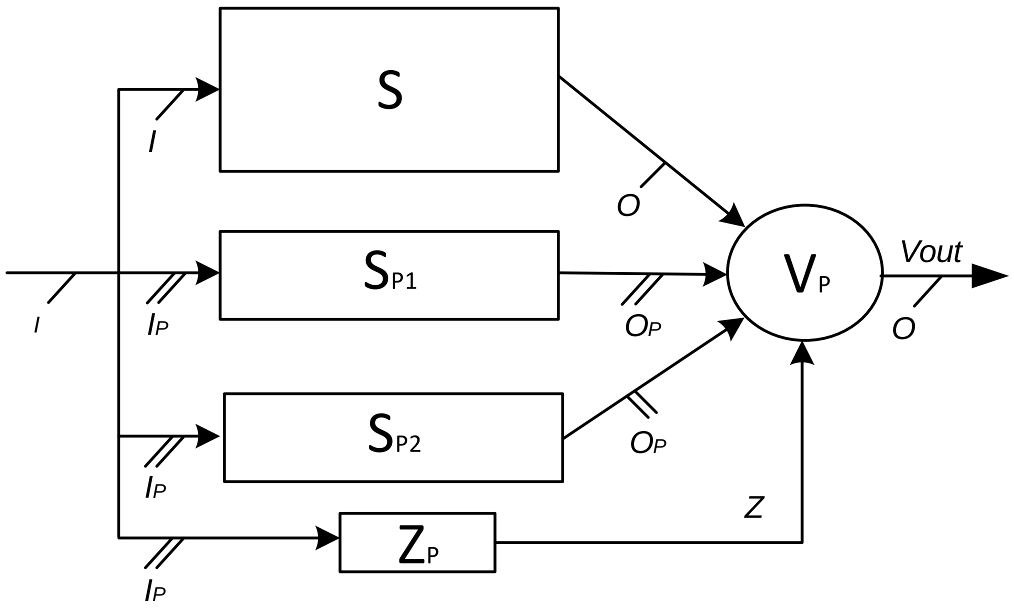


FIGURE 3.3: Proposed Probabilistic Selective Fault Tolerance design

3.3 Proposed Probabilistic Selective Fault Tolerance (PSFT) Design Technique

This section describes the proposed Probabilistic Selective Fault Tolerance (PSFT) design technique.

3.3.1 PSFT Design

The Probabilistic Selective Fault Tolerance (PSFT) design is presented in Figure 3.3. The PSFT design consists of a partial TMR scheme of the original circuit S, two smaller redundant circuits S_P, and a Z_P characteristic function. The latter validates when the inputs of the S_P units belong to the protected input pattern set. Different from the previous SFT design, shown in Figure 3.1, the original circuit S is connected to all the inputs nodes, while the S_P and Z_P units are only connected to the input nodes of the logic cones selected for protection ($I_p < I$, $O_p < O$). A majority voter V_P is used at the outputs of the circuit which operates only when the output is asserted ($Z = 1$). If $Z = 0$, then the voter propagates the outputs of the original circuit S. This functionality is described in equations (3.8) and (3.9).

$$S_{p1}(x) = S_{p2}(x) = S(x) \qquad x \in Z_p \tag{3.8}$$

$$S_p(x) = dont\ care \qquad x \notin Z_p \tag{3.9}$$

3.3.2 Proposed PSFT design flow

Figure 3.4 presents the flow diagram for the proposed PSFT design technique. The number of patterns to protect (N) and the percentage of cones to protect C_p are considered as parameters of the proposed technique. The proposed technique consists of two processes. First, the *logic cone selection*, determines which logic cones of the circuit to protect given the C_p parameter, and produces the Selected Cones list S_c. The next process is the *pattern ranking and selection*, which consists of two different sub-processes depending on the workload type (uncorrelated or application-specific). For uncorrelated workload, a timing-independent ranking is performed on a large number of patterns and the N patterns who exhibit the highest output deviation metric are selected. For application-specific workload, a timing-dependent ranking of consecutive pattern pairs is deployed to select the $N/2$ consecutive pairs that maximize the output deviation metric. Finally, the list of protected patterns is ready and may be synthesized.

3.3.3 Process 1 : Logic cone selection

The C_p (cone percentage) parameter defines the percentage of the cones to be protected. This parameter allows that the largest cones, in which errors are most likely to occur, are prioritized for protection, similarly to the cone selection technique presented in [109]. Initially, the cones are weighted according to their *exclusive size* $|C_{es}|$. The exclusive size of a cone $|C_{es}|$ is the number of cells included in that cone that are not contained in any previously selected cone. The process begins by setting the percentage of selected cones C_{sp} to $1/(\#\ cones)$. The cone with the highest exclusive size $|C_{es}|$ is picked for protection by including it in the selected cones list S_C. The percentage of selected cones P_s is increased by $1/(\#\ cones)$ and the exclusive sizes of the each cone are updated. The process repeats until P_s is higher or equal to the target C_p value. Finally, the selected cones list S_c is passed on to the pattern ranking and selection process.

Example: The logic cone selection process for a small circuit is shown in Figure 3.5. The three logic cones of the circuit have been marked and ranked according to their exclusive size $|C_{es}|$. The cone Z_1 has a $|C_{es}|$ of 8 cells, the cone Z_2 of 4 and the cone Z_3 of 1 cell. Note that the Z_2 cone has an actual size of 7 cells and an exclusive size of 4, because it shares 3 cells with cone Z_1 that have been discarded when calculating their exclusive sizes. The C_p parameter allows a trade-off between area overhead and error coverage. Particularly for this example, when $C_p = 0.3$, the Z_1 cone will be selected. When $C_p = 0.6$, both Z_1 and Z_2 cones are selected. Finally, with a $C_p = 1.0$, all three cones are selected. This trade-off is explored in Section IV. •

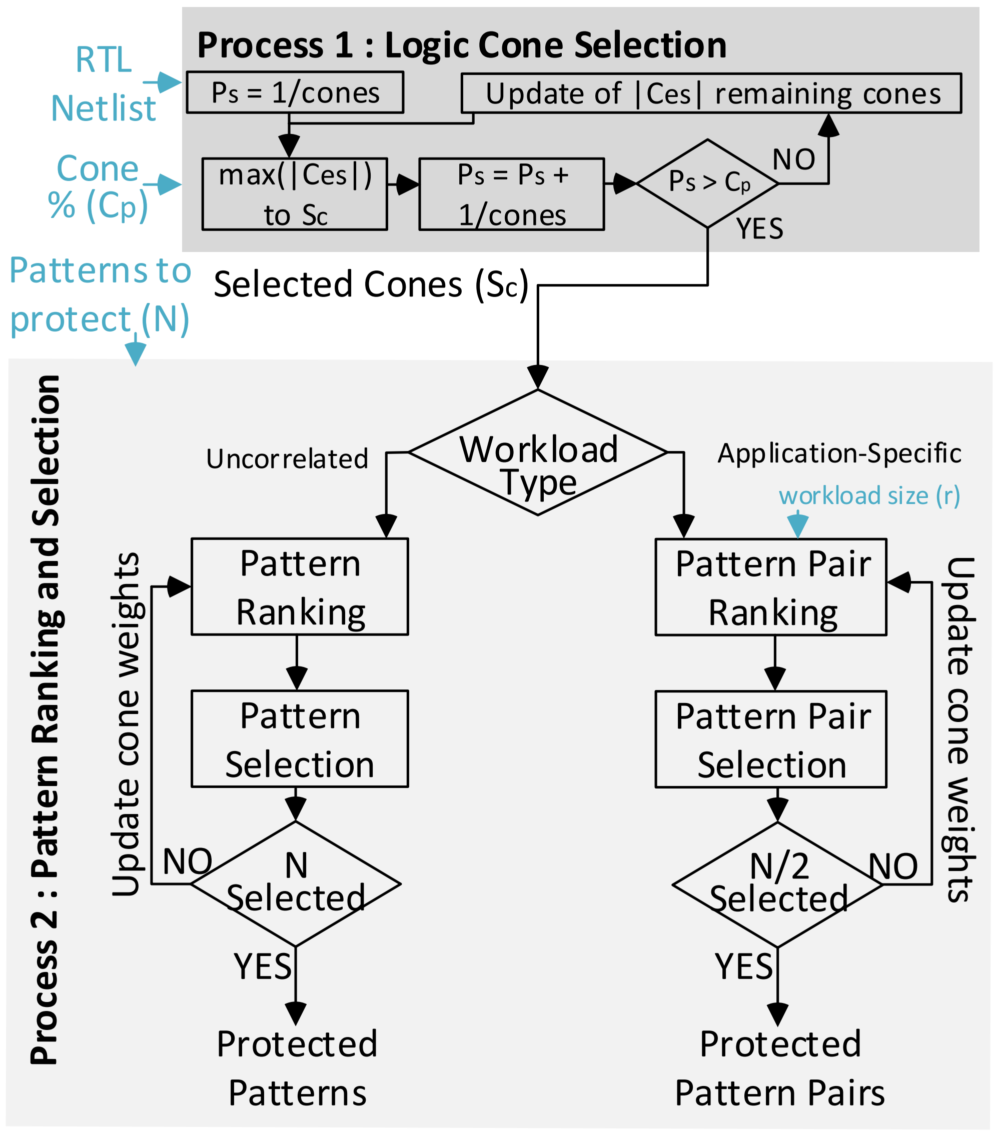


FIGURE 3.4: Proposed design technique flow diagram.

3.3.4 Process 2 : Pattern ranking and pattern selection

First, all cones in the selected cones list S_c are assigned an initial weight $W(c) = 1$. The weights are used by the pattern ranking process to ensure that the selected patterns to protect are not all biased towards a particular cone in the S_c. Next, according to the workload type, either uncorrelated or application specific, different pattern ranking processes are performed.

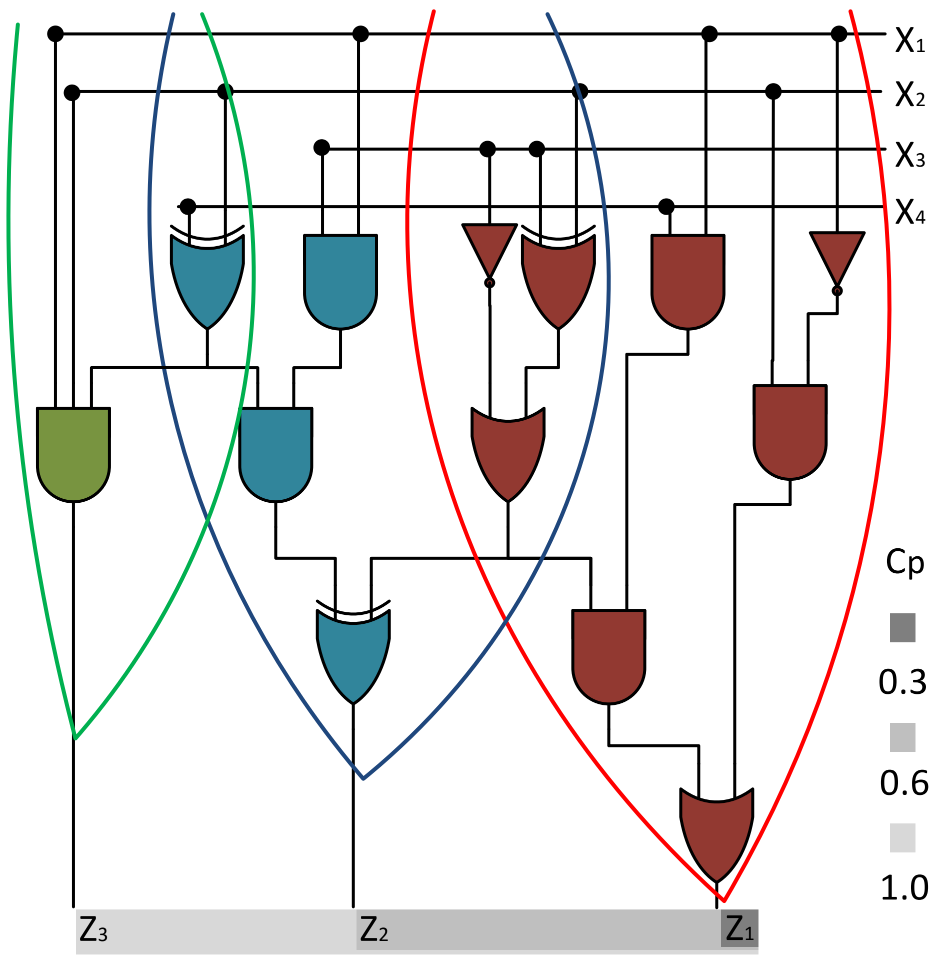


FIGURE 3.5: Example of logic cone selection with different C_p

3.3.4.1 Timing-independent pattern ranking for uncorrelated workload

For an uncorrelated workload, where all input patterns are considered equally likely to occur without a known sequence, a ranking of a large number of patterns for protection of timing-independent faults is performed. Timing-Independent Deviation $TID(p, c)$ is defined as the output deviation-based metric for **ranking input patterns where no sequence of patterns is known**.

Let $TID(p, c)$ be the deviation computed of pattern p for cone $c \in S_c$. Then the selected probabilistic pattern Pp is given by:

$$max(W(c) \cdot TID(p, c)) = W(c) \cdot TID(p, c) \implies Pp = p \tag{3.10}$$

The selected probabilistic pattern Pp shown in (3.10) is the one in which the multiplication of the $TID(p, c)$ with the cone weights $W(c)$, is the maximum after the $TID(p, c)$

of a large number of patterns are computed. The weight of the selected cone c_s in which the maximum $TID(p,c)$ was observed, is updated according to:

$$W(c_s) = W(c_s) \cdot P, \quad \text{with } \ P = 0.9 \tag{3.11}$$

This operation is performed in order to reduce the weight of cones that have been already targeted by the selected probabilistic patterns. This way the selection process avoids selecting patterns that only target the largest cones. The whole ranking process is repeated until all the required N probabilistic patterns have been selected.

3.3.4.2 Timing-dependent pattern ranking for application-specific workload

In the case of an application-specific workload, where a sequence of patterns of size r is expected, a pattern ranking for timing-dependent faults is deployed. Timing-Dependent Deviation $TDD([p_k, p_{k+1}], c)$ is the output deviation based-metric for **ranking consecutive input patterns pairs in a known pattern sequence**.

Let $TDD([p_k, p_{k+1}], c)$ be the deviation computed of consecutive pattern pairs $[p_k, p_{k+1}]$ for cone $c \in S_c$ where $k = (0, 1, 2, ...r)$ and r is the size of the application-specific workload. The selected probabilistic pattern pair $[P_p, P_{p+1}]$ is calculated as follows:

$$max(W(c) \cdot TDD([p_k, p_{k+1}], c)) = W(c) \cdot TID(p, c)$$
$$\implies [P_p, P_{p+1}] = [p_k, p_{k+1}] \tag{3.12}$$

The consecutive pattern pair $[P_p, P_{p+1}]$ shown in (3.12) is the one in which the multiplication of the $TDD([p_k, p_{k+1}], c)$ with the cone weights $W(c)$, is the maximum of all possible consecutive pattern pairs in the application-specific workload. Similarly to the timing-independent ranking, the weight of the cone c_s in which the maximum $TDD([p_k, p_{k+1}], c)$ was observed, is reduced according to (3.11). This process is repeated until $N/2$ consecutive pattern pairs have been selected.

3.4 Experimental Validation

This section evaluates the proposed selective fault tolerance design technique by applying it on a subset of combinational circuits from the LGSynth'91 and ISCAS'85 benchmark suites (Table 3.2). A comparison of computational effort between the proposed technique and statistical fault injection is performed. The simulation setup for this evaluation is presented as well as comparison results against randomly protecting workload. The area overhead of the proposed technique and a trade-off analysis for various number of patterns (N) and cone percentage (C_p) are also presented.

Benchmark	I/O	Gates	Benchmark	I/O	Gates
pdc	16/40	806	ex5	8/63	256
table3	14/14	445	c880	60/26	383
t481	16/1	389	c3540	50/22	1699
c6288	32/32	2406	c7552	207/108	3512

Table 3.2: Benchmark circuit Inputs/Outputs and Gates

3.4.1 Computational Effort Comparison

A comparison is made between the CPU time necessary to compute the most susceptible patterns in an uncorrelated workload of 2000 patterns (Section 2.3) by the proposed technique and by a statistical fault injection (SFI) simulation. The SFI simulation consists of five different stuck-at fault injection campaigns on 50% of all possible fault sites while executing the same uncorrelated workload. The circuit outputs are compared against the error-free case each cycle to determine if an error has occurred. The N patterns that exhibited the highest number of errors across all fault injection campaigns were deemed as the most susceptible for the SFI simulation. The pattern-ranking presented in Section 3.4.1 is deployed on the same workload to find the N most susceptible patterns given by the proposed technique. The experiments were performed on a Linux x64 desktop machine with an Intel Core i7-3770 CPU and 16GB of available RAM.

Table 3.3 shows the CPU runtime of both the SFI simulation and the pattern ranking of the proposed PSFT technique. The second column shows the number of selected patterns N. The third column presents the CPU runtime in seconds required by the SFI to run the five different fault injection campaigns. Note that the time is the same for all N as the simulation must run all the 2000 patterns of the workload to then select the N patterns. Column 4 shows the time required to find the N most susceptible patterns using the proposed technique. Finally, column 5 presents the speed up achieved by the proposed technique compared to the SFI simulation. The time required by the proposed technique increases as the number of patterns increase, although even for 1024 patterns, the proposed technique is still orders of magnitude faster finding the most susceptible patterns in a workload. Additionally, the patterns selected by the proposed technique are not biased towards any particular fault model while those selected by the SFI simulation are biased towards the stuck-at fault model.

3.4.2 Simulation Setup

First, the most susceptible patterns and pattern pairs are synthesized using the ABC synthesis tool [121] into the proposed partial TMR scheme (Figure 3.3) used by the PSFT technique. Figure 3.6 depicts the simulation setup used for the evaluation of the error coverage (EC) achieved by the proposed technique against errors induced by permanent and transient faults. For the permanent faults evaluation, single stuck-at

	N	SFI (s)	PSFT (s)	x Speedup
ex5	4		1.1	14560.2
	16		3.3	4915.9
	64	16293.5	12.1	1344.7
	256		48.6	335.2
	1024		190.6	85.5
t481	4		0.7	24267.9
	16		1.6	10028.5
	64	16469.2	5.4	3029.9
	256		21.4	771.0
	1024		82.4	199.9
pdc	4		1.5	51985.0
	16		3.8	20902.3
	64	78470.0	12.8	6138.1
	256		48.8	1606.8
	1024		194.1	404.2
c3540	4		1.8	20751.4
	16		4.8	7626.7
	64	36796.5	17.5	2108.1
	256		68.6	536.5
	1024		269.7	136.5
c880	4		0.679	6098.6
	16		1.9775	2094.0
	64	4140.9	7.1614	578.2
	256		27.765	149.1
	1024		111.3	37.2
table3	4		1.1379	86972.1
	16		3.3383	29645.4
	64	98964.5	12.245	8082.2
	256		47.747	2072.7
	1024		189.4	522.5
c6288	4		3.0	45000.4
	16		8.6	15575.6
	64	134628.3	31.5	4273.7
	256		131.2	1026.0
	1024		510.3	263.8
c7552	4		3.4	24210.7
	16		10.2	8185.8
	64	83125.7	36.6	2270.5
	256		147.2	564.8
	1024		572.4	145.2

Table 3.3: CPU runtime of SFI and the proposed PSFT technique

(SSA) faults and transition faults are injected and fault simulated using commercial tools in order to obtain the EC of errors *induced by SSA* (**ibSSA EC**) and *induced by transitions* (**ibTran EC**). For the transient faults evaluation, bit-flips are injected at the inputs of the circuit to obtain the EC of errors *induced by bit-flips* (**ibBF EC**). The ibBF EC is computed by injecting (using the fault injection tool described in Appendix A) such random upsets and finding $50K$ events in which a bit-flip on an input pattern propagates an error through the whole circuit and reaches the output of the unprotected circuit (S). The ibBF EC is the percentage of these upsets that are masked by the redundant circuits (S_p) of the PSFT design and therefore are not affecting the protected

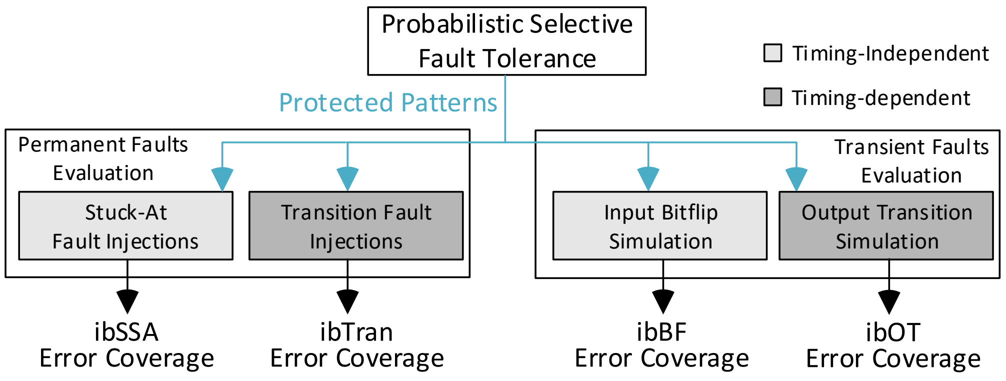

Probabilistic Selective Fault Tolerance
☐ Timing-Independent
▨ Timing-dependent
Protected Patterns
Permanent Faults Evaluation
Stuck-At Fault Injections
Transition Fault Injections
Transient Faults Evaluation
Input Bitflip Simulation
Output Transition Simulation
ibSSA Error Coverage
ibTran Error Coverage
ibBF Error Coverage
ibOT Error Coverage

FIGURE 3.6: Simulation Setup

circuit. Furthermore, the EC of errors *induced by temporary erroneous output transitions* (**ibEOT EC**) is calculated for an application-specific workload of size $r = 20K$. The ibEOT EC is the percentage of transitions at the output occurring while executing such workload that are protected by the selected consecutive pattern pairs set.

3.4.3 Simulation Results

3.4.3.1 Errors Induced by Single Stuck-At Faults

Table 3.4 presents a comparison of the ibSSA EC in circuit S of a random patterns (S_{rp}) set and the ranked probabilistic patterns (S_{pp}) set for a large uncorrelated workload. As shown in Table 3.3, SFI execution times are particularly prohibiting for large workloads, therefore random patterns were used for comparison as they emulate an arbitrary selection of workload with no susceptibility information. The values shown in the S_{rp} column is the average of 30 different random patterns selections. The S_{pp} column shows the ibSSA EC of the ranked probabilistic patterns. The Impr (%) column shows the improvement of the S_{pp} over the S_{rp} calculated as: $Impr\% {=} (S_{pp} - S_{rp})/S_{rp} \cdot 100$. Note that the S_{pp} consistently exhibit a higher ibSSA EC than the S_{rp}. This improvement saturates as the number of patterns N that is protected increases. This is attributed to the increased probability that the random patterns S_{rp} contain highly susceptible patterns.

Figure 3.7 presents the resulting ibSSA EC and area cost of the PSFT design for the circuit *c880*. The results for a various number of protected patterns (N) are shown for a selected cone percentage $C_p = 0.1$. The left axis corresponds to the ibSSA EC and the right axis to the area cost of the proposed PSFT design. The area cost of the proposed PSFT design is the sum of the area costs of the three blocks (S, S_P & Z_P) (Figure 3.3) divided by the size of the original circuit: PSFT Area Cost $= (2 \cdot S_P + Z_P)/S$. For the scope of this work the cost of the voters will be ignored. The average area cost resulting after synthesizing many S_{rp} sets is similar to that of synthesizing the S_{pp} set when

	ex5			pdc		
N	ibSSA EC (%)		Impr (%)	ibSSA EC (%)		Impr (%)
	S_{rp}	$S_{(pp)}$		S_{rp}	S_{pp}	
2	15.5	16.4	5.8	2.3	5.1	122.3
16	47.7	59.3	24.3	5.3	19.6	268.6
128	75.2	79.9	6.3	21.6	42.1	94.7
1024	84.0	91.0	8.4	39.0	61.8	58.5
	c880			table3		
N	ibSSA EC (%)		Impr (%)	ibSSA EC (%)		Impr (%)
	S_{rp}	S_{pp}		S_{rp}	S_{pp}	
2	22.5	39.5	75.8	2.6	6.1	134.4
16	62.5	76.5	22.3	11.2	28.5	155.5
128	84.2	95.7	13.6	33.0	51.5	56.3
1024	95.1	99.8	5.1	74.5	76.9	3.3
	t481			c3540		
N	ibSSA EC (%)		Impr (%)	ibSSA EC (%)		Impr (%)
	S_{rp}	S_{pp}		S_{rp}	S_{pp}	
2	12.4	12.5	0.8	16.1	22.0	36.5
16	23.8	28.1	18.0	45.1	57.7	27.9
128	45.2	47.0	4.0	71.5	86.9	21.6
1024	61.7	62.0	0.5	93.7	97.4	97.4
	c6288			c7552		
N	ibSSA EC (%)		Impr (%)	ibSSA EC (%)		Impr (%)
	S_{rp}	S_{pp}		S_{rp}	S_{pp}	
2	51.5	58.4	13.5	25.9	29.8	15.0
16	89.3	93.9	5.1	64.2	68.1	6.2
128	95.9	99.8	4.1	84.1	86.4	2.7
1024	98.7	100	1.3	89.8	92.1	2.6

Table 3.4: Improvement of Error Coverage (EC) of errors induced by Single Stuck-At faults (ibSSA)

protecting the same number of patterns. Note that power consumption is proportional to the area cost. Similar to the results shown in Table 3.4, the ibSSA EC of the pp is consistently higher than that the one of the rp for all examined N values.

The computation of the ibSSA EC of the PSFT design is calculated by adding the coverage in each of the blocks of the design. The EC in the original circuit S is obtained by the protected patterns (S_{pp}). The coverage of the S_P and Z_P blocks is 100%, as the protected patterns sensitize them fully. The ibSSA EC of the PSFT design is computed as:

$$EC_{PSFT} = \frac{(2 \cdot |S_P| + |Z_P|) + (S_{pp}) \cdot |S|)}{2 \cdot |S_P| + |Z_P| + |S|} \tag{3.13}$$

where $|S|$, $|S_p|$, and $|Z_P|$ are the sizes of the blocks depicted in Figure 3.3 and S_{pp} is the ibSSA EC of the ranked probabilistic patterns.

3.4.3.2 Errors Induced by Transition Faults

Table 3.5 shows results obtained for the transition faults evaluation. The comparison of the ibTRAN EC of the random patterns S_{rp} and the ranked probabilistic patterns S_{pp} is presented. The results are shown in the same format as Table 3.4. Note that the

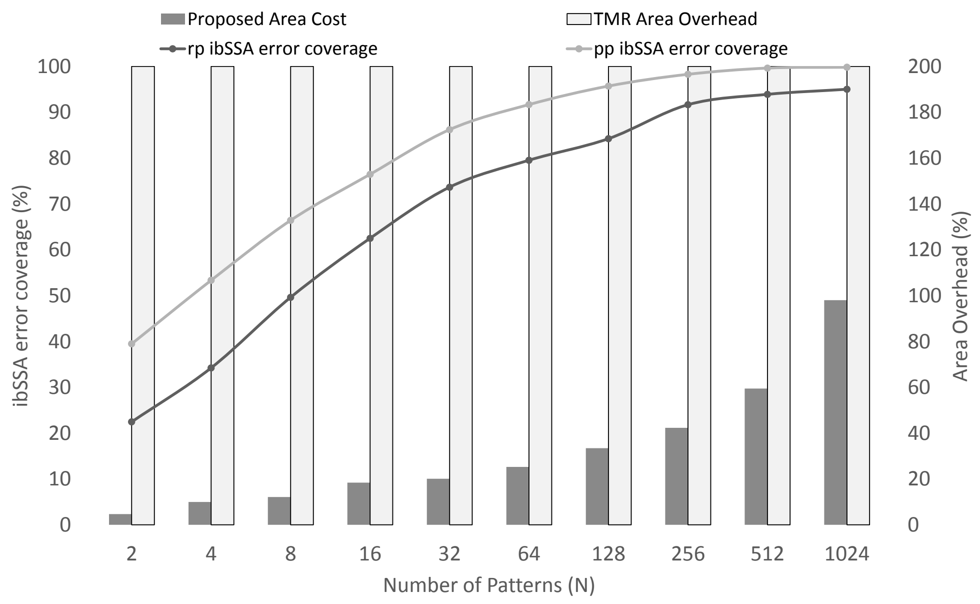


FIGURE 3.7: Area cost of Benchmark *c880* and ibSSA EC for a selected cone percentage $C_p = 0.1$

S_{pp} also exhibits a higher ibTran EC than the S_{rp}, despite transition faults not being targeted specifically by the pattern ranking using the output deviation-based metric.

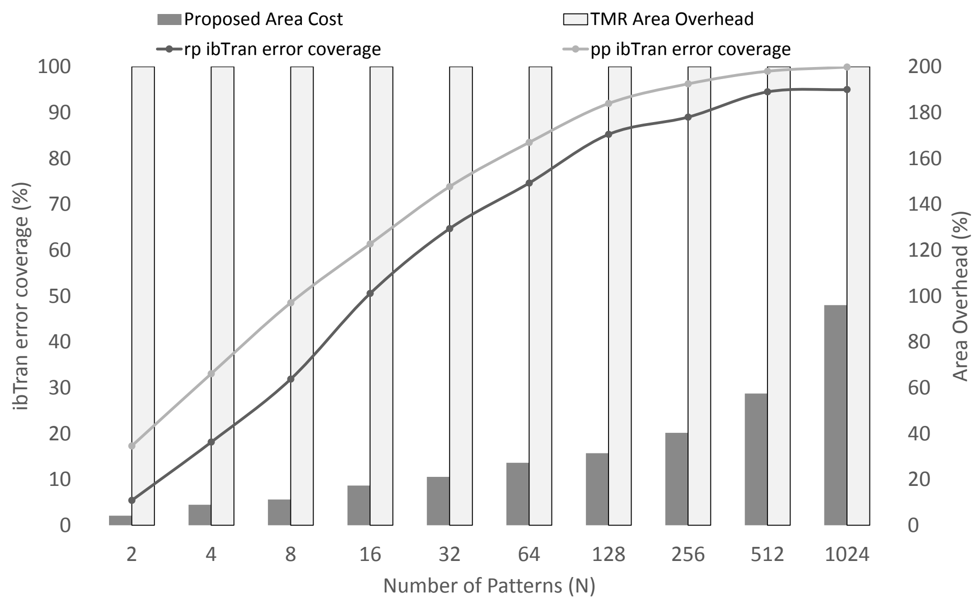


FIGURE 3.8: Area cost of Benchmark *c880* and ibTran EC for a selected cone percentage $C_p = 0.1$

The resulting ibTran EC and area cost of the PSFT design for the circuit *c880* are presented in Figure 3.8. Similarly to Figure 3.7, these results were obtained for a selected cone percentage $C_p = 0.1$, thus protecting only the 10% largest cones in the circuit. In

	ex5			pdc		
N	ibTran EC(%)		Impr (%)	ibTran EC(%)		Impr (%)
	S_{rp}	S_{pp}		S_{rp}	S_{pp}	
2	5.6	9.0	60.7	0.4	1.2	237.1
16	32.3	38.7	19.8	4.4	11.5	162.4
128	56.7	67.1	18.3	10.4	32.6	213.5
1024	81.3	93.5	15.0	33.9	57.3	69.0
	c880			table3		
N	ibTran EC(%)		Impr (%)	ibTran EC(%)		Impr (%)
	S_{rp}	S_{pp}		S_{rp}	S_{pp}	
2	5.4	12.9	138.9	0.8	2.5	219.5
16	47.5	58.1	22.3	3.8	13.5	255.3
128	74.6	91.9	23.2	21.3	24.5	15.0
1024	92.9	99.6	7.2	49.2	53.1	7.9
	t481			c3540		
N	ibTran EC(%)		Impr (%)	ibTran EC(%)		Impr (%)
	S_{rp}	S_{pp}		S_{rp}	S_{pp}	
2	0.1	0.4	300.0	4.8	10.2	112.3
16	1.7	5.6	233.9	31.6	35.9	13.6
128	15.2	18.7	23.2	62.4	68.8	10.3
1024	40.8	44.0	7.9	84.6	93.5	10.5
	c6288			c7552		
N	ibTran EC(%)		Impr (%)	ibTran EC(%)		Impr (%)
	S_{rp}	S_{pp}		S_{rp}	S_{pp}	
2	15.5	30.8	98.9	9.9	15.3	54.7
16	81.5	83.0	1.9	49.3	52.7	6.8
128	97.7	99.2	1.6	79.7	83.3	4.5
1024	99.8	100	0.2	89.1	89.3	0.2

Table 3.5: Improvement of Error Coverage (EC) of errors induced by Transition faults (ibTran)

both Figure 3.7 and 3.8, the EC provided by the S_{pp} is consistently higher than for the S_{rp}, even though neither stuck-at faults nor transition faults were targeted when selecting the protected patterns.

3.4.3.3 Overall Error Coverage Improvements and Area Cost Trade-Off

$C_p = 0.1$, N=8	ibBF EC	ibEOT EC	ibSSA EC (%)				ibTran EC (%)				Area cost (%)			TMR
Benchmark	Impr.(%)	Impr.(%)	S_{pp}	S_{rp}	Impr.(%)	EC_{PSFT}	S_{pp}	S_{rp}	Impr.(%)	EC_{PSFT}	S_P	Z_P	PSFT	Impr.(%)
pdc	71.4	162.5	14.2	4.9	189.8	19.5	7.0	1.75	300.0	12.8	1.6	3.4	6.6	193.4
table3	45.2	183.3	19.0	7.5	153.3	26.1	7.6	1.92	295.3	15.7	2.9	3.8	9.7	190.3
ex5	231.6	133.3	44.5	33.2	34.0	47.6	26.0	21.5	21.2	33.8	5.8	0.9	13.4	186.6
c880	55.1	44.7	63.4	59.8	6.0	67.3	41.8	36.9	13.4	48.1	5.5	9.0	14.5	185.5
t481	47.2	180.0	21.4	18.1	18.2	28.5	2.9	0.4	663.2	11.7	4.7	0.6	10.1	189.9
c3540	222.6	328.6	43.6	39.4	10.5	46.8	24.3	21.4	13.3	31.1	2.9	0.4	6.2	193.8
c6288	243.5	50.0	87.0	83.8	3.8	87.2	68.2	64.2	6.2	71.0	0.8	0.1	1.7	198.3
c7552	143.9	275.0	57.8	55.0	5.05	69.0	41.2	38.5	7.0	46.5	17.1	2.2	36.5	163.5
Average	**132.6**	**169.7**	**43.9**	**37.7**	**52.6**	**49.0**	**27.4**	**23.6**	**165.0**	**33.8**	**5.2**	**2.6**	**12.3**	**187.7**

$C_p = 0.1$, N=32	ibBF EC	ibEOT EC	ibSSA EC (%)				ibTran EC (%)				Area cost (%)			TMR
Benchmark	Impr.(%)	Impr.(%)	S_{pp}	S_{rp}	Impr.(%)	EC_{PSFT}	S_{pp}	S_{rp}	Impr.(%)	EC_{PSFT}	S_P	Z_P	PSFT	Impr.(%)
pdc	75.4	308.7	26.6	6.5	311.1	37.8	19.0	6.3	202.2	31.3	3.7	10.7	18.0	182.0
table3	25.1	257.1	37.0	15.4	140.0	48.3	20.2	6.9	191.5	34.6	7.6	8.8	24.0	176.0
ex5	266.8	230.8	71.0	60.8	16.8	75.9	53.1	50.6	5.0	61.0	10.1	3.8	24.0	176.0
c880	67.6	59.5	86.2	78.0	10.5	88.5	73.8	64.7	14.2	78.2	8.0	4.1	20.2	179.2
t481	24.4	158.3	34.2	30.4	12.5	48.1	7.3	5.2	41.5	15.7	12.7	1.7	27.1	172.9
c3540	138.1	200.0	67.7	59.4	13.9	74.3	49.5	46.8	5.8	54.1	12.2	1.6	26.0	174.0
c6288	79.7	36.4	97.0	95.9	1.18	97.1	92.2	90.8	1.56	92.9	1.3	0.2	2.7	197.3
c7552	88.7	53.8	78.7	75.9	3.7	86.1	67.3	64.9	3.6	70.2	25.5	3.3	54.2	145.8
Average	**95.7**	**163.1**	**67.4**	**52.8**	**63.5**	**69.5**	**47.8**	**42.0**	**58.2**	**54.7**	**10.1**	**4.3**	**24.5**	**175.4**

Table 3.6: Permanent (induced by single stuck-at and transition) and Transient (induced by erroneous output transitions and by bit-flips) EC, area cost and the improvement over TMR of the proposed technique with $C_p = 0.1$

Table 3.6 shows the ibBF EC and the ibOT EC improvement of S_{pp} over S_{rp}, the ibSSA EC, ibTran EC, area cost and TMR area improvement when only the top 10% largest logic cones are selected ($C_p = 0.1$) for 8 and 32 protected patterns. The second column shows the ibBF EC improvement calculated by the input bit-flip simulation applied on an uncorrelated workload, while the third column shows the ibEOT EC improvement of the ranked consecutive pattern pairs of an application-specific workload of size $r = 20K$. The ibSSA and ibTran EC of the S_{pp} and S_{rp} sets and the improvement of S_{pp} over S_{rp} are presented in the next columns. Following is the total ibSSA and ibTran EC of the whole PSFT design, as calculated by Equation 3.13. The area cost of S_P and Z_P blocks (Figure 3.3) as well as the overall area cost of the proposed PSFT design are presented in the next three columns, respectively. The improvement in area cost over TMR is presented in the last column, which is calculated as TMR$_{impr}$ = 200 - area cosf of PSFT.

When 32 patterns are protected, the table shows an average ibBF EC improvement of 95.7% and an average ibOT EC improvement of 163.1% of S_{pp} over S_{rp}, an average ibSSA EC of 69.5% with an average improvement of 63.5% and an average ibTran EC of 54.7% with an average improvement of 58.2%. This results in an overall average improvement of 95.1%. An area cost is observed in the range of 18.0-54.2% for all circuits, which corresponds to a 145.8-182.0% reduction compared to TMR. Note that for circuit *c880* using only 32 susceptible patterns selected with the output deviations-based metric, provides an ibSSA EC of 88.5% and an ibTran EC of 78.2% with an area cost of only 20.2%. The results of circuit *c880* exhibit on average a ibBF EC of 4.47% for S_{rp} and of 7.49% for S_{pp}, an improvement of 67.6%. The logic cones selected with a $C_p = 0.1$ have an input space of 2^{10} (10 inputs), therefore, with just 32 out of 2^{10} patterns ($32/2^{10}$ = 3.13%), the proposed technique can cover 7.49% of bit-flips at the inputs. Circuit *ex5* exhibits a large ibBF EC improvement compared to the other benchmarks due to the small input space (2^8), which allows for a simpler identification of the susceptible patterns in a workload.

When a specific error coverage constraint is set, the size of the ranked probabilistic patterns set (S_{pp}) is consistently smaller than the size of the S_{rp} set. Particularly, for an ibSSA EC of 80%, the S_{pp} is 12% to 63% smaller than the S_{rp} set. Similarly, for an ibTran EC of 70%, the S_{pp} is 13% to 61% smaller than the S_{rp} set. Considering that the same number of random patterns incurs in a similar area cost, these smaller S_{pp} sets achieve a 39.5% average area reduction compared to the required S_{rp} set size to obtain the same error coverage.

The selective fault tolerance techniques presented in [102] and [103] rely on an arbitrary selection of the workload to protect without examining the susceptibility of such workload to either faults or errors. Additionally, their target is to provide protection to a subset of all the possible input patterns of a combinational circuit, without ranking any of those patterns according to their susceptibility to any type of errors. Furthermore,

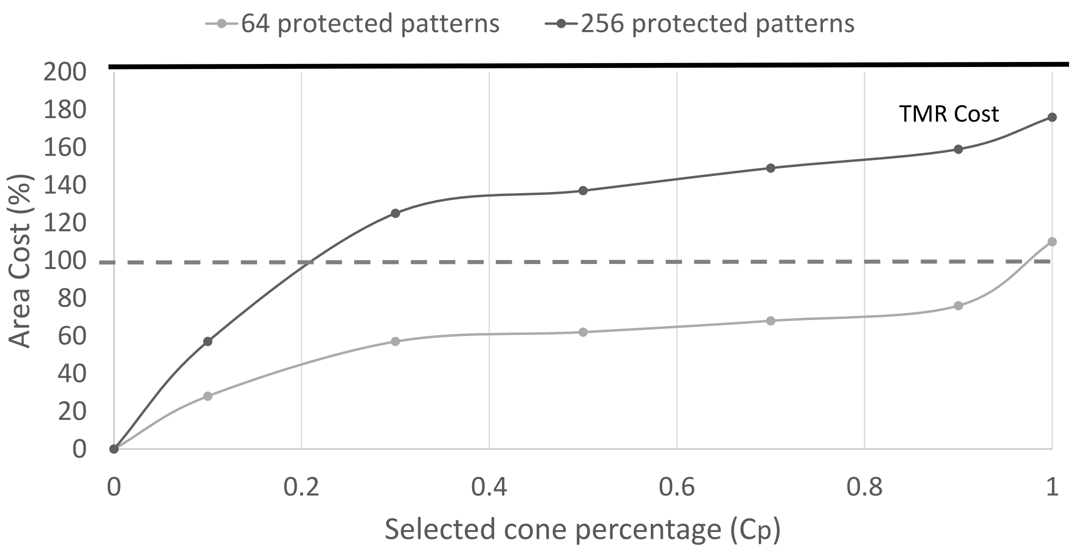


FIGURE 3.9: Area cost of different C_p for benchmark *pdc*

the results shown in [102] and [103] are only for small circuits of the LGSynth'91 benchmark suite. Considering those limitations, a direct comparison between that technique and the proposed PSFT technique is not possible.

Figure 3.9 presents the trade-off between area cost of the PSFT design and different cone percentage C_p values for the circuit *pdc*. The area cost of the protected patterns N = 64 and N = 256 are shown for all C_p values. With a $C_p = 1$, all the logic cones are selected and with $C_p = 0.1$, only the largest 10% of the logic cones in the circuit are chosen. When $C_p = 1$, the PSFT design is synthesized for all cones, which yields a high area cost even for a small number of patterns. This is due to the intrinsic logic sharing present in most circuits which the synthesis tool is unable to simply. It can be seen as expected, that for both N = 64 and N = 256, the area cost decreases until reaching a $C_p = 0$. Note that for both 256 and 64 protected patterns, the area cost for $C_p = 1$ is 176% and 110% respectively, which decreases to 57% and 28% for $C_p = 0.1$.

3.5 Concluding Remarks

This chapter showed that not every workload is equally susceptible to errors induced by permanent or transient faults, which results in some input patterns being less protected by the inherent logic masking of the circuit (Table 3.1). It was proposed to rank this susceptibility to errors in order to protect those patterns that have the most likelihood of propagating errors to the output of the most susceptible logic cones. By combining the technique of Selective Fault Tolerance (Fig. 3.1) and a probabilistic fault model based on the theory of output deviations (Fig. 3.2), a low power selective fault tolerance design technique (Fig. 3.3 & 3.4) was proposed. The proposed technique protects the most susceptible workload using a partial TMR scheme. This technique was evaluated by considering as surrogate error models the timing-independent errors induced by permanent stuck-at faults and transient input bit-flips and the timing-dependent errors induced by permanent transition faults and non-permanent output transition errors on a set of benchmarks (Table 3.2). This technique can achieve a similar error coverage when protecting the same number of patterns with an average 39.7% area/power cost reduction. Furthermore, it can improve by 95.1% on average the achieved error coverage with a similar area/power cost. Particularly, when protecting only the 32 most susceptible patterns, an average error coverage improvement of 63.5% and 58.2% against errors induced by stuck-at and transition faults is achieved, respectively, compared to the case where the same number of patterns are protected without any ranking. Additionally, this technique produces an average error coverage improvement of 163% and 96% against temporary erroneous output transition and errors induced by bit-flips, respectively. These error coverage improvements incur in an area/power cost in the range of 18.0-54.2%, which corresponds to a 145.8-182.0% reduction compared to TMR. Trade-offs between achieved tolerance against permanent (Tables 3.4 and 3.5) and transient (Table 3.6) errors, together with area cost (Fig. 3.9) are also presented. The protection of the most susceptible workload through a probabilistic fault model that is unbiased towards any type of fault, ensures that the fault tolerance against any type of errors is enriched. Therefore, the usage of output deviations to determine the most susceptible workload in an application may assist in enhancing circuit reliability beyond the scope of a partial TMR scheme.

Chapter 4

Online Monitoring of Erroneous Behaviour in the Field

Concurrent error detection (CED) techniques and built-in-self-testing (BIST) can be used to monitor if errors or faults appear in-the-field (Section 2.3). However, traditional implementations of these techniques are often expensive or have an impact on the performance of cost constrained embedded systems. CED techniques target an immediate detection of errors but incur in high area costs. BIST mechanisms achieve a very coverage of faults appearing in the system, but they are unable to perform concurrently to normal operation. The probabilistic selective fault tolerance technique presented in Chapter 3 protects the workload most susceptible to errors. In that chapter, it was shown that not all faults propagate the errors to the outputs. This knowledge can be useful to reduce the cost of monitoring erroneous behaviour of cost constrained systems in-the-field, as only the faults that propagate errors are of concern. Furthermore, devices in cost constrained embedded systems are used for a variety of IoT applications, such as geo-monitoring, parking sensors and surveillance. Such applications may tolerate few errors and may not be constrained by a strict error detection latency requirement. However, with the increasing appearance of intermittent and permanent faults (Section 2.2) in-the-field [122], [123], devices that exhibit systematic erroneous behaviour must be eventually identified and replaced.

This chapter introduces the concept of online signal probabilities at the outputs, based on which a novel low cost probabilistic online error monitoring technique is presented. The technique produces an alarm signal when systematic erroneous behaviour has occurred over a pre-defined time interval [124]. Particularly, this technique monitors the signal probabilities at the outputs of the most vulnerable logic cones of a circuit concurrently to its normal operation. To detect systematic erroneous behaviour, the collected data is compared on-chip against the signature of error-free behaviour. This technique is applied to a set of the EPFL and ISCAS benchmarks and results demonstrate that it is capable of detecting errors caused by intermittent faults with a very low area cost.

This chapter is organized as follows: Section 4.2 proposes the analysis of the online signal probabilities. Section 4.3 presents the proposed on-chip monitoring architectures. Section 4.4 presents the results of the proposed technique followed by the conclusions in Section 4.5.

4.1 Background

Intermittent faults at circuits manufactured using Very Deep Sub-Micron (VDSM) technologies manifest as bursts of errors that repeat periodically at the same places [53], [54], [122], [125], [126], causing the circuit to exhibit systematic erroneous behaviour (SEB). These faults may be activated in-the-field by permanent defects that escaped manufacturing testing [123], by systematic violation of power or thermal constraints or by aging [31], [122], [123]. Devices in the field that exhibit SEB must be identified and maintained.

The maintenance of electronic devices used in low-power IoT applications requires often physical access which might be impractical and has to be planned in advance [127]. Thus, the maintainability of those devices can be assisted by monitoring their behaviour in-the-field. As a result, a low cost solution for monitoring SEB online is required.

Concurrent error detection (CED) techniques and built-in-self-testing (BIST) may be used to monitor SEB. CED techniques using duplication with comparison are applicable to any circuit, and detect almost 100% of single errors with a low error detection latency, as they target an immediate detection of errors as they occur [46], but incur an area and power overhead of more than 100% [128]. CED techniques using error detecting codes achieve a lower error coverage, but with less overhead compared to CED using duplication. However, they may have an impact on system performance and are traditionally used for memories or control logic [59], [95], [129]. On the other hand, built-in-self-test (BIST) mechanisms are tailored specifically for the circuits they test [74], [79], and offer a low cost solution for testing the device in-the-field. However, these techniques may be unable to perform the test concurrently to normal operation, thus SEB caused by intermittent faults appearing only in normal mode will not be detected.

Many applications for low power embedded devices such as geo-monitoring, parking sensors, or surveillance, may tolerate some errors [130]. Such devices are not constrained by a strict error detection latency requirement, thus detecting errors immediately as they occur may not be required, as long as they continue to offer their intended service. Additionally, the high fault coverage achieved by BIST mechanisms may be unnecessary, as not all faults propagate errors to the output during normal operation. However, being able to quantify the amount of erroneous behaviour exhibited by each device, would be beneficial for maintainability purposes.

In this chapter, an analysis of the online signal probabilities at the outputs of a circuit when the latter exhibits systematic erroneous behaviour is presented, and a novel low cost probabilistic online monitoring technique is proposed. The proposed technique may be used to detect SEB in circuits used in cost constrained error-tolerant applications where zero or near-zero error detection latency is not a priority. The logic cones of circuits are monitored by a novel low power and area monitoring architecture with no impact on system performance. The monitors consist of on-chip counters for collecting signal probability information concurrently to normal operation, and control logic to produce an alarm signal if SEB has been detected. To evaluate the SEB detection capabilities of this technique, injections of multiple intermittent bit-flips and intermittent stuck-at faults [122], [123] were applied on a set of the EPFL and the ISCAS combinational benchmark (Using the fault simulation tool detailed in Appendix A).

4.2 Analysis of Systematic Erroneous Behaviour using Online Signal Probabilities

This section proposes an analysis of the online signal probabilities and how they can be used to determine if a circuit has exhibited systematic erroneous behaviour (SEB) at the output of its logic cones. Fault simulation is applied to an example circuit as a proof of concept.

4.2.1 Analysis

Signal probability is defined as the probability of a signal to be a logic 1. Figure 4.1 presents the concept of online signal probabilities. The set of input patterns are referred to as the workload and the number of input patterns in a workload as workload size, denoted by S. During an error-free normal operation, the online signal probabilities at a given node may vary depending on the workload. The smaller the size of the workload, the higher the variation of the online signal probability. As the S increases, the variation of the online signal probability at the output decreases and it starts to converge. We refer to the value to which the signal probability converges the *mean signal probability*, denoted by M_{sp}. The variation of the signal probability during an error-free operation is referred to as *signature window* (w), with W_{max} and W_{min} as the *upper and lower bounds* respectively. The expected M_{sp} and the W_{max} and W_{min} signature window bounds are dependent on the input signal probabilities.

Systematic erroneous behaviour (SEB) is defined as the event in which, for a particular workload size S, systematic errors occur at a high enough rate, that the online signal

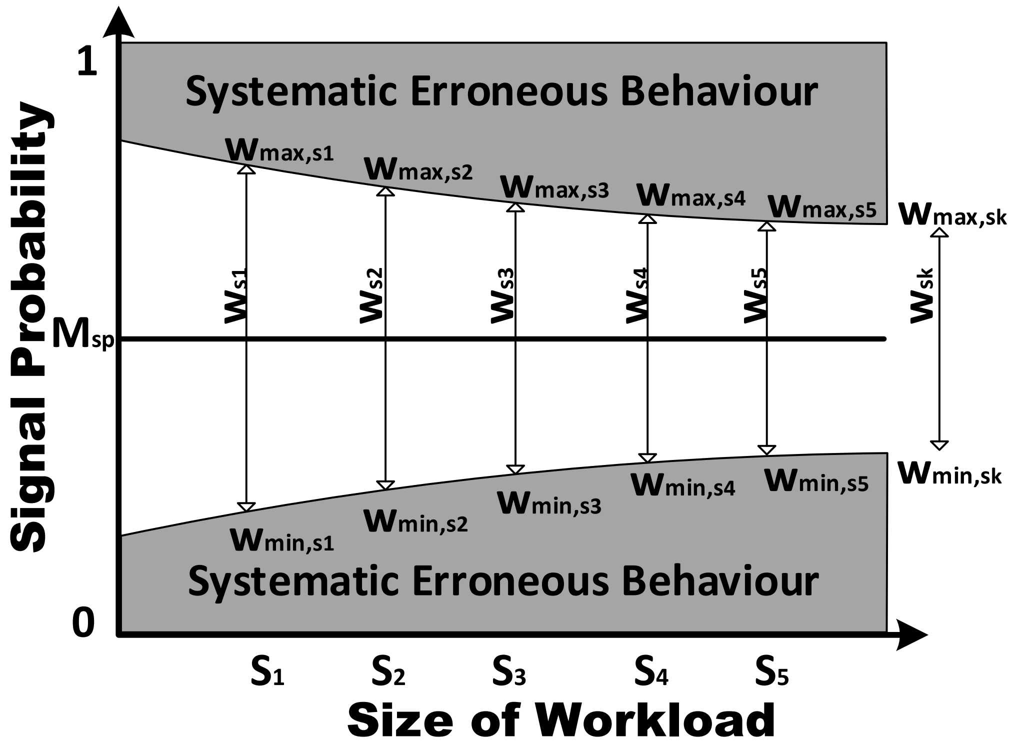


Figure 4.1: Online signal probabilities

probability of an output falls outside the signature window w. SEB may occur in-the-field due to intermittent faults caused by defects escaping manufacturing testing, wearout and aging[122]. Such faults may manifest periodically as *multiple bit-flips* or exhibit a behaviour similar to *permanent faults* under specific operating conditions [123]. When an intermittent fault is activated, it may generate enough errors that the online signal probability at the output falls outside the signature window bounds (lower than the W_{min} or higher than the W_{max}).

In the presence of a fault, for a given input pattern, an error is considered to have occurred only when the output of the circuit is different than that of the error-free case. That is:

$$error = \begin{cases} 1 & \left[o_k^{if}, p_k\right] \neq \left[o_k^{ff}, p_k\right] \\ 0 & \left[o_k^{if}, p_k\right] = \left[o_k^{ff}, p_k\right] \end{cases} \tag{4.1}$$

Where o_k^{if} is the output when a fault is present, o_k^{ff} is the output of the error-free case and p_k is the input pattern.

In application-specific ICs, where the workload might be known during design time, the signal probabilities of the workload tend to be *biased* towards the application. This causes the input patterns to be heavily correlated and the expected behaviour to be

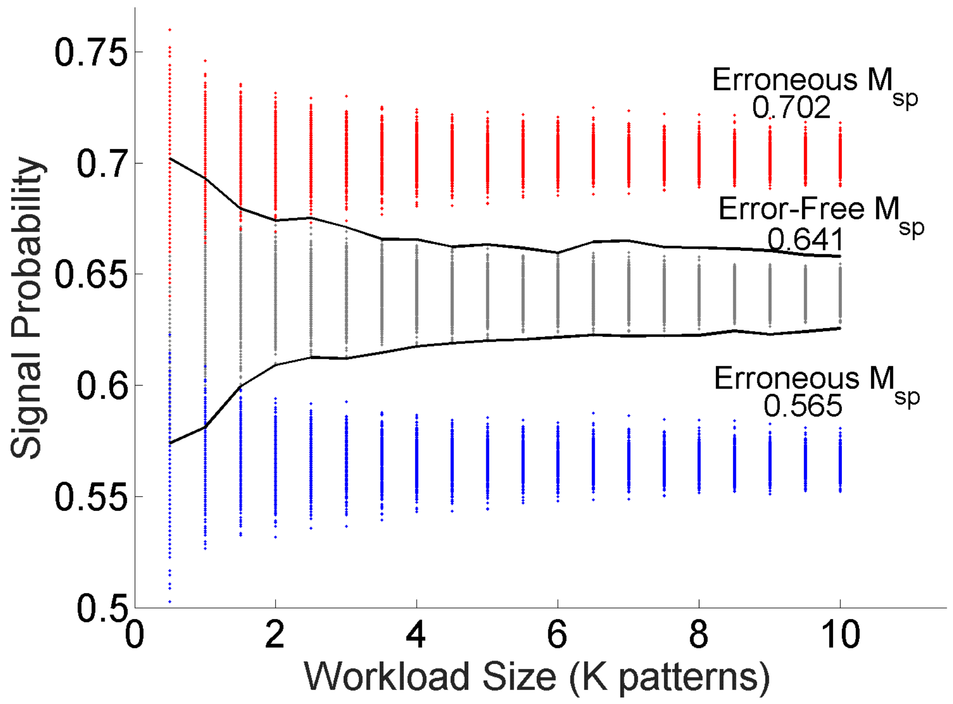


FIGURE 4.2: Online signal probabilities of 1000 different workloads for circuit t481

known. In such cases, the M_{sp} is known and the width of w might be small. In the case of general purpose devices, the workload is unknown during design time as it may vary substantially in-the-field and its patterns appear uncorrelated. If a workload is unknown, all its input patterns are considered to be random and equally likely to occur. The workload is *unbiased* towards a particular application, which makes it necessary to profile the signal probabilities to compute the M_{sp} and w. The analysis of online signal probabilities described in Figure 4.1 is applicable for either a biased or unbiased workload.

4.2.2 Example: LGSynth'91 Benchmark t481

The t481 circuit of the LGSynth'91 benchmarks consists of a single logic cone of 388 gates with 16 inputs and 1 output. The error-free (grey) and erroneous (red and blue) online signal probabilities of 1000 different unbiased workloads, each of them consisting of 10K input patterns, are shown in Figure 4.2. The erroneous online signal probabilities where produced after the insertion of a single stuck-at fault for each case. The mean online signal probability M_{sp} of the error-free case is 0.641. As expected, the variation of the online signal probability decreases as the workload size increases, and when the circuits exhibits SEB then the online signal probabilities converge to different than the error-free M_{sp} values. Note that for small workload sizes, there is an overlap of the error-free

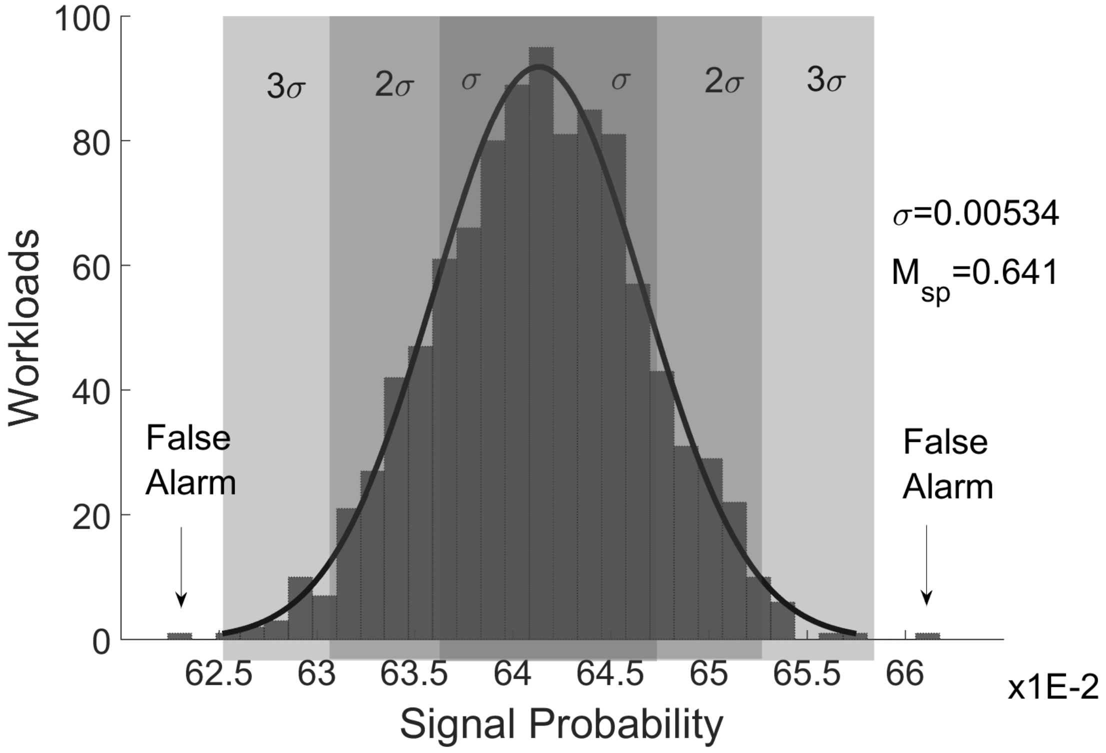


FIGURE 4.3: Histogram of error-free online signal probabilities of circuit t481

and erroneous online signal probabilities, making it impossible to clearly differentiate error-free behaviour from SEB.

4.2.2.1 Probability of False Alarms

As evidenced by Figure 4.3, the online signal probability of an error-free execution of a workload follows a normal distribution, which can be modelled. By computing the mean value and the standard deviation σ of the normal distribution, we can select appropriately the signature window so as to avoid false alarms. For example, approximately 99.7% of the error-free online signal probabilities are within a signature window w=[$M_{sp} \pm 3\sigma$], 95.5% are within a w=[$M_{sp} \pm 2\sigma$], and 68.3% are within a w=[$M_{sp} \pm \sigma$]. Thus, the probability of having a false alarm is a function of the selected window, which corresponds to 0.3%, 4.5% and 31.7% for the windows w=[$M_{sp} \pm 3, 2, 1\sigma$] respectively.

4.2.2.2 Signal Probabilities in the Presence of Faults

Figure 4.4 presents the histogram of online signal probabilities for circuit t481 after performing a single stuck-at (SSA) fault injection simulation on all possible fault sites. The permanent fault model of SSA is used to emulate the SEB that may occur in-the-field, given that intermittent faults behave similarly to permanent faults, under specific operating conditions [122]. The horizontal axis shows the online signal probabilities

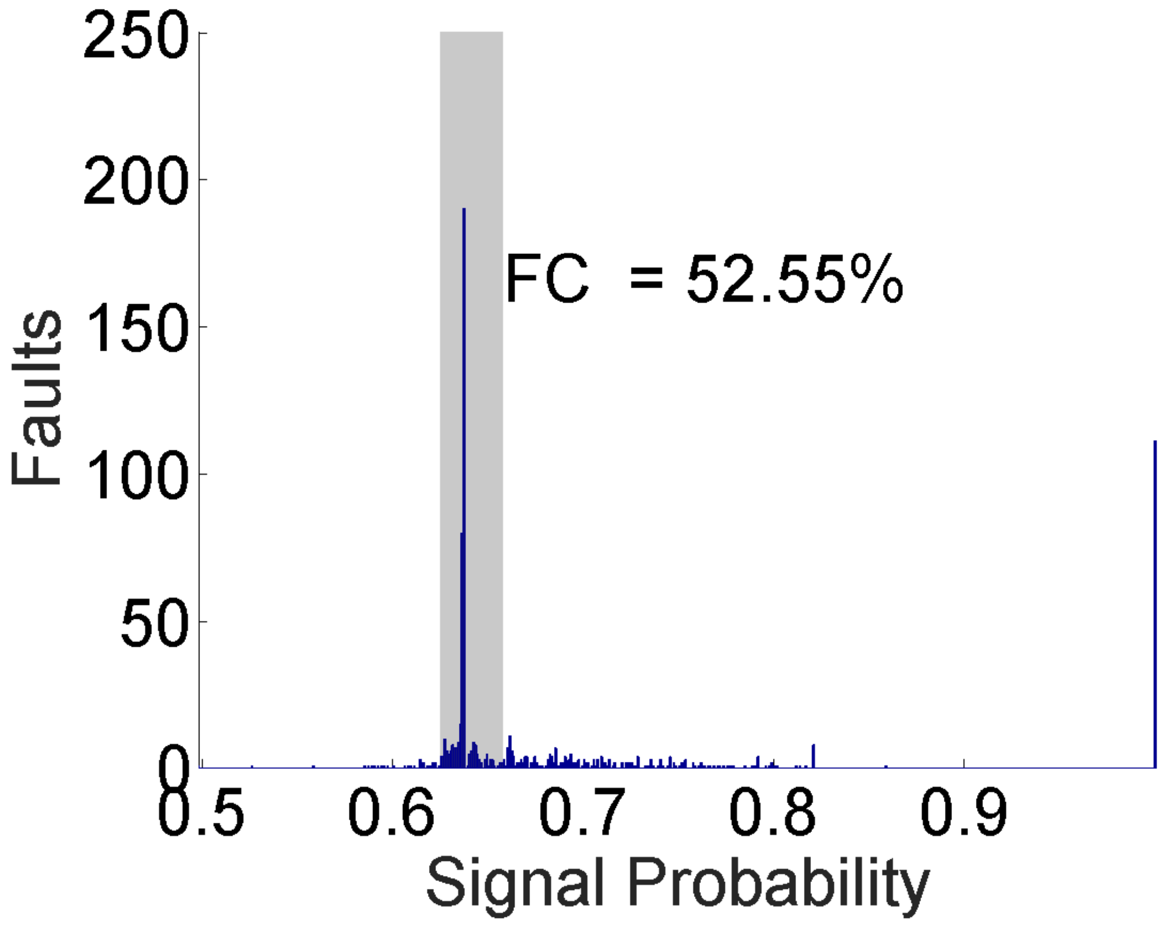


Figure 4.4: Online signal probabilities of stuck-at faults for circuit t481 with $w = M_{sp} \pm 3\sigma$.

while the vertical axis shows the number of injected faults that exhibit the corresponding online signal probability. For this example, an unbiased workload of size $S = 5000$ is used and the signature window is defined as $w = [M_{sp} \pm 3\sigma]$, with $\sigma = 0.00549$. These window bounds provide a probability of having a false alarm of 0.3%. The fault coverage of 52.55% is calculated by counting all the SSA faults that cause the online signal probability to fall outside the signature window. Not all faults occurring during normal operation bypass the inherent logic masking of a circuit [104]. When an intermittent fault is masked, it has a small effect in the online signal probabilities since few errors are propagated. That fault coverage indicates that 47.45% of the possible SSA faults have a small impact on the online signal probability.

4.2.2.3 Signal Probabilities and Errors

The histogram of online signal probabilities and of the errors propagated to the output by the injected stuck-at faults for circuit t481 is presented in Figure 4.5. The horizontal

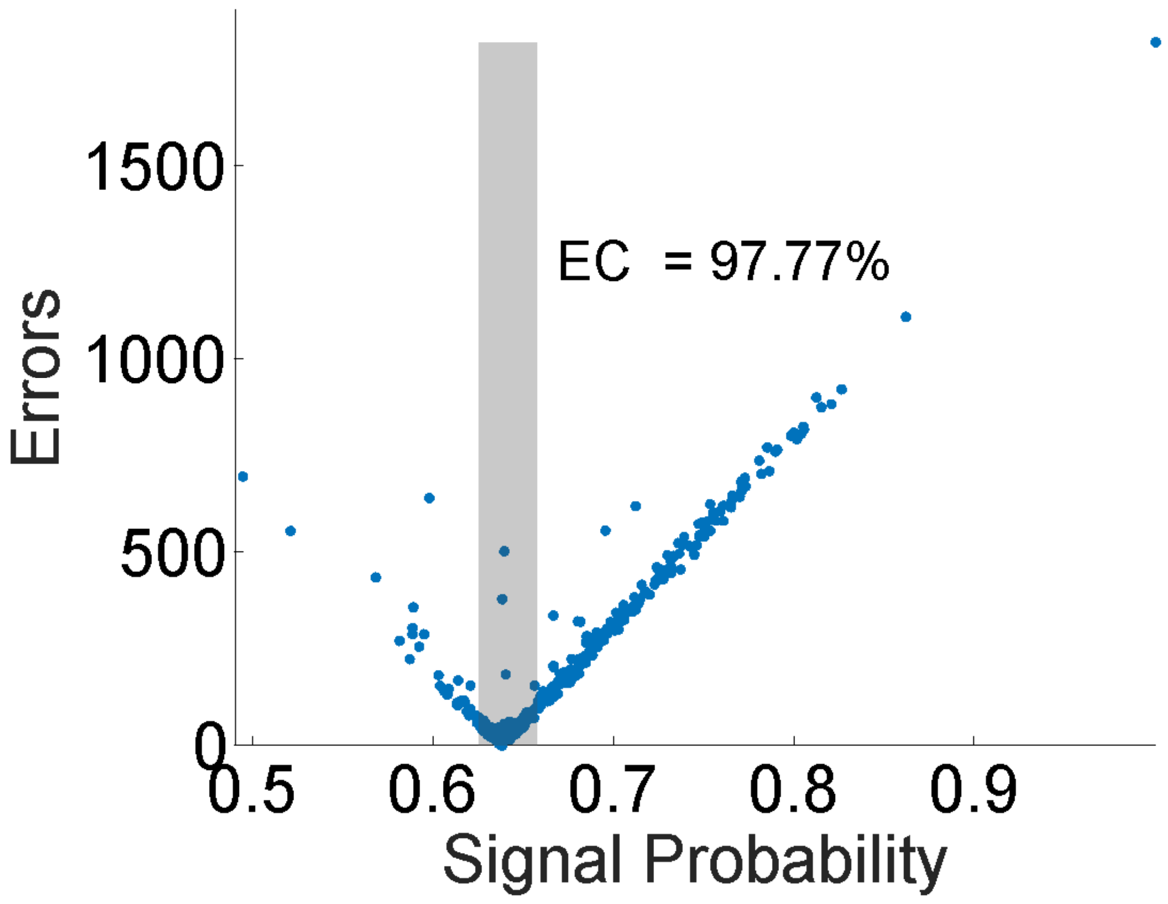


FIGURE 4.5: Online signal probabilities of propagated errors for circuit t481 with w $=M_{sp} \pm 3\sigma$.

axis shows the signal probabilities while the vertical axis shows the number of errors at the output as defined in Equation 4.1. Each dot represents the number of errors observed for each of the injected faults. During the execution of the 5000 patterns, each SSA fault generates a different amount of errors at the output. Faults closer to the output produce more errors due to being less protected by the inherent logic masking of the circuit. The error coverage is defined as the number of errors produced by each injected fault and whose online signal probability falls outside the signature window, divided by the total number of errors. The error coverage for this circuit is 97.77%, which implies that most errors observed at the output are associated to a signal probability that deviates significantly from the mean signal probability M_{sp}. In contrast, only 2.23% of the errors fall within the signature window w, indicating that few errors are propagated to the output when the signal probability is inside the signature window w.

Figure 4.6 presents the error coverage for different workload sizes with w=[$M_{sp} \pm 3\sigma$].

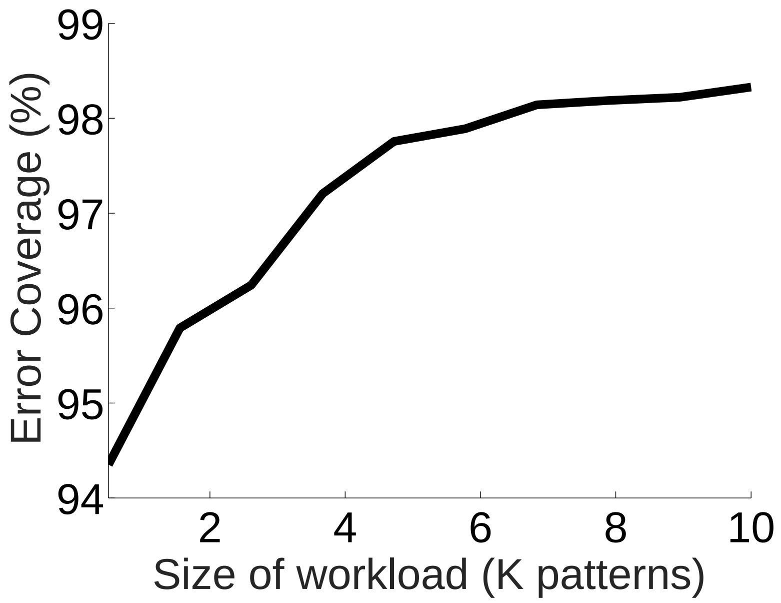


Figure 4.6: Error Coverage for different workload sizes.

The error coverage starts to converge at a workload size of 5000 patterns. Larger workload sizes require longer monitoring times incurring in higher error detection latency, therefore, increasing the monitoring time beyond the convergence point might not provide a significant advantage.

4.3 On-Chip Monitoring Architecture

This section presents the proposed monitoring architecture of online signal probabilities. The design flow to synthesize the monitors is described and two different architectures are proposed.

4.3.1 Monitoring architecture design flow

Figure 4.7 presents the proposed design flow to synthesize the monitors used by the proposed technique. The process of workload profiling is performed depending whether the workload is biased or unbiased. For a biased workload, the correlation and variations of the input patterns are known, which makes the mean signal probability M_{sp} and the signature window bounds W_{min} and W_{max} also biased. For an unbiased workload, where its all input patterns are uncorrelated and considered to be equally likely to occur, an error-free simulation of a large number of unbiased workloads is required to compute the M_{sp}, as presented in the example for circuit t481 of Section 4.2.2. The workload size S is determined once the online signal probabilities have converged. The signature window bounds W_{min}, W_{max} may be set to $M_{sp} \pm 3\sigma$ for a 0.3% probability of having

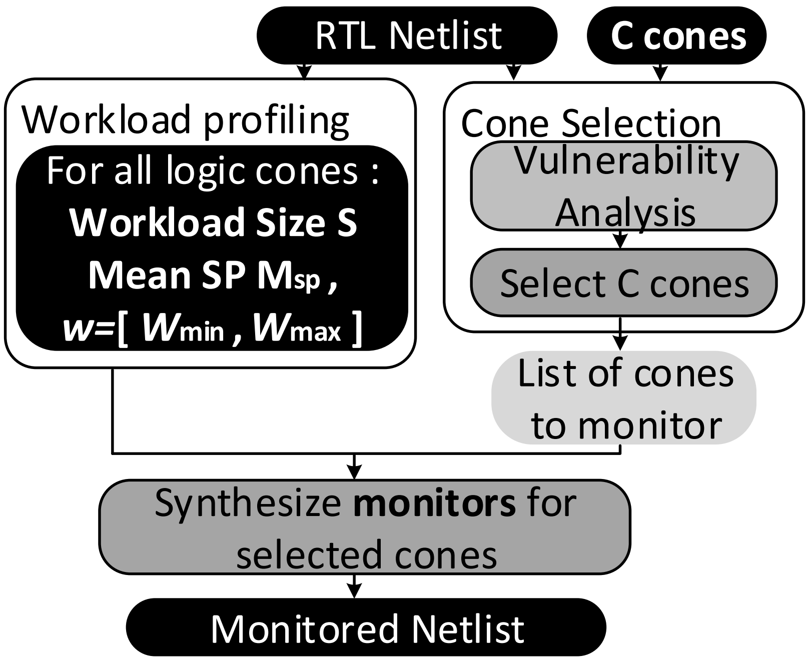


Figure 4.7: Monitoring technique design flow

a false alarm. However, the signature window can be narrower, which increases error coverage but increases the probability of having false alarms. This trade-off is explored in Section 4.4.

A logic cone selection process is necessary to determine which are the C cones to monitor. Logic cones of any size and any number of inputs that are bounded by either primary inputs and outputs (PI/PO), or by sequential elements (SE) may be selected. The simplest cone selection process consists of selecting the C cones that exhibit the highest number of errors, although this selection may be based on different vulnerability analysis methodologies [94], [109], [131].

4.3.2 Monitoring architecture designs

Two monitoring architectures are proposed. A single counter design which provides a lower area cost but a higher monitoring time, and a multiple counter design, that allows to monitor multiple cones at the same time but with a higher area cost. Figures 4.8 and 4.9 present each of the designs.

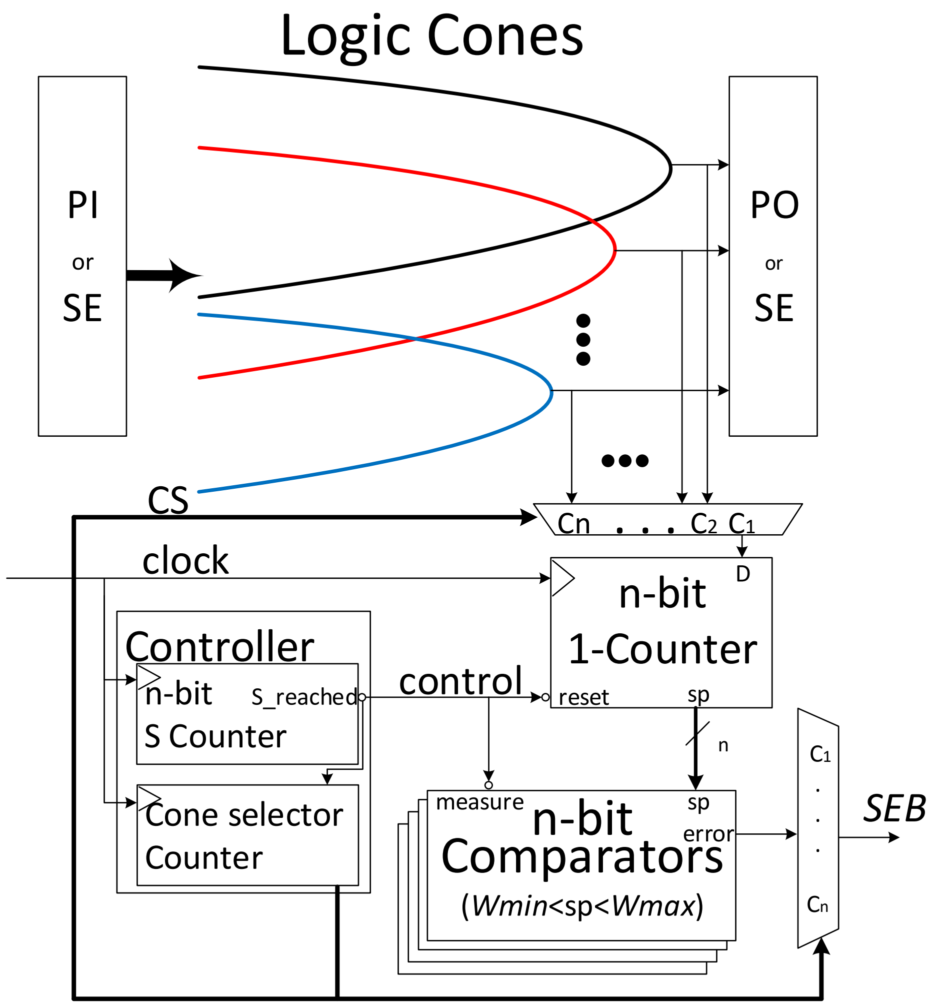


FIGURE 4.8: Single counter design

4.3.2.1 Single counter design

The single counter design (Fig. 4.8) consists of a *n-bit counter* that increases on the rising edge of the clock when the input D is asserted, effectively counting the number of logic 1's. The *n-bit comparators* will assert the *SEB* output if the signal probability *sp* coming from the counter is outside the $Wmin$ and $Wmax$ values when the inverted *measure* input is deasserted. The measure input is deasserted when the S *counter* in the controller has counted S clock cycles. This signal also increases the *Cone selector counter*, whose output *CS* selects which cone to monitor in a round-robin fashion. If the *SEB* output is 0, then no SEB has been detected, otherwise if it is 1, then SEB has been occurring. The value of n is determined by the workload size S according to (4.2). That will result in the minimum n required to count up to S.

$$n = \lceil log_2(S) \rceil \tag{4.2}$$

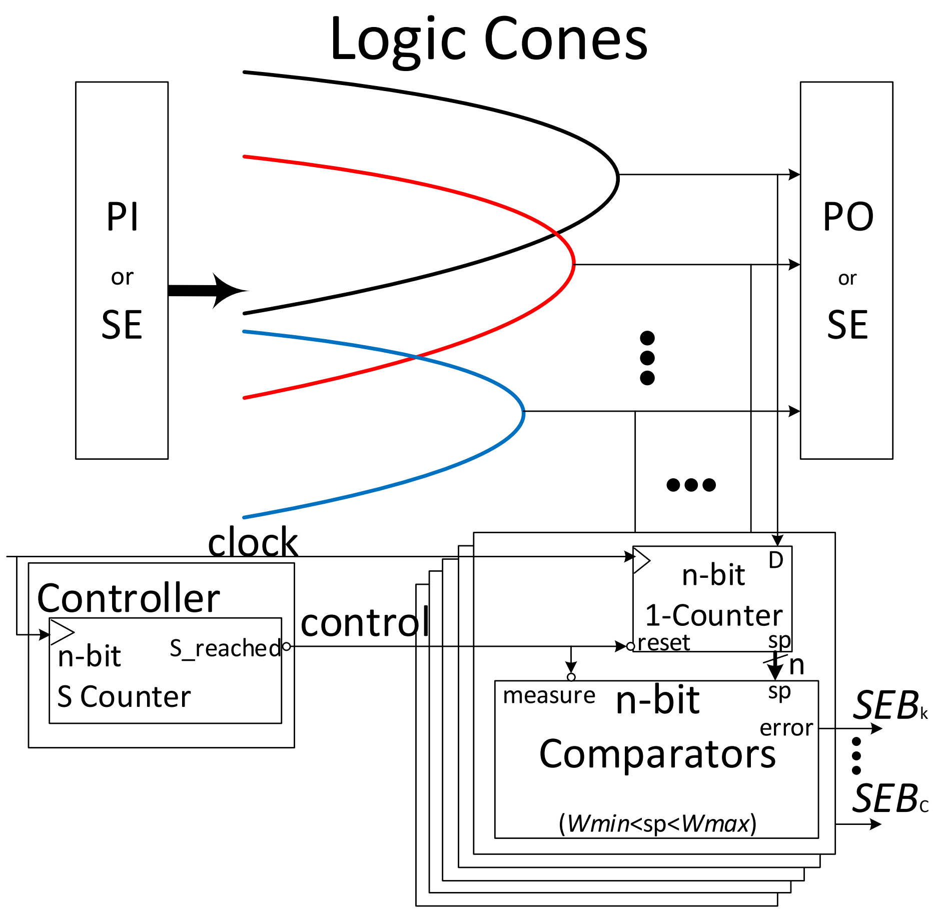


FIGURE 4.9: Multiple counter design

4.3.2.2 Multiple counter design

The multiple counter design (Fig. 4.9) allows to monitor all the cones simultaneously. It consists of an *n-bit counter* and *n-bit comparator* for each of the monitored cones, in addition of a controller which consists of a single *S counter* which counts up to S. All the *SEB* outputs are asserted or de-asserted at the same time when the *measure* input signal of the comparators is de-asserted, producing a simultaneous response for all monitored cones.

Note that a single counter incurs a lower area cost compared to a multiple counter architecture, however, such design is only able to monitor SEB for a single cone at a time, increasing error detection latency for the other cones. On the other hand, the multiple counter architecture would allow to monitor all the selected cones at the same time, reducing monitoring time and error detection latency, but increasing the area cost. Both the single and multiple counter designs may be clock or power gated, enabling the

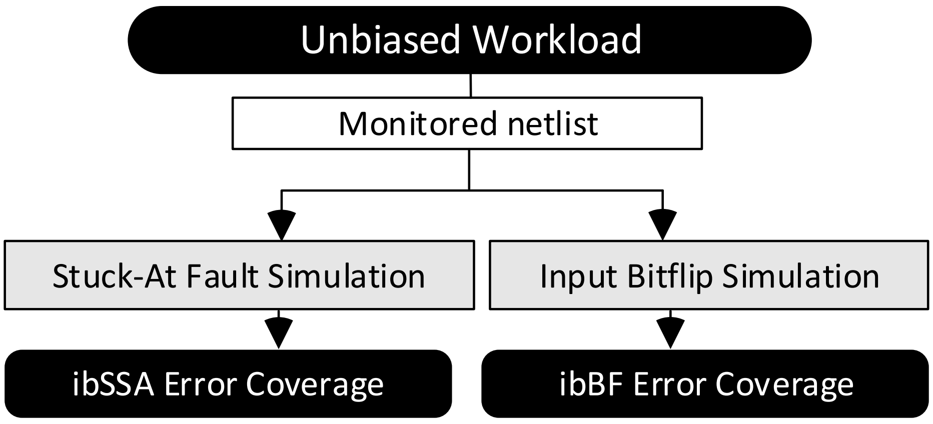


Figure 4.10: Simulation Setup

monitors only when online monitoring is required by a host application or embedded operative system.

4.4 Simulation Results

This section presents the results of the online signal probabilities monitoring technique to a subset of the ISCAS and EPFL benchmarks [132]. An analysis of the trade-off between error coverage and area cost is presented and the along with a discussion on the impact of the signature window width on error coverage.

4.4.1 Simulation setup

The simulation setup is presented in Figure 4.10. The evaluation of this technique was performed for errors induced by stuck-at faults and multiple bit-flips, as these error models produce a behaviour similar to that of long duration intermittent faults occurring in-the-field [122], [123]. Unbiased workloads of different sizes of uncorrelated random patterns were applied during simulations. Multiple bit-flips are injected to emulate upsets in sequential elements at the inputs of the monitored logic cones. Errors at the output are considered to be those where the bit-flip bypasses the inherent logic masking of the cone from the input to the output. These errors are used to compute the error coverage (EC) of errors induced by bit-flips (ibBF). Single stuck-at injection simulation of all possible faults sites is performed to calculate the EC of errors induced by single stuck-at faults (ibSSA). The cones selected to monitor were those that exhibited the highest number of errors, however, as mentioned in section 4.3, this selection may be based on different vulnerability analyses.

Benchmark	Logic Cones	Circuit gates	Workload Size S / EDL	Monitored Cones C	ibBF EC (%) of selected cones						ibSSA EC (%)				Area Cost (%)	
					$w = M_{sp} \pm 3\sigma$			$w = M_{sp} \pm \sigma$			$w = M_{sp} \pm 3\sigma$		$w = M_{sp} \pm \sigma$		Single counter	Multiple counters
					1 bit-flip	2 bit-flips	3 bit-flips	1 bit-flip	2 bit-flips	3 bit-flips	Whole Circuit	Selected Cones	Whole Circuit	Selected Cones		
c1908	25	380	5000	1	72.09	82.38	97.92	75.98	85.67	90.23	9.37	96.67	9.69	89.51	32.81	32.81
				5	38.33	41.61	53.33	54.83	65.63	77.75	29.16	58.65	45.57	93.52	51.88	164.05
				10	36.83	33.23	41.79	36.61	68.18	68.69	40.80	47.13	75.82	88.47	63.93	328.11
				15	35.47	27.71	39.39	37.67	52.38	67.78	40.80	44.82	81.36	89.16	75.97	492.16
c2670	140	736	5000	1	100	100	100	100	100	100	10.18	92.05	10.87	98.24	16.94	16.94
				5	91.45	94.87	96.76	100	100	100	22.32	56.74	41.04	90.24	26.79	84.70
				10	62.96	84.07	78.77	88.96	86.85	100	35.19	57.20	56.75	91.95	33.01	169.40
				15	48.02	63.91	74.32	59.69	85.38	86.9	44.36	62.66	66.49	91.35	39.22	254.11
c3540	22	936	6000	1	63.73	100	64.41	100	100	56.7	12.67	61.91	18.14	88.65	13.32	13.32
				5	27.24	69.02	51.47	92.2	80.35	64.85	41.09	66.79	55.29	89.55	21.06	66.60
				10	24.56	33.41	43.65	73.9	66.1	66.59	58.01	70.32	74.86	90.75	25.95	133.21
				15	21.01	23.08	39.56	50.96	61.36	65.2	67.82	71.26	86.66	91.06	30.84	199.81
c5315	122	1452	6000	1	71.33	49.27	100	100	100	100	1.38	96.21	4.39	83.62	8.59	8.59
				5	59.26	41.76	97.83	96.84	97.3	85.71	6.27	61.81	18.30	80.03	13.58	42.93
				10	36.06	36.28	95.22	89.31	90.68	90.18	12.44	57.01	28.66	84.86	16.73	85.87
				15	27.77	29.38	84.56	81.02	81.02	91.96	17.30	61.49	35.28	87.13	19.88	128.80
c6288	32	2437	7000	1	30.07	39.98	70.76	70.37	69.74	69.74	2.84	50.37	5.11	90.65	5.12	5.12
				5	35.86	29.25	64.18	66.94	63.48	58.13	9.00	55.24	22.51	78.64	8.09	25.58
				10	20.13	20.74	55.41	59.36	56.81	59.97	15.32	36.79	43.88	75.97	9.97	51.16
				15	18.24	18.86	31.21	51.35	51.39	54.09	18.53	31.25	55.44	74.99	11.85	76.74
c7552	108	1897	7000	1	28.45	97.42	100	100	100	100	1.70	98.22	3.95	60.86	6.57	6.57
				5	23.99	86.56	99.97	82.73	94.13	99.91	9.93	91.24	18.39	62.55	10.39	32.86
				10	17.54	51.68	94.65	78.78	80.05	98.23	15.29	83.68	27.79	70.43	12.81	65.73
				15	12.22	39.33	87.92	60.47	75.22	91.07	18.98	79.71	34.46	74.04	15.22	98.59
s9234	275	4090	7000	1	100	100	99.94	100	100	100	0.80	76.41	3.13	43.53	3.05	3.05
				5	98.21	99.57	98.95	95.08	99.12	99.98	3.72	79.56	7.92	61.79	4.82	15.24
				10	96.76	97.18	98.58	94.45	97.66	99.53	7.90	82.86	12.66	70.81	5.94	30.48
				15	96.39	96.95	98.46	92.66	95.87	96.98	11.01	82.34	16.32	70.29	7.06	45.73
sin	25	5416	7000	1	32.91	67.11	43.08	50.91	80.94	66.83	1.37	38.64	4.77	70.78	2.30	2.30
				5	22.32	34.75	37.68	49.97	72.75	63.87	5.76	26.67	26.80	71.51	2.98	11.51
				10	18.27	25.39	28.43	46.84	65.46	61.39	9.19	22.31	43.82	71.44	3.82	23.02
				15	12.19	19.04	22.86	42.22	54.67	56.07	11.40	22.28	56.44	70.53	4.67	34.53
voter	1	13758	10000	1	42.31	65.15	82.78	48.32	73.57	90.74	90.12		95.91		0.91	0.91
log2	32	32060	10000	1	39.56	50.54	76.23	69.49	79.97	79.97	1.94	70.84	3.60	82.09	0.39	0.39
				5	41.92	42.29	47.95	62.35	66.95	68.15	9.72	62.86	19.25	74.88	0.50	1.94
				10	36.96	39.03	42.23	57.83	59.77	61.92	13.88	44.78	33.39	74.07	0.65	3.89
				15	33.00	37.85	39.68	51.17	57.46	55.48	18.88	35.29	43.78	72.93	0.79	5.83

Table 4.1: ibBF and ibSSA EC and area cost of different monitor designs

4.4.2 Simulation Results

Table 4.1 presents the results obtained by applying the proposed monitoring technique to a subset of the ISCAS and EPFL benchmarks. The first column shows benchmark circuit, followed by the number of logic cones and the area given in number of gates in the circuit. The next column shows the error detection latency (EDL), which is given by the workload size S required for the online signal probabilities to converge. Following is the number of monitored cones $C = [1, 5, 10, 15]$. The next columns present the EC of errors induced by 1, 2, or 3 input bit-flips in the selected cones, which are calculated as shown in (4.3). Similarly, the ibSSA EC of the selected cones and of the whole circuit, which is calculated according to (4.4), are also shown.

$$Selected\,Cones\;EC \;=\; \frac{\sum_{k=1}^{C} Sel(EC)_k \cdot Sel(E)_k}{\sum_{n=1}^{C} Sel(E)_n} \tag{4.3}$$

$$Whole\;EC \;=\; \frac{\sum_{k=1}^{C} Sel(EC)_k \cdot Sel(E)_k}{\sum_{n=1}^{T} E_n} \tag{4.4}$$

Where C is the number of selected cones and T is the total number of cones. E_n are the number of errors at each cone, $Sel(E)_k$ is the number of errors in the selected cones and $Sel(EC)_k$ is the EC of each of the selected cones obtained with the different signature windows. The last columns show the area cost for both the single and multiple counter designs.

The EC of the whole circuit increases as more cones are monitored. When all cones are monitored, the EC of the selected cones and of the whole circuit converges to the maximum EC observable for each signature window. Using a signature window $w = M_{sp} \pm \sigma$ (31.7% probability of false alarms) for circuit *log2*, Table 4.1 shows an ibBF EC on the selected cones of 79.97% when monitoring 1 cone, with an area cost of 0.39%, and an EC of 55.48% when monitoring 15 cones, with a respective area cost of 0.79%. Note that the ibBF EC is higher for 3 input bit-flips than for 1 input bit-flip. This is expected, as more bit-flips are more likely to propagate errors to the output, producing a more observable SEB. Additionally, Table 4.1 also shows an ibSSA EC of 43.78% and 72.93% on the whole circuit and the 15 selected cones respectively, with the same area cost of 0.79%. If all 32 logic cones are monitored, the ibSSA EC of the whole circuit increases to 71.85% with an area cost of 1.27% using a single counter monitor. Monitoring the logic cone that exhibits the most errors on the four largest circuits using a signature window $w = M_{sp} \pm 3\sigma$ (0.3% probability of false alarms), results in an average ibBF and ibSSA EC of 75.5% and 69.1% respectively, with an average area cost of 1.66%. Using a signature window $w = M_{sp} \pm 3\sigma$, we can see an average ibBF and ibSSA EC of 84.4% and 73.1% respectively, with the same average area cost of 1.66%. An error detection latency estimation for these circuits can be performed by synthesizing them with a standard 90nm cell library using commercial tools. The resulting operating frequency is

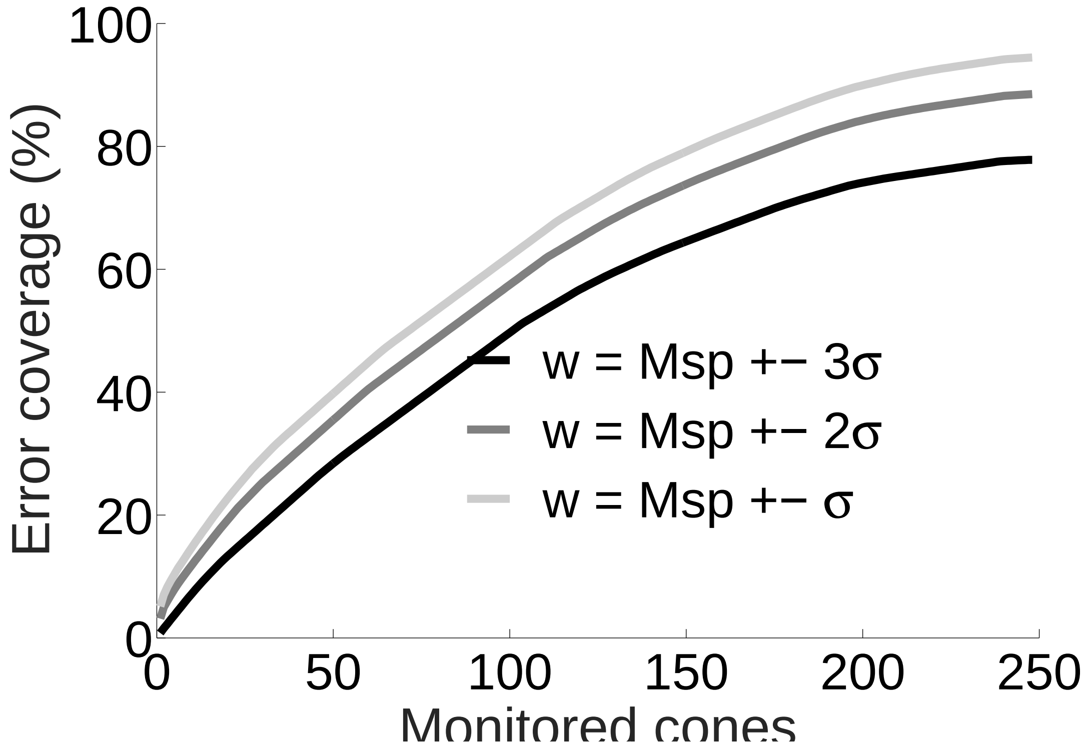


FIGURE 4.11: ibSSA EC vs monitored cones with $S = 7000$ for circuit s9234

in the range of [3MHz, 1.1GHz], which produces an error detection latency in the range of [0.01, 3.3] milliseconds when detecting SEB after 10000 clock cycles.

The EC for narrow signature windows w is higher than for wide windows. The three signature windows $w = M_{sp} \pm \{3,2,1\}\sigma$ shown in Figure 4.11, have a 0.3%, 4.5% and 31.7% respective probability of raising a false alarm. Narrower windows are stricter on the online signal probability variations that can be detected, resulting in higher EC. Additionally, narrower windows will detect SEB at a higher rate than wider windows, however, some of these detections may be false alarms. For maintainability planning purposes, a device that exhibits SEB at a higher rate than other identical devices in-the-field, even if some SEB detections are false, may be prioritized for maintenance.

Figure 4.12 presents the trend of the area cost of the monitoring architecture vs the size of the monitored circuit. For the larger circuits, the area cost percentage is lower than for smaller circuits. This is expected, as the area cost of the monitors is only dependent on the number of monitored cones (C), the area of the S-counter of the *controller* and the area of each monitor ($m(S)$) (Fig. 4.8). The area of each monitor depends on the workload size S, which determines the size of the counter and the comparators. The area cost of a multiple counter design is presented in (4.5), where *size* is the size of the monitored circuit.

$$Cost = \frac{controller + C \cdot m(S)}{size} \tag{4.5}$$

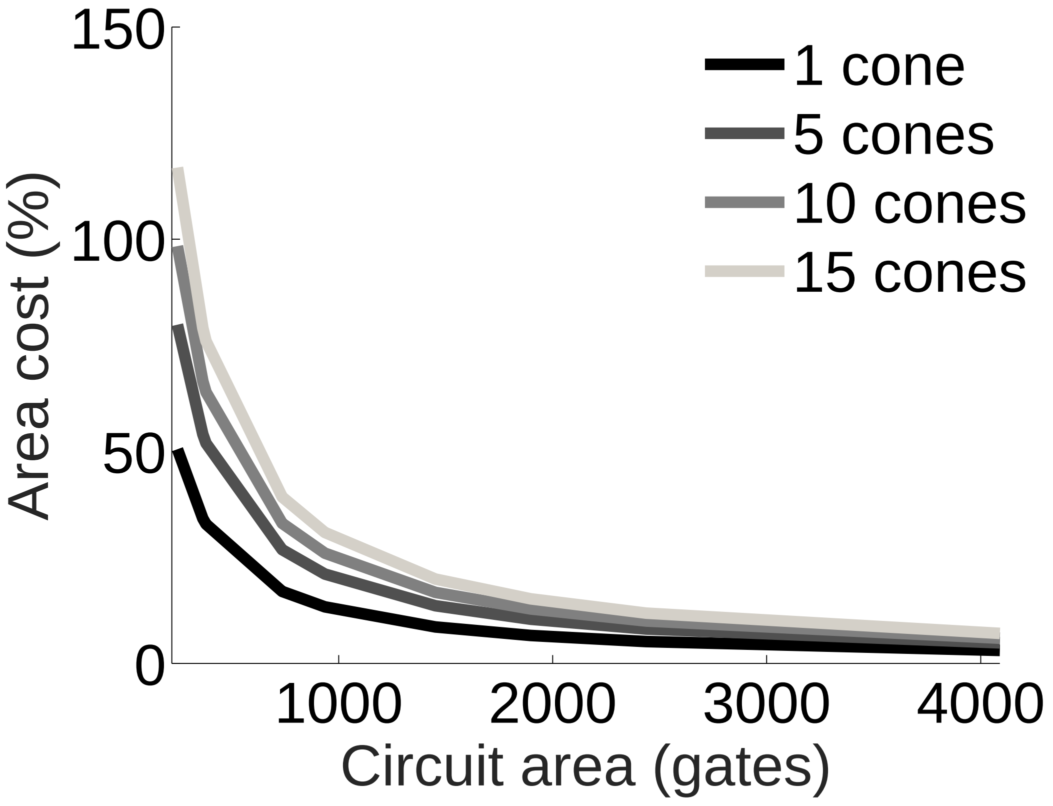


Figure 4.12: Area cost of single counter monitors compared to the size of the circuit

The number of POs of logic circuits is bounded due to physical constraints, therefore, monitoring only POs would incur in relatively low area cost. An estimation of the possible implementation of this technique for the circuit *twentythree* of the EPFL benchmarks with more than 23 million gates, indicates that monitoring all 68 of the PO would incur an approximate area cost of 0.0066% using a single counter design, and 0.076% using a multiple counter design.

4.5 Discussion

The proposed technique offers a better cost-error correction ability trade-off than duplication-based CED, since duplication-based CED can detect almost 100% of single errors with a low error detection latency, but incur an area and power overhead of more than 100% [128]. CED techniques using error detecting codes can achieve more than 90% error coverage with an overhead in the range of 18% to 84%, but may have an impact on system performance and require custom design approaches [59], [95], [129]. Online BIST techniques achieve a near 100% fault coverage with an area cost in the range of 2% to 165% for the selected circuits. These values were calculated considering the decompressor and the best compression efficiency reported in [133]. Online BIST techniques however, are unable to operate concurrently and are therefore, incapable of detecting SEB caused by intermittent faults if it does not manifest while in test mode. The proposed technique does not target fault detection or diagnosis. However, it is capable to operate concurrently and detect SEB caused by intermittent faults. Moreover, through constant monitoring of the online signal probabilities, it may be possible to determine if permanent faults are present in the circuit, when the mean signal probability falls outside the signature window consistently.

The proposed technique is capable of detecting systematic erroneous behaviour caused by intermittent bit-flips and stuck-at faults. However, intermittent faults activated by power or thermal constraint violations, aging, or power-management techniques like dynamic voltage and frequency scaling (DVFS), may also manifest as intermittent delay on critical paths. Therefore, further investigation of the impact of intermittent delay faults on the online signal probabilities is required.

4.6 Concluding Remarks

This chapter presented a novel technique for monitoring systematic erroneous behaviour online. First, an analysis was presented for the online signal probabilities (Section 4.2) at the outputs of a circuit when the latter exhibits systematic erroneous behaviour. Based on this analysis, a RTL design methodology was proposed (Section 4.3) for two monitoring architectures (Fig. 4.7) in order to observe the online signal probabilities at the logic cones of a circuit. The two architectures offer various trade-offs between error detection latency and area cost. The technique was applied on a set of EPFL and ISCAS benchmark circuits. The proposed technique is scalable to large designs, as its area cost depends only on the number of monitored logic cones, not on the size of the monitored circuit. It was demonstrated (Table 4.1) that the proposed monitoring technique achieves, on the largest circuits, an average error coverage of 84.4% and 73.1% of errors induced by intermittent bit-flips and intermittent stuck-at faults, respectively, with an average area cost of 1.66% and an error detection latency in the range of [0.01,3.3] milliseconds.

Chapter 5

Single Gate Approximate Circuits

Low cost and low power have become a priority in the design of modern ICs, particularly after the recent growth of embedded and mobile devices. Many of the applications of such devices like media processing, data mining, geo-monitoring or data recognition, do not often require exact computations and approximate results suffice [134]. Traditionally hardware implementations of these applications were designed for an exact computation of all of their tasks. However, approximate circuits have appeared in recent years that challenge the strict notion of exact computations [135], in order to produce a approximate computations that are sufficient for the target application, while reducing area cost and power consumption. This low cost property of approximate circuits generates interest in investigating their use for error monitoring and protection of cost constrained systems.

This chapter presents a novel circuit approximation technique that can be used for low cost non-intrusive concurrent error detection (CED) and for selective fault tolerance (SFT). The circuit approximation is based on finding functionality that behaves similarly to single logic gates or constant values. A brute-force algorithm is proposed to select the input subsets to approximate. This algorithm is applied to arithmetic circuits as a proof of concept and to a set of the LGSynth91 benchmarks. Additionally, using the generated approximate logic circuits for SFT results in significant area cost reductions compared to existing techniques. Furthermore, it is shown that the generated approximate logic circuits can be used for low cost concurrent error detection.

This chapter is organized as follows. Section 5.1 presents a background on approximate logic circuits and their applications to error masking and concurrent error detection. Section 5.2 introduces the premise of the proposed single gate approximation for the creation of approximate logic circuits (ALCs) and provides circuit examples as a proof of concept. Section 5.3 details the Input Subset Selection Algorithm along with a detailed explanation of its functionality assisted by multipliers and random combinational circuit examples. Section 5.4 presents the results after applying this algorithm to a set of the LGSynth 91 benchmarks. Section 5.5 shows a comparison between using the generated

ALCs of this technique for SFT and an existing SFT methodology. Section 5.6 details the application of ALCs for concurrent error detection. Section 5.7 provides a discussion on this technique and outlines the future research directions, followed by the conclusions on Section 5.8.

5.1 Background: Approximate logic circuits

Approximate logic circuits (ALCs) disregard the exact input-output boolean representation of a logic circuit while producing a sufficient result within the specifications of an imprecision-tolerant application. As such applications are computation-heavy tasks, most of the implementations of ALCs are applied to common arithmetic circuit structures like adders [136]–[138] and multipliers [139], [140]. On the other hand, circuit approximations have been proposed in the past as a means to reduce the cost of adding error detection and protection mechanisms. Following are some of the techniques proposed using ALCs

A methodology to reduce Soft Error Rate (**SER**) of logic circuits using approximate logic functions was proposed in [141]. They propose to use an error masking function in the form of:

$$\hat{G} = F + GH \tag{5.1}$$

Where G is the original logic function of the circuit and F & H are approximations of G such that the set of minterms is:

$$F \leq G \leq H \tag{5.2}$$

According to equation 5.2, if an input $\vec{x}$ is a minterm of G then it must be a minterm of H and that all minterms of F are minterms of G. The error masking capabilities of this function are similar to that of a voting function used in Triple Modular Redundancy (TMR) schemes. One of the limitations of this approach is that the picking of F & H is done manually on a trial and error basis for relatively small logic functions.

Logic masking solution for the mitigation of single event upsets using ALCs, has also been proposed based on the concept of unate functions [142]. Unate functions are defined as follows [143]:

- **Positive Unate**: Function F is positive unate for an independent variable x_i IFF $\bar{x_i}$ is not part of a minterm of in the sum of products of F.

- **Negative Unate**: Function F is negative unate for an independent variable x_i IFF x_i is not part of a minterm of in the sum of products of F.

Unate functions approximate the circuits in two different types: 1-approximation & 0-approximation. Both approximations are computed for a specific output of the circuit then one of the two is chosen. However, this approach is dependent on knowledge of the exact gate structures of the circuit to approximate [144], [145].

ALCs have also been proposed for concurrent error detection (CED). Techniques using ALCs have been used for detecting errors caused by stuck-at faults and single event upsets [146] and by delay faults [147] . The approximate logic circuits are built by only using the minterms with the highest impact of a boolean function of either the on-set (output is logic-1) or the off-set (output is logic-0). For example, in the boolean function $f = a + \bar{b}c + b\bar{c}d\bar{e}$, the contribution of the minterm $b\bar{c}d\bar{e}$ has less impact and is discarded, The discarded minterms are converted into *don't-cares*. By using *don't-cares* it is possible to explore a vast set of approximations, however that requires the use of satisfiability algorithms to select those that provide the desired result within specification.

This chapter presents a novel technique that proposes a new type of circuit approximation for general logic. This approximation is based on finding functionality that mimics the behaviour of basic logic elements such as logic gates. Next section details the premise under which the approximate logic circuits are generated.

5.2 Single Gate Approximation Premise

The premise of this circuit approximation is the following: consider that for each output z in a circuit, let $z = x$ where $\vec{x} = \{x_0, x_1, \cdots, x_n\}$ (n-input variables) with an input pattern set $X = \{0, 1, \cdots 2^{n-1}\}$, then there exists one or more input subspaces $\hat{X} \subseteq X$ that exhibit a functionality equivalent to a basic logic gate. Figure 5.1 shows an example of the input subspace selection.

With the following truth table:

Input	x_2	x_1	x_0	z
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	**1**	**0**	**0**	**0**
5	**1**	**0**	**1**	**1**
6	**1**	**1**	**0**	**1**
7	**1**	**1**	**1**	**1**

Table 5.1: Example input subspace selection

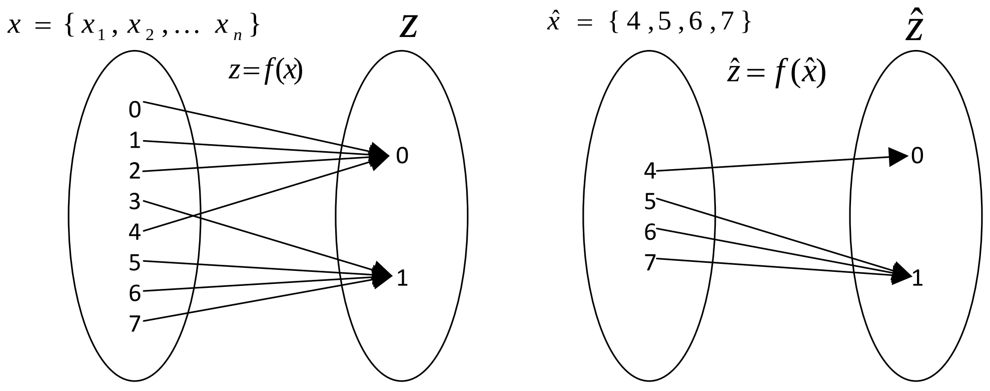


FIGURE 5.1: Example of input subspace selection

The selected subspace of inputs $\{4, 5, 6, 7\}$ shown in figure 5.1 mimics the behaviour of an OR gate.

5.2.1 Full Adder Example

The input subspace selection shown in Figure 5.1 is an example approximation of the C_{out} output of a basic full adder. The truth table of a full adder is shown in Table 5.2. After simplification, the boolean equation for each of the outputs are the following: For the Carry Out, $C_{out} = (x \cdot y) + (C_{in} \cdot (x \oplus y))$ and for the Sum $S = x \oplus y \oplus C_{in}$. Those boolean functions produce the well-known 5-gate implementation shown in Figure 5.2.

x	y	C_{in}	C_{out}	S
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 5.2: Truth table of a Full Adder

Using the same input subspace as Figure 5.1, the reduced truth table for C_{out} results are shown in Table 5.3. From that, it is possible to create an ALC by simplifying directly while setting all the input combinations outside the selected input subspace logical 0's, as shown in the Karnaugh map of Figure 5.3.

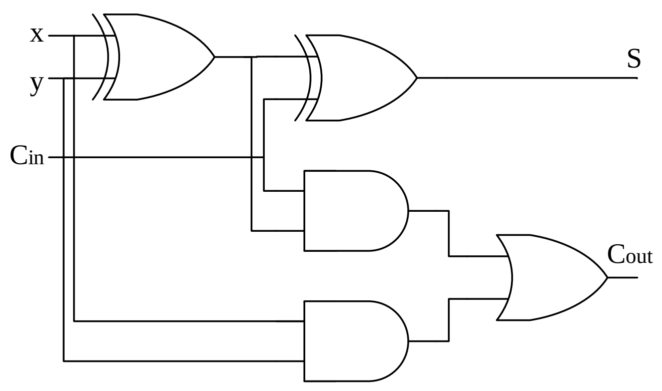


FIGURE 5.2: Full adder circuit

x	y	C_{in}	C_{out}
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 5.3: Reduced Truth table of C_{out}

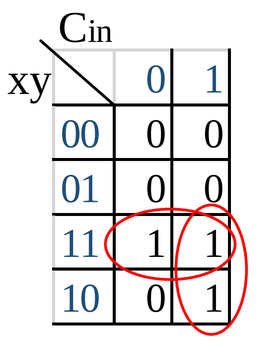


FIGURE 5.3: Karnaugh map for the approximation of C_{out}.

Which will produce an equivalent circuit shown in Figure 5.4. This circuit is composed of a single gate and is capable of reproducing C_{out} using a simple OR gate between y and C_{in} when $x = 1$, giving a resulting boolean equation in the form: $C_{out} = x \cdot (y + C_{in})$.

This example shows that an approximation $\hat{C}_{out} = y + C_{in}$ is possible, only with $x = 1$. In this example, the input x serves as the validation section (Highlighted in red), serving a purpose similar to the characteristic function $\chi(x)$ presented in Section 3.2.1. Note that another ALC could have been generated for the C_{out} output using the set of inputs

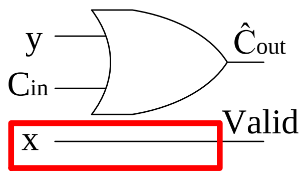


FIGURE 5.4: ALC of the C_{out} of a Full Adder

.

$\{0, 1, 2, 3\}$, or other subspaces, however, other subspaces might not generate a 1-gate approximation. ALCs for the S output of the full adder can be generated using similar steps.

5.2.2 4-Input Circuit Example

Consider the 4-input 1-output random logic circuit described by the truth table in Table 5.4 with output vector z. The resulting boolean function is $z = \bar{x}_0 \cdot x_1 + x_0 \cdot x_3\,(x_1 \oplus x_2 + \bar{x}_2) + x_0 \cdot x_2\,(x_1 \oplus x_3)$ and logic circuit using ten gates and two inverters (Figure 5.5).

x_0	x_1	x_2	x_3	z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Table 5.4: Truth table of a random 4-input 1-output circuit

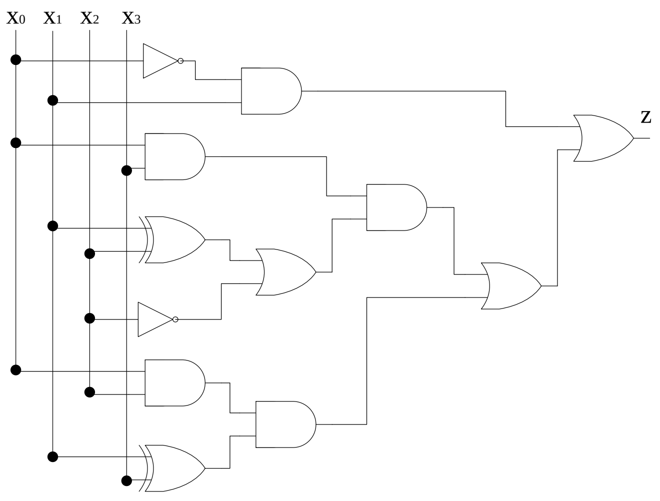


FIGURE 5.5: Simplified circuit of a random 4-input 1-output circuit

.

Following is the truth table of two input subspaces A and B and their corresponding ALC.

x_0	x_1	x_2	x_3	z
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1

Table 5.5: Truth table of set A

With a boolean function $z = x_0 \cdot \bar{x}_1 \cdot (x_2 \cdot x_3)$.

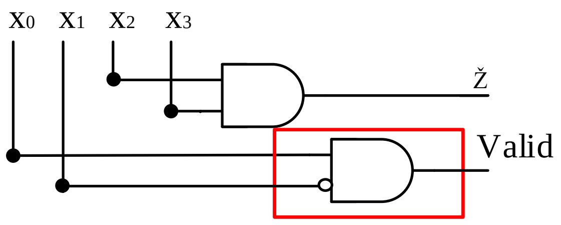


FIGURE 5.6: AL for the random 4-input 1-output circuit using set A

.

x_0	x_1	x_2	x_3	z
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Table 5.6: Truth table of B

With boolean function $z = x_0 \cdot x_1 \cdot (x_2 \oplus x_3)$.

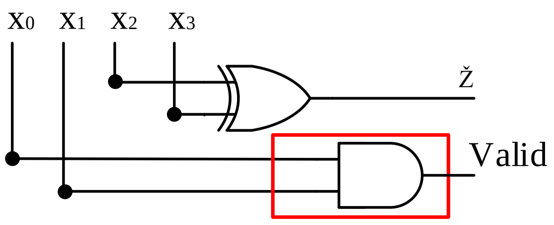


FIGURE 5.7: AL for the random 4-input 1-output circuit using set B

.

Both circuits in Figures 5.6 and 5.7 provide similar solutions where the upper parts of the circuit compute an approximate output $\hat{z}$ and the lower part validates whenever the input pattern belongs in the selected subspace. In those two examples, inputs x_0, x_1 provide the same validation as x did on the full adder example. This solution requires all of the 4 inputs, but only 4 of the 16 possible input combinations of the full input space.

5.2.3 Second 4-input Example

Another example of an ALC for random logic is presented as follows. A 4-input 1-output circuit was generated followed by the selection of input subspace I. The resulting boolean function of the reduced truth table presented in Table 5.7 is $z = x_0 \cdot x_1 \cdot x_3$, which simplifies to the ALC shown in Figure 5.8.

x_0	x_1	x_2	x_3	z
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Table 5.7: Truth table of I

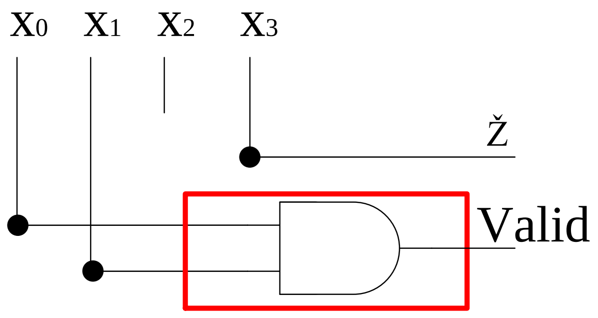


FIGURE 5.8: AL for the random 4-input 1-output circuit using set I

.

In this particular example, x_3 can be mapped directly to z when $x_0 \cdot x_1 = 1$. Input x_2 was eliminated by simplifying the logic of subspace I. In this example, an input behaves the same as the approximated output whenever the *Valid* is asserted. This indicates that an input-output bypass approximation is possible.

These example circuits show that these types of ALCs consist of two parts:

- **Approximation:** The part that approximates the output using single gates or an input-output bypass.
- **Validation:** The part that determines when the current input pattern belongs in the select input subspace of the approximated circuit. This part validates whenever the approximation section has correctly approximated the output. The behaviour is similar to the $\chi(x)$ function used in Section 3.2.1.

5.2.4 Types of Approximate Logic Circuits

The examples shown in Figures 5.6, 5.7 and 5.8 show that there are different types of ALCs that can be generated. From the first example circuit, Figures 5.6 and 5.7 show that two inputs, x_2 & x_3, correctly approximate the output Z for their respective input subspace. Figure 5.6 approximates the output with a NAND gate while 5.7 does it with an XOR gate. For both of this cases, the approximation is done with a simple 2-input gate. In the case of figure 5.8, the approximation is done by bypassing the input x_3 to the output Z. Note that the validation section of the ALCs is only asserted (logic 1) when the pattern present at the inputs belong to the selected input subspace for which the ALC was generated.

Three types of generated AL can be identified depending of the behaviour of the selected input subspace: Those where the approximation section behaves as a simple gate, those that bypass the input directly to the output and those that keep a constant logic value for the whole selected subspace. Following is a classification of the ALC types.

- **Single gate:** The **SG** approximation type is done by a single basic N-input gate. For all validated input patterns, the output behaves as logic gate. The possible logic gates are: INV, NAND, NOR, AND, OR, XOR, XNOR.
- **IO Bypass:** The **IOB** approximation type is merely a bypass from the input to the output. This means that for input patterns corresponding to the selected input subspace(s) of the validation section, the output follows or mimics one of the inputs.
- **Logic Constant:** Similar to IOB, the **LC** approximation type holds a logic value for all the input patterns belonging to the previously selected input subspace(s) for the validation section.

5.3 Input Subspace Selection Algorithm

Given a functional description in the form of a truth table, a per-output (logic cone) search algorithm is required to find the input subspaces that exhibit a behaviour similar to one of the ALC types described above. The selection of which output (logic-cone) to approximate may be decided by the user or by any type of vulnerability analysis [94], [109], [131], similar to the technique presented in Chapter 4. Figure 5.9 presents the flowchart of the input subspace search algorithm. This is a brute-force algorithm with a complexity of $\mathcal{O}(2^N)$. The algorithm requires the truth table of the logic cone, with an 2^I input space size for I inputs, in addition to the parameter g. This parameter determines the size of the subspaces that will be searched by the algorithm as 2^g. All possible permutations of the inputs are explored. For each permutation, the permutated input subspace is divided in 2^g subspaces and each of those subspaces are checked to determine whether they exhibit a behaviour similar to the **SG, IOB or LC** ALC types. When a subspace does, it is stored. The input coverage is calculated by adding the number of stored subspaces of each type of ALC and multiplied by g. Finally, the type of ALC with the highest input coverage is selected as the best candidate for approximation.

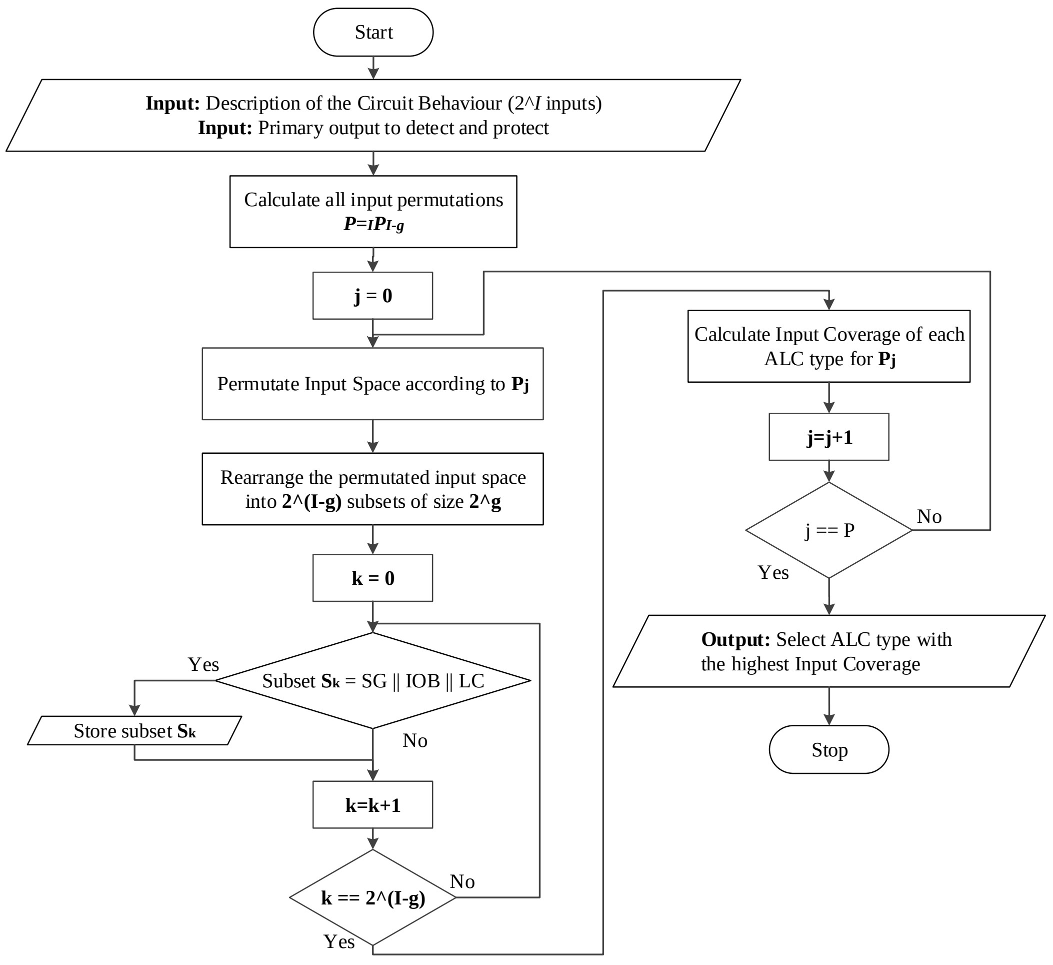

Start
Input: Description of the Circuit Behaviour (2^*I* inputs)
Input: Primary output to detect and protect
Calculate all input permutations
P=*IPI-g*
j = 0
Permutate Input Space according to **Pj**
Rearrange the permutated input space into **2^(I-g)** subsets of size **2^g**
k = 0
Subset **Sk** = SG || IOB || LC
Yes
No
Store subset **Sk**
k=k+1
k == 2^(I-g)
No
Yes
Calculate Input Coverage of each ALC type for **Pj**
j=j+1
j == P
No
Yes
Output: Select ALC type with the highest Input Coverage
Stop

FIGURE 5.9: Input subspace selection algorithm

5.3.1 Parallel Multiplier Examples

The full adder example presented in Section 5.2.1 is too simple to use as an example for the Input subspace selection algorithm. The algorithm is therefore applied to two parallel multiplier circuits of different sizes. First a 2-bit multiplier is presented. The truth table for this circuit is show in table 5.8.

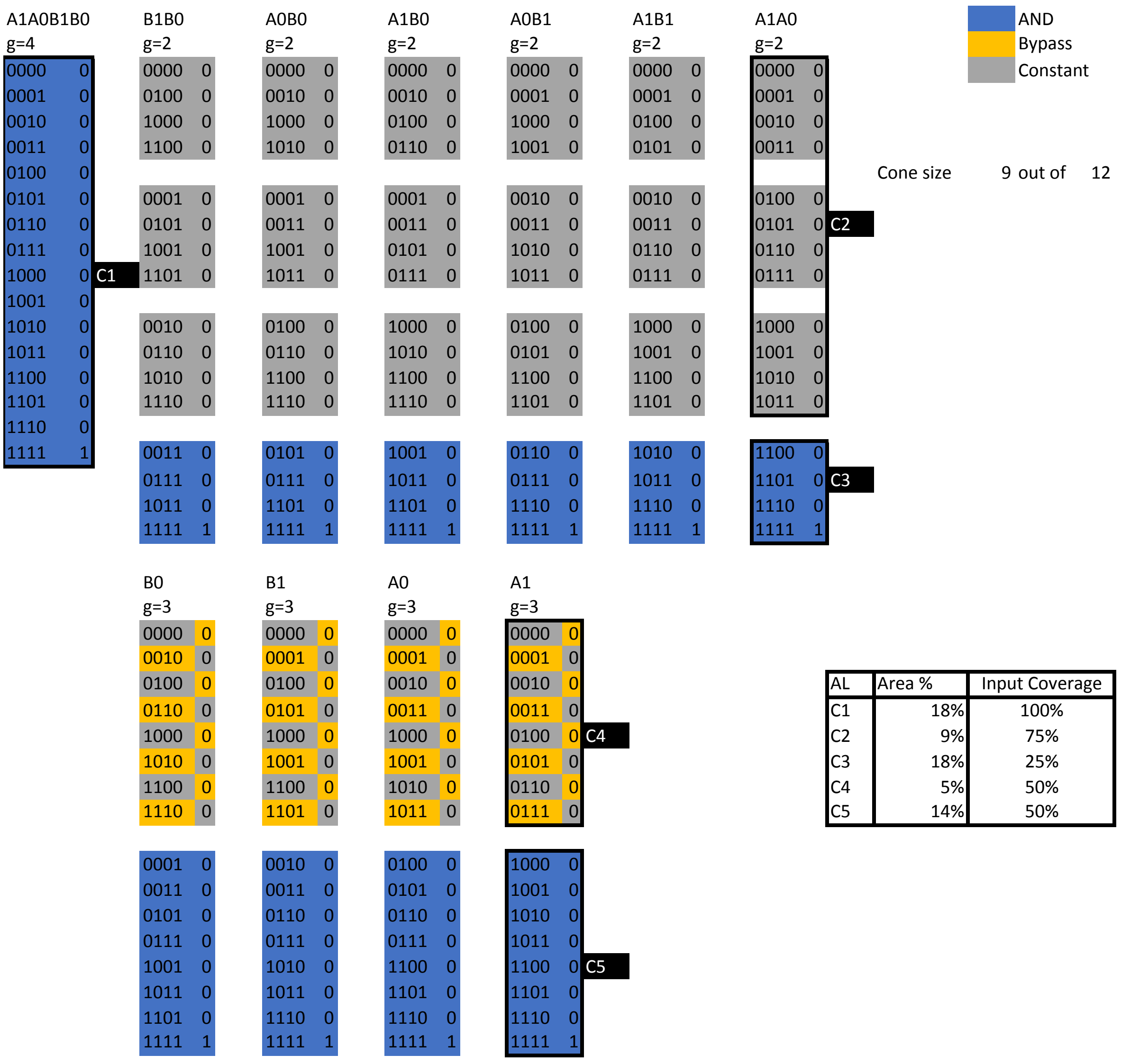
Algorithm 1: Input Subspace Selection

```
  inputs : Functional description (Truth Table) $F$ with $I$ number inputs ; subspace
           size $g$
  outputs: Input space coverage of each ALC type
  // Compute $I-g$ of input permutations $P = {}_IP_{I-g}$;
1 foreach $P$ do
      // Assign current permutation to the input indexes $t$;
2     $t_{I-g}\cdots t_0 = P_{I-g}\cdots P_0$;
      // For all the possible subspaces of size $2^g$ ($g-$input gates);
3     for $k = 0$ to $2^{I-g}$ do
          // Generate a subspace $s$ of size $2^g$ and assign its inputs
          according to the input indexes $t$
4         $s_{0,I-m}\cdots s_{2^g,0} = t_{I-g}\cdots t_0$;
          // Calculate each of the outputs $so$ of the subspace $s$ according
          to the functional description $F$;
5         $so_0 = Fs$;
          // Check if the subspace $s$ exhibits a SG, IOB or LC type of
          behaviour;
6         if subspace2SG($s, so$) == SG or IOB or LC then
              // Store the subspace;
7             Store()
      // Report the number of occurrences of each ALC type for each $P$;
8     Report()
  // Calculate input coverage by adding the the number of occurrences
  of each ALC type Calculate()
```

A_1	A_0	B_1	B_0	Z_3	Z_2	Z_1	Z_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Table 5.8: 2-bit Parallel Multiplier Truth Table

Applying the algorithm on all of the 4 outputs of the circuit produces various ALC candidates for each individual output in the Z vector. Figures 5.13, 5.12, 5.11 and 5.10, show how the truth table is rearranged based on different g parameters, how the selection of input subspaces is performed, and the resulting area overhead and input coverage of the selected ALC candidate for each output. The first column represents a $g = 4$, which yields a subspace size of 16. The next column on the right represent a $g = 2$ or subspace size of 4. The columns below show a $g = 3$ with 8-element subspaces. The colour code legend is shown at the top right. This indicates the type of approximation (SG, IOB or LC) corresponding to the generated ALC and which basic gate in the case of an SG approximation type. The selected input subspaces are boxed and tagged as C1, C2 and so forth. Finally, on the lower right there is a table with the information of each of the generated ALC with its corresponding Area Overhead (Area %) and Input Coverage.



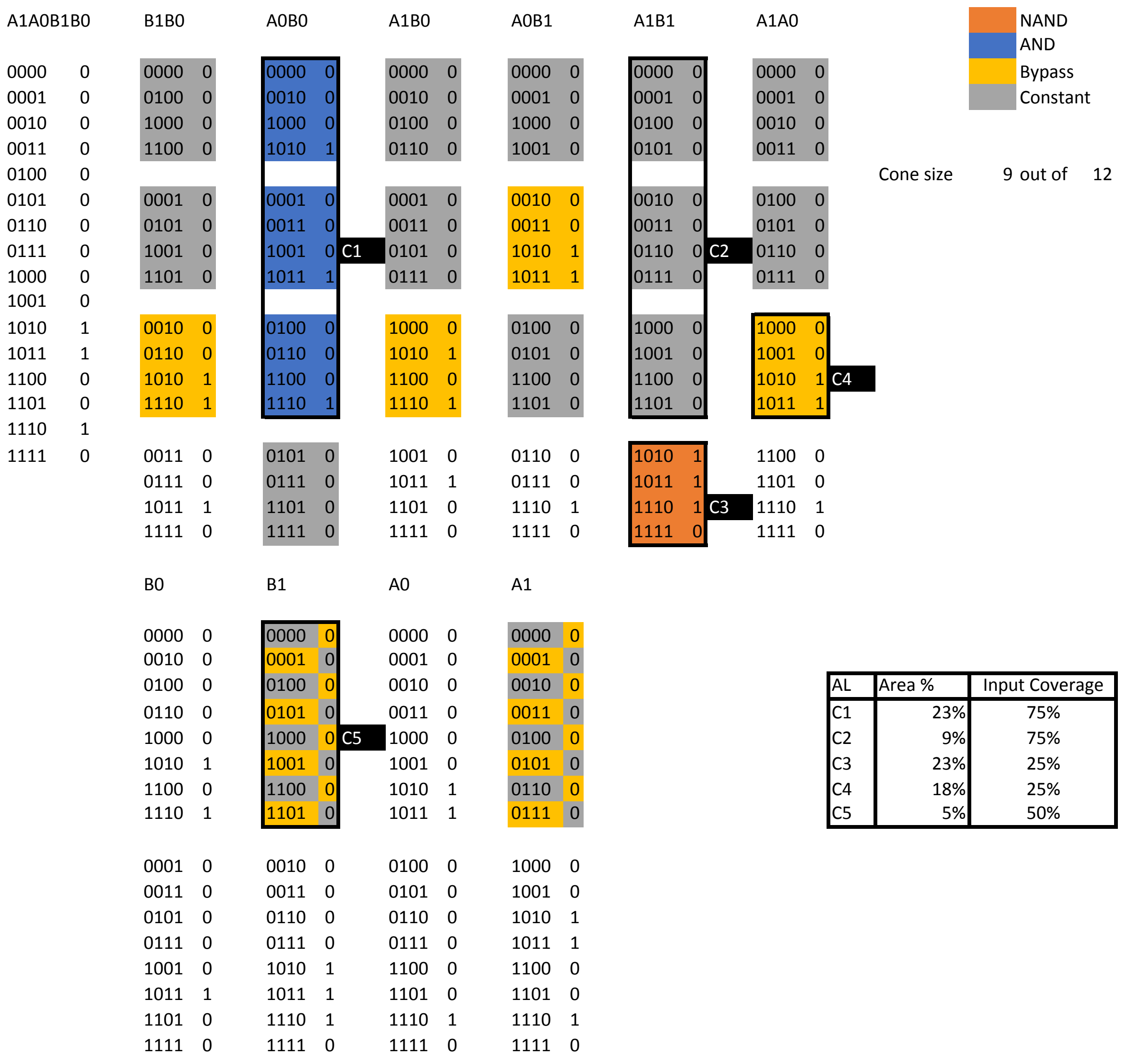
A1A0B1B0 g=4		B1B0 g=2		A0B0 g=2		A1B0 g=2		A0B1 g=2		A1B1 g=2		A1A0 g=2	
0000	0	0000	0	0000	0	0000	0	0000	0	0000	0	0000	0
0001	0	0100	0	0010	0	0010	0	0001	0	0001	0	0001	0
0010	0	1000	0	1000	0	0100	0	1000	0	0100	0	0010	0
0011	0	1100	0	1010	0	0110	0	1001	0	0101	0	0011	0
0100	0	0001	0	0001	0	0001	0	0010	0	0010	0	0100	0
0101	0	0101	0	0011	0	0011	0	0011	0	0011	0	0101	0
0110	0	1001	0	1001	0	0101	0	1010	0	0110	0	0110	0
0111	0	1101	0	1011	0	0111	0	1011	0	0111	0	0111	0
1000	0	0010	0	0100	0	1000	0	0100	0	1000	0	1000	0
1001	0	0110	0	0110	0	1010	0	0101	0	1001	0	1001	0
1010	0	1010	0	1100	0	1100	0	1100	0	1100	0	1010	0
1011	0	1110	0	1110	0	1110	0	1101	0	1101	0	1011	0
1100	0	0011	0	0101	0	1001	0	0110	0	1010	0	1100	0
1101	0	0111	0	0111	0	1011	0	0111	0	1011	0	1101	0
1110	0	1011	0	1101	0	1101	0	1110	0	1110	0	1110	0
1111	1	1111	1	1111	1	1111	1	1111	1	1111	1	1111	1

B0 g=3		B1 g=3		A0 g=3		A1 g=3	
0000	0	0000	0	0000	0	0000	0
0010	0	0001	0	0001	0	0001	0
0100	0	0100	0	0010	0	0010	0
0110	0	0101	0	0011	0	0011	0
1000	0	1000	0	1000	0	0100	0
1010	0	1001	0	1001	0	0101	0
1100	0	1100	0	1010	0	0110	0
1110	0	1101	0	1011	0	0111	0
0001	0	0010	0	0100	0	1000	0
0011	0	0011	0	0101	0	1001	0
0101	0	0110	0	0110	0	1010	0
0111	0	0111	0	0111	0	1011	0
1001	0	1010	0	1100	0	1100	0
1011	0	1011	0	1101	0	1101	0
1101	0	1110	0	1110	0	1110	0
1111	1	1111	1	1111	1	1111	1

AL	Area %	Input Coverage
C1	18%	100%
C2	9%	75%
C3	18%	25%
C4	5%	50%
C5	14%	50%

FIGURE 5.10: Algorithm applied to the output Z_3 of the 2-bit parallel multiplier.

Figure 5.10 shows the results for the output Z_3. The **C1** ALC candidate is an SG type generated with a $g = 4$ which synthesizes as a 4-input AND gate and a validation as a constant 1. **C2** is an LC type where the approximation section is a constant 0 and the validation is a 2-bit NAND gate whose inputs are A_1 and A_0. ALC **C3** is similar to C1, although both the approximation and validations sections are synthesized to a 2-bit AND gate each. In this case, the output $\hat{Z}_3$ is approximated with an AND gate with inputs B_1 and B_0, and the validation part consists of an AND gate with inputs A_1 and A_0. The ALC **C4**, with $g = 3$, can be generated as either an IOB or an LC type, both yielding the same result. As an IOB, the approximation is $\hat{Z}_3 = A_1$ whenever $A_1 = 0$, needing just an INV as the validation. As an LC, the approximation becomes a constant 0 whenever $A_1 = 0$, with the same area overhead as the IOB case. Finally for **C5**, the approximation part is a 3-input NAND gate and the validation is simplified as $V = A_1$.



A1A0B1B0	
0000	0
0001	0
0010	0
0011	0
0100	0
0101	0
0110	0
0111	0
1000	0
1001	0
1010	1
1011	1
1100	0
1101	0
1110	1
1111	0

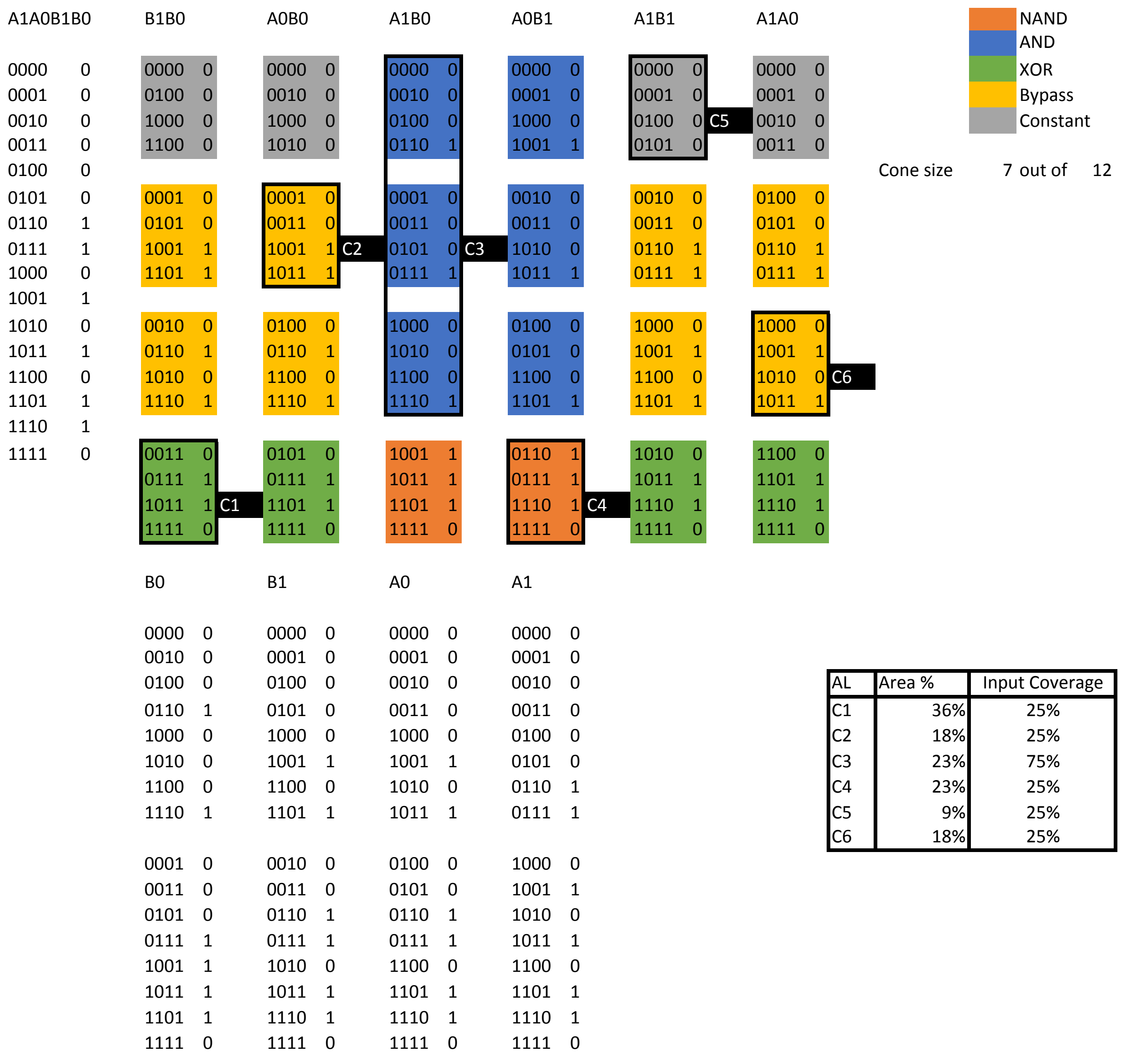
B1B0		A0B0		A1B0		A0B1		A1B1		A1A0	
0000	0	0000	0	0000	0	0000	0	0000	0	0000	0
0100	0	0010	0	0010	0	0001	0	0001	0	0001	0
1000	0	1000	0	0100	0	1000	0	0100	0	0010	0
1100	0	1010	1	0110	0	1001	0	0101	0	0011	0
0001	0	0001	0	0001	0	0010	0	0010	0	0100	0
0101	0	0011	0	0011	0	0011	0	0011	0	0101	0
1001	0	1001	0	0101	0	1010	1	0110	0	0110	0
1101	0	1011	1	0111	0	1011	1	0111	0	0111	0
0010	0	0100	0	1000	0	0100	0	1000	0	1000	0
0110	0	0110	0	1010	1	0101	0	1001	0	1001	0
1010	1	1100	0	1100	0	1100	0	1100	0	1010	1
1110	1	1110	1	1110	1	1101	0	1101	0	1011	1
0011	0	0101	0	1001	0	0110	0	1010	1	1100	0
0111	0	0111	0	1011	1	0111	0	1011	1	1101	0
1011	1	1101	0	1101	0	1110	1	1110	1	1110	1
1111	0	1111	0	1111	0	1111	0	1111	0	1111	0

B0		B1		A0		A1	
0000	0	0000	0	0000	0	0000	0
0010	0	0001	0	0001	0	0001	0
0100	0	0100	0	0010	0	0010	0
0110	0	0101	0	0011	0	0011	0
1000	0	1000	0	1000	0	0100	0
1010	1	1001	0	1001	0	0101	0
1100	0	1100	0	1010	1	0110	0
1110	1	1101	0	1011	1	0111	0
0001	0	0010	0	0100	0	1000	0
0011	0	0011	0	0101	0	1001	0
0101	0	0110	0	0110	0	1010	1
0111	0	0111	0	0111	0	1011	1
1001	0	1010	1	1100	0	1100	0
1011	1	1011	1	1101	0	1101	0
1101	0	1110	1	1110	1	1110	1
1111	0	1111	0	1111	0	1111	0

AL	Area %	Input Coverage
C1	23%	75%
C2	9%	75%
C3	23%	25%
C4	18%	25%
C5	5%	50%

FIGURE 5.11: Algorithm applied to the output Z_2 of the 2-bit parallel multiplier.

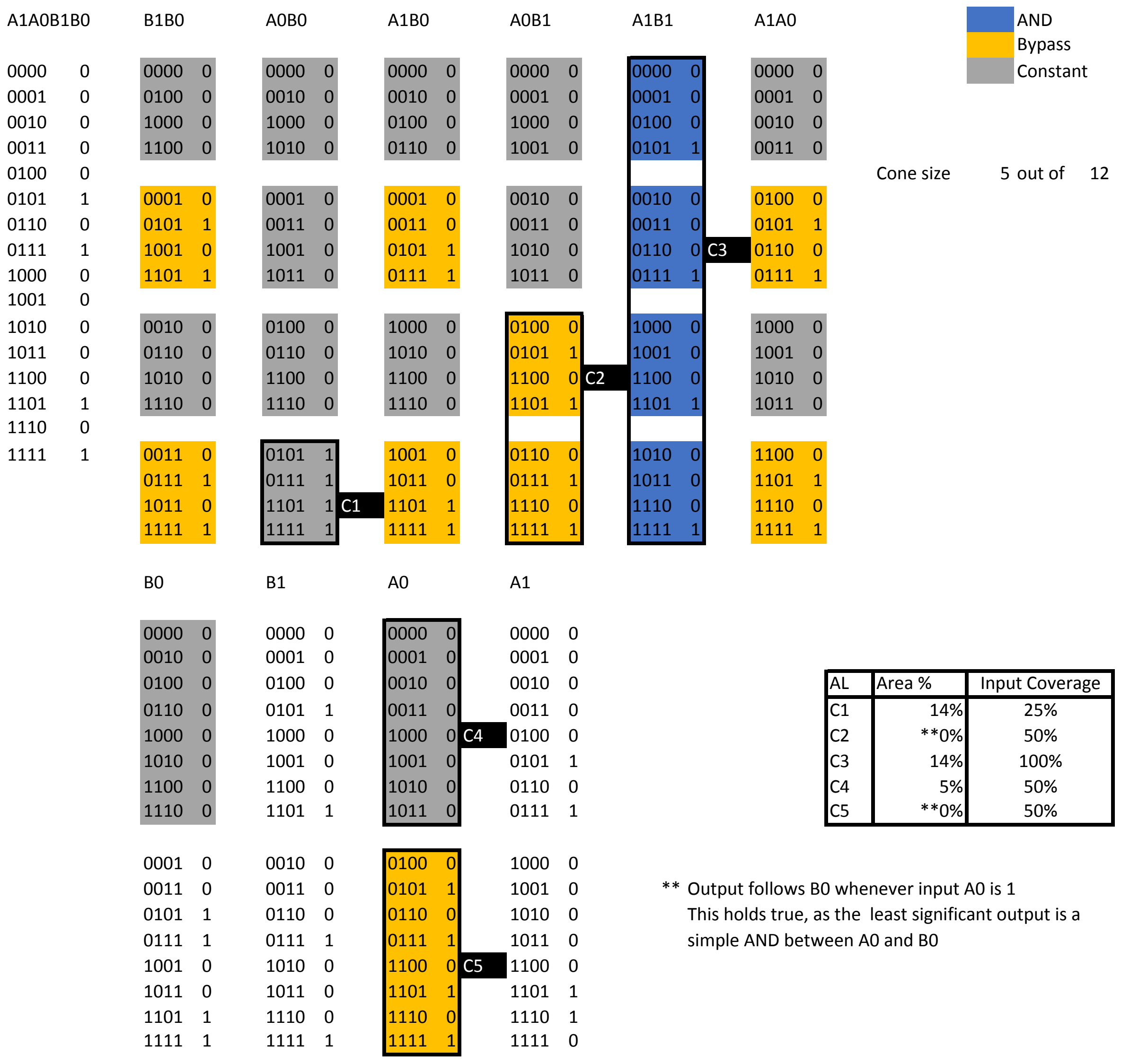
Figure 5.11 presents the algorithm applied to the output Z_2. The **C1** is an SG type generated with a $g = 2$ which synthesizes as a 2-input AND gate of inputs A_1 and B_1, and a validation consisting of a 2-input NAND with inputs A_0 and B_0. ALC **C2** is an LC type where approximation is a constant 0 and the validation is a 2-input NAND gate with inputs A_1 and B_1. ALC **C3** has approximation part consisting of a 2-input NAND with inputs A_0 and B_0, and a 2-input AND as validation. The ALC **C4** approximation output follows the input such that $Z_2 = B_1$ while the validation is a 2-input AND gate with inputs A_1 and A_0. Similarly to the C4 AL in the previous example, **C5**'s approximation section could be either an IOB or an LC with the same resulting ALC. As an IOB, the approximation is $Z_2 = B_1$ whenever $B_1 = 0$, needing just a single INV as the validation. As an LC, the approximation section is a constant 0 whenever $B_1 = 0$, with the same area overhead and input coverage as the IOB case.



AL	Area %	Input Coverage
C1	36%	25%
C2	18%	25%
C3	23%	75%
C4	23%	25%
C5	9%	25%
C6	18%	25%

FIGURE 5.12: Algorithm applied to the output Z_1 of the 2-bit parallel multiplier.

For the application of the algorithm for the output Z_1 shown in Figure 5.12, the **C1** is an SG type generated with a $g = 2$ which synthesizes as a 2-input XOR gate and a validation of another 2-input AND. **C2** is an IOB type where the approximated output mimics A_1 such that $\hat{Z}_1 = A_1$ whenever $A_0 = 0 \& B_0 = 1$ (an INV plus a 2-Input AND). **C3** is an SG type generated consisting of a 2-input AND gate and a validation of another 2-input NAND. **C4**'s approximation is a 2-input NAND gate whose validation consists on a 2-input NAND. The LC type ALC, **C5** is a constant logic 0 when $A_1 = 0 \& B_1 = 0$ (a 2-Input NOR gate). Finally, the approximation part of circuit **C6** is an IOB type where $Z_1 = B_0$ when $A_1 = 0 \& A_0 = 1$, making the validation section an INV and a 2-Input AND gate.



AL	Area %	Input Coverage
C1	14%	25%
C2	**0%	50%
C3	14%	100%
C4	5%	50%
C5	**0%	50%

** Output follows B0 whenever input A0 is 1
This holds true, as the least significant output is a simple AND between A0 and B0

FIGURE 5.13: Algorithm applied to the output Z_0 of the 2-bit parallel multiplier.

Finally, applying the input subspace search algorithm on output Z_0 results in a corner case, as the logic cone of this output in the original 2-bit multiplier consists of a single AND gate. This example shows what happens when the algorithm is applied on the smallest logic cone possible. As shown in Figure 5.13, **C1** is an LC ALC type with $g = 2$ in which $Z_0 = 1$ when $A_0 = 1 \& B_0 = 1$ (2-input AND gate). ALC candidates **C2** and **C5** present an area overhead of 0 when simplified due to the output following B_0 whenever input $A_0 = 1$, which are both just wires. This behaviour is the same as having an AND gate serving as a buffer for one of its inputs, in this case B_0. For ALC **C3**, the approximation section simplifies to a 2-input AND gate while the validation is a constant 1 (always valid), which indicates that this is not an approximation as it replicates the whole Z_0 logic cone, as expected. Finally, circuit **C4**'s approximation is a static 0 and the validation section is a single INV. All of this ALC candidates are equivalent to the single 2-input AND gate of the original logic cone Z_0. This corner case evidences that there is no possible approximation of such a small logic cone.

Table 5.9 presents the two candidates with the highest input coverage for each of the logic cones in the 2-bit parallel multiplier example. The first column is the logic cone (or output), followed by the area overhead and the input coverage. The next column shows the type of ALC approximation used. Finally, the last column references the ALC candidate as shown in the figures of this example. Figure 5.14 shows the original 2-bit parallel multiplier circuit and the selected ALC for each output.

Output	Area Cost (%)	Input Coverage (%)	AL Type	AL Candidate
Z_3	18	100	SG	C1
Z_3	9	75	LC	C2
Z_2	23	75	SG	C1
Z_2	9	75	LC	C2
Z_1	23	75	SG	C3
Z_1	9	25	LC	C5
Z_0*	14	100	SG	C3
Z_0*	0	50	IOB	C2

Table 5.9: Results of the selected ALC candidates for each output of a 2-bit parallel multiplier. *Logic cone consists of a single gate

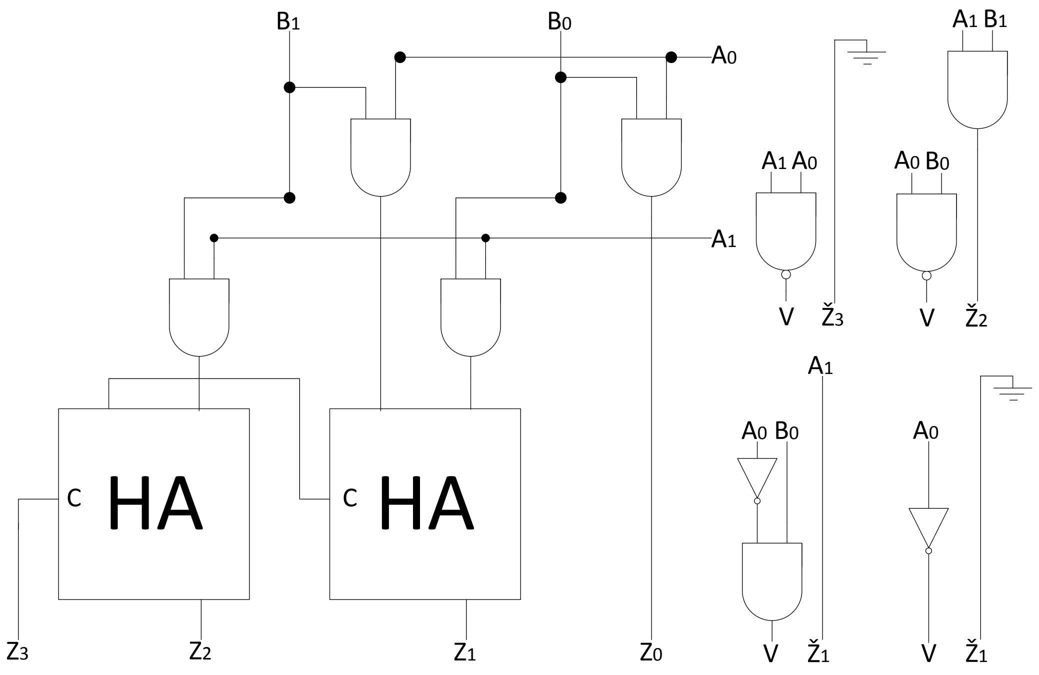


FIGURE 5.14: 2-bit Parallel Multiplier and ALCs for each output.

To demonstrate the scalability of the input subset selection algorithm for ALC generation, it was applied to a 4-bit parallel multiplier with an 8-bit output vector Z and an area of 686 area units (Synthesized using the ABC synthesis tool [121] mapping to a generic MCNC library for quick synthesis and benchmarking). In this example, only output Z_6 was selected as it belongs to the critical path and shares the largest logic cone in the circuit along with Z_7. Figure 5.15 shows the original 4-bit parallel multiplier circuit and three ALC candidates selected for the output Z_6. Table 5.10 summarizes the results of each of the selected ALC candidate. Note that for the same output, various ALC with different area overheads and input coverages can be generated. The ALC candidates presented are some few of the more than 20 possible candidates for this output, all with a different input subspaces.

Output	Area Overhead (%)	Input Coverage (%)	AL Type	AL Candidate
Z_6	1.90	25.00	SG	C1
Z_6	1.31	43.75	LC	C2
Z_6	2.77	21.88	IOB	C3

Table 5.10: Results of the selected ALC for the Z_6 output

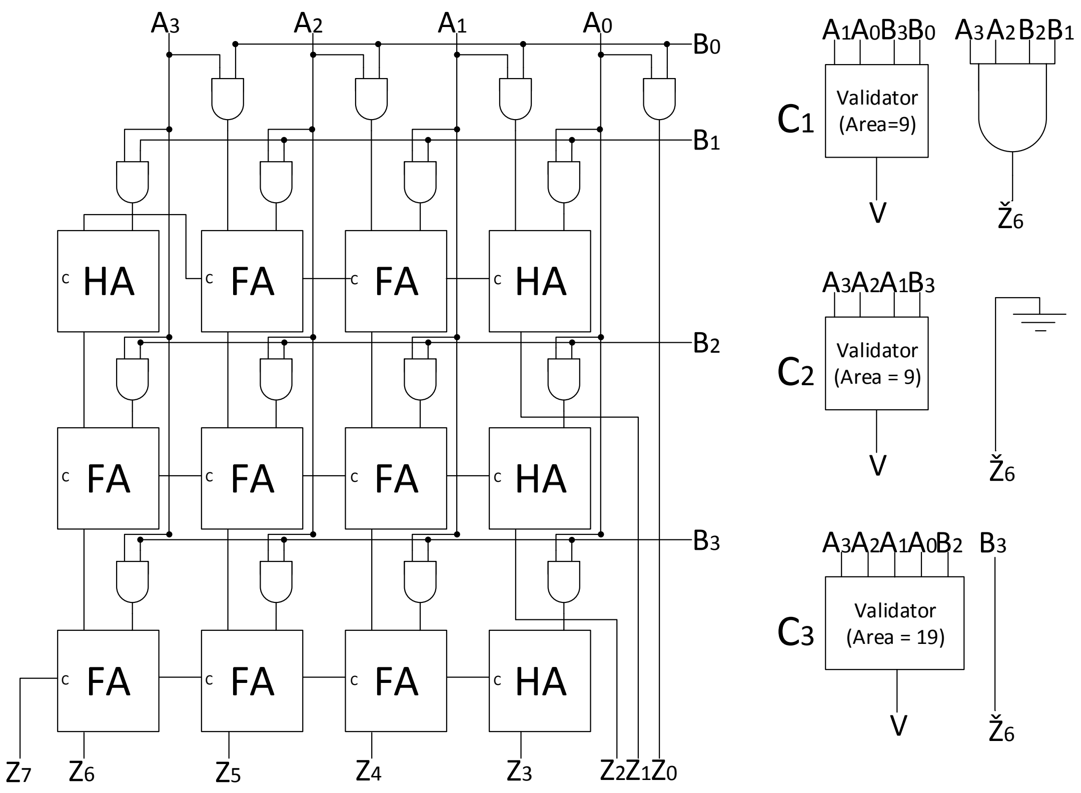


Figure 5.15: Some AL candidates for the Z_6 output.

In the AL **C1** presented in figure 5.15, the SG approximation is achieved with a 4-input NAND gate. The validation section was synthesized in ABC ensuring that $V = 1$ for the input combinations of A_1, A_0, B_3, B_0 where the approximation condition holds true, producing a circuit of 9 area units (mapped to the same generic MCNC library). With the size of a 4-input NAND gate being 4, the total size of the ALC is 13 area units, which results in a total area overhead of 1.90% and an input coverage of 25%. Since **C2** is an LC type approximation, the output behaves as a logic 0 (GND) whenever the valid condition is true. The size of the validation section for this ALC candidate is 9 area units, however, the achieved input coverage is greater than C1, with 43.75% of the original input space covered by this approximation. Finally, ALC candidate **C3** is an IOB type in which the output $Z_6 = B_3$ whenever the valid condition is asserted. The validation section has an area of 19 area units, making it the largest of the three chosen ALC candidates, in addition to being the candidate with the lowest input coverage at 21.88%. Its important to note that due to the cascading nature of the parallel multiplier, either outputs Z_7 or Z_6 make the best logic cones to approximate, as they are both in the critical path of this circuit.

5.3.2 Benchmark t481 Example

The input subset selection algorithm applied to the t481 circuit of the LGSynth'91 benchmarks, which consists of a single logic cone of 388 gates with 16 inputs and 1 output. Benchmark circuit t481 is a randomly generated combinational circuit whose PLA description consists of 481 random input lines and was originally devised as a benchmark for synthesis algorithms. Therefore, this benchmark circuit presents an interesting case for the input subset selection algorithm.

Table 5.11 presents the results of each type of approximate logic circuit for this circuit example. The first column shows the ALC type. Each subscript denotes the subtype of each ALC. For instance, SG_{AND} indicates a single gate type consisting of and AND gate, and a IOB_{INV} indicates an input-output bypass that is inverted, consisting of a single inverter. ALC types $LC_{1,0}$ indicate logic constant 1 or 0. Results were obtained for $g = [2, 3, 4]$ (subset sizes of 4,8,16 respectively). The m show the number of occurrences of a subset with the behaviour associated to that ALC type. The *IC (%)* columns present the input coverage attained, which is calculated by Equation 5.3. Finally, columns *Area (%)* show the area cost, which is mostly dependant on the validation section of the ALC.

$$IC = \frac{m \cdot g}{2^{16}} \tag{5.3}$$

The SG_{XOR} and SG_{XNOR} approximations achieve the same input coverage with slightly different area costs. This is due to the input subsets for which the synthesis tool achieves a different optimization in each case, similar to the IOB_{BUFF} and IOB_{INV} cases. The logic constant LC_1 ALC type results in the highest input coverage for all gs. However, $g = 3$ and $g = 4$ show a smaller coverage due to the inability to find subsets of sizes 8 and 16 in which all the inputs lead to a logic 1. On the other hand, $g = 2$ results in twice as much area cost compared to $g = 3$ and $g = 4$ due to the extra logic required to cover that 2.2% difference in input coverage. No occurrences of SG types were obtained for the larger $g = 3$ and $g = 4$. This indicates that finding behaviour similar to 3-input or 4-input gates is not possible for this circuit.

ALC Type	$g = 2$			$g = 3$			$g = 4$		
	m	IC (%)	Area (%)	m	IC (%)	Area (%)	m	IC (%)	Area (%)
SG_{AND}	540	3.3	3.24	0	-	-	0	-	-
SG_{NAND}	540	3.3	3.24	0	-	-	0	-	-
SG_{OR}	1764	10.8	3.18	0	-	-	0	-	-
SG_{NOR}	1764	10.8	3.18	0	-	-	0	-	-
SG_{XOR}	1960	12.0	3.58	0	-	-	0	-	-
SG_{XNOR}	1960	12.0	3.47	0	-	-	0	-	-
IOB_{BUFF}	2760	16.9	5.34	1260	15.4	3.07	600	14.7	2.67
IOB_{INV}	2760	16.9	5.46	1260	15.4	2.96	600	14.7	2.56
LC_1	8200	50.1	6.88	3920	47.9	3.18	1960	47.9	3.18
LC_0	3576	21.8	5.57	1236	15.1	6.54	600	14.7	3.18

Table 5.11: Results of each type of ALC for the t481 benchmark circuit

5.4 Benchmark Results

Circuit	Area	Inputs	Sel cone size (%)	Area (%)	IC (%)	ALC Type
$9sym$	417	9	100.00	13.9	31.25	LC
$Z9sym$	410	9	100.00	10.5	31.25	SG
$xor5$	20	5	100.00	75	50.00	SG
$clip$	291	9	26.65	1.72	50.00	LC
$con1$	32	7	53.20	6.25	25.00	LC
$rd84$	553	8	45.57	10.85	31.25	SG
$squar5$	93	5	23.37	16.13	50.00	SG
$squar5$	93	5	23.37	6.45	25.00	SG
$x2$	77	10	31.88	2.63	25.00	LC
Average	-	-	-	**7.43**	**33.59**	-

Table 5.12: Results of applying the Input Subset Selection Algorithm on the LGSynth91 benchmark circuits

Table 5.12 shows the results after applying this approximation technique to a set of the LGSynth91 benchmark circuits. The benchmarks were synthesized using the ABC synthesis tool, the area is given in generic area units from the MCNC generic library which is technology independent and contains information on the typical size relations of the most common standard cells, whose basic unit is the size of an inverter. The first column shows the benchmark circuit, followed by the area given in generic area units and the number of inputs of the circuit. The next column, selected cone size, indicates the size of the selected output cone with respect to the size of the whole circuit. For instance circuit $9sym$ consists of a single logic cone, therefore the selected cone for approximation is a 100% of the original circuit. For circuits with multiple outputs, the largest logic cone was selected. The next column presents the area cost of the synthesized approximate logic circuit including both the approximation and the validation section. The IC column shows the input coverage attained by the ALC. Finally, the last column presents the type of ALC that was used. Results show an average area cost of 7.43% with an average input coverage of 33.59%. In particular, the ALC on the approximated logic cone (output) for circuit *clip* shows an area cost of 1.72% with an input coverage of 50%. Figure 5.16 displays the resulting ALCs. For simplicity, the validation section is shown as a block with only the required inputs.

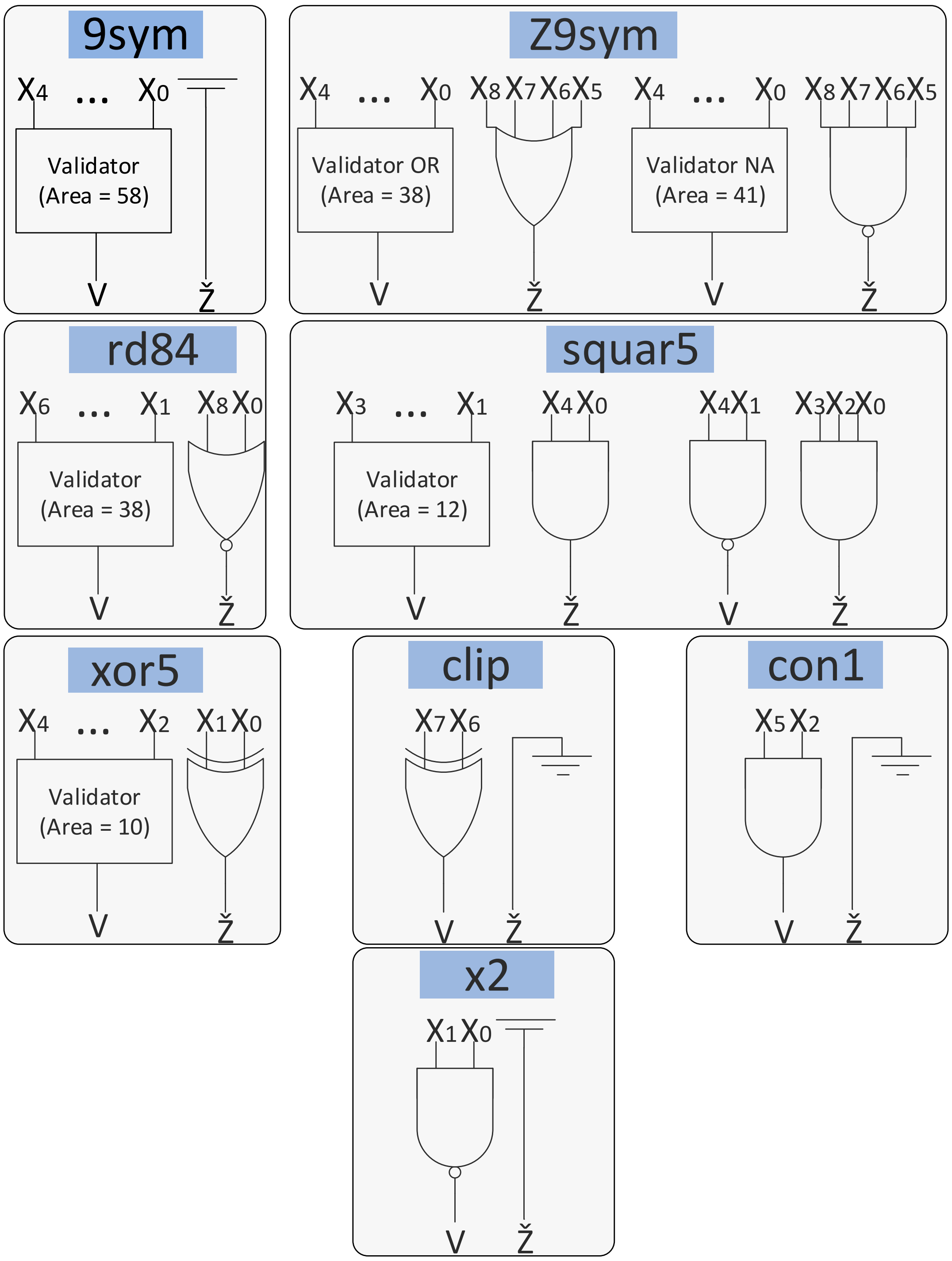


FIGURE 5.16: Approximate Logic Circuits for the LGSynth91 benchmark circuits.

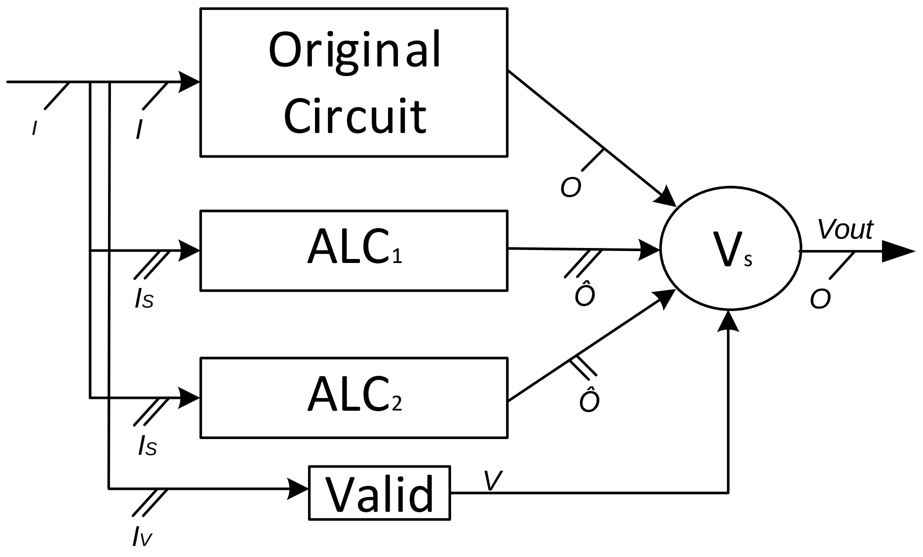


FIGURE 5.17: Selective Fault Tolerance using Approximate Logic Circuits

5.5 Selective Fault Tolerance using Approximate Logic Circuits

Figure 5.17 presents a selective fault tolerance (SFT) system using ALCs generated by the proposed technique. Selective Fault Tolerance (SFT) was proposed as a low cost alternative to TMR by protecting the functionality of the circuit for only a subset of input space [102]. The input subset is selected randomly or arbitrarily by the designer. The input patterns within the subset are ensured to be protected with the same level of reliability of TMR, while the rest are not guaranteed protection. (Please refer to Section 3.2.1 for a more thorough discussion on Selective Fault Tolerance). In the SFT scheme shown in Figure 5.17, ALC_1 and ALC_2 are identical whose inputs (I_s) are only those needed to approximate the output O. Similarly, the validation section receives only the inputs (I_V) required to determine when an input pattern belongs to the selected input subsets.

Table 5.13 shows a comparison between the results reported in [102] and the SFT scheme using ALCs. For the scope of this work, the cost of the voter circuits is ignored. The first two columns present the area cost and the input coverage (IC) results of the original SFT technique. The last two columns show the results of the proposed SFT scheme

Benchmarks	SFT [102]		This technique	
	IC (%)	Area (%)	IC (%)	Area (%)
$xor5$	50%	137%	100%	150%
$Z9sym$	60%	156%	63%	22%
$t481$	50%	153%	50%	7%

Table 5.13: Comparison between SFT and this ALC technique

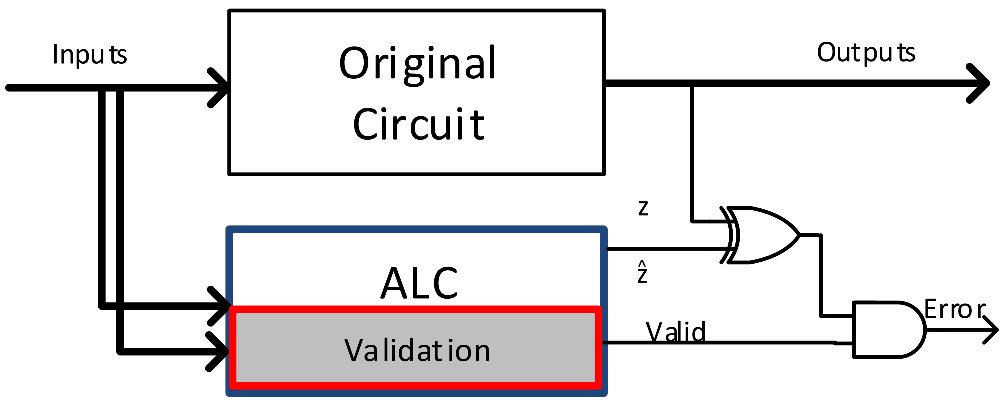


FIGURE 5.18: Concurrent Error Detection using Approximate Logic Circuits.

using ALCs. The results shown for this approximation technique on circuits *xor5* and *Z9sym* were obtained by using two approximations with no input coverage overlap and by adding the area cost of each of the validation sections of the two approximations. For circuit *xor5*, applying the proposed technique results in 13% greater area cost, however, it covers 100% of the input space. In the case of the circuits *Z9sym* and *t481*, the proposed technique offers significant reduction of the area cost associated with covering a similar percentage of the input space.

5.6 Concurrent Error Detection using Approximate Logic Circuits

These type of approximate logic circuits can be used for concurrent error detection of logic errors. Figure 5.18 shows the CED mechanism using ALCs. The ALC approximates the output whenever a pattern belonging to the selected subspace is present at the inputs. If a logic error occurs, the output z and the approximate output $\hat{z}$ will be different, in which case, an error is detected when the Valid signal of the ALC is asserted. Logic errors caused by both transient or permanent faults are detected for all the input patterns of the selected input subset.

The CED technique presented in [147] reports a detection of 45% of errors induced by stuck-at faults for the *x2* benchmark circuit while incurring in 27% of area cost. On the other hand, as evidenced in Table 5.12, the resulting ALC for that circuit shows that logic errors caused by any fault in 31.88% of the circuit can be detected with an input coverage of 25%, incurring in 2.63% of area cost.

5.7 Discussion and Future Work

This technique does not allow the designer to select the input space to approximate, however it is possible to limit the search space of the input subset selection algorithm. This limited search may be performed on an input space delimited by using the probabilistic fault model of output deviations (See Section 3.2.2). The search space can be limited to the patterns that exhibit the highest probability of propagating errors to the output, thus increasing the likelihood of detecting errors and protecting the circuit against them.

Another improvement can be possible by adapting the input subspace selection algorithm to utilise two or more ALCs to approximate the same logic cone. Two or more ALCs with different approximated input subspaces may be combined to increase the input coverage, as was manually done for circuits *xor5* and *Z9sym*. Doing so may offer a trade-off between input coverage and area cost.

5.8 Concluding remarks

This chapter presented a circuit approximation technique that can be used for concurrent error detection of logic errors and for circuit protection. The approximation is based on finding functionality that behaves similarly to logic gates or constant values. An algorithm to automate the input subset selection was proposed. This algorithm was applied to arithmetic circuits of different sizes, and to a set of the LGSynth91 benchmark circuits. Results on these benchmarks show an average input coverage of 33.59% of all the input space with an average 7.43% area cost. Furthermore, using the approximate logic circuits generated by this technique for selective fault tolerance, results in significant area cost reduction compared to an existing technique. The generated approximate logic circuits may also be used for very low cost concurrent error detection. Further exploration of the applications of single gate approximate logic circuits has been outlined.

Chapter 6

Conclusions

With technology scaling, the reliability concerns of integrated circuits are continuously growing. The appearance of logic errors in-the-field caused by faults escaping manufacturing testing, single-event upsets, aging, or process variations, is increasing. Traditional techniques for online testing and circuit protection often require a high design effort or incur in high area overhead and power consumption. The contributions presented in this thesis provided novel low cost techniques for online error detection and fault tolerance to be used in systems where area cost and power consumption are constrained. Each of these techniques offer a novel approach to enhancing circuit reliability. The rest of this chapter is organised as follows. Section 6.1 presents a summary of the contributions made in this thesis and Section 6.2 discusses the possible future research directions of each technique.

6.1 Summary of Research Contributions

The objective of this thesis is to develop low cost fault tolerance & online monitoring techniques that enhance the reliability of ICs in cost constrained systems. The novel techniques proposed in this thesis leverage the trade-off between fault and error coverage vs area overhead and power consumption in order to enable different levels of protection at various costs. Furthermore, the proposed techniques are applicable at the logic level to any circuit and are developed so that they are compatible with the standard electronic design automation (EDA) flow. The following contributions have been made in this thesis:

Chapter 3 presented a novel low cost fault tolerance design technique applicable at the logic level, that selects and protects the most susceptible workload on the most susceptible logic cones of a circuit, by targeting both timing-independent and timing-dependent errors. The workload susceptibility was ranked as the likelihood of any error to bypass the inherent logic masking of the circuit and propagate an erroneous response to its outputs when that workload is executed. The susceptible workload was protected by a partial Triple Modular Redundancy (TMR) scheme. For the evaluation of the fault-tolerance ability of this technique, the surrogate error models of timing-independent errors induced by stuck-at faults and transient input bit-flips were considered. Additionally considered were the timing-dependent errors induced by transition faults and temporary erroneous output transitions. This technique achieves a similar error coverage when protecting the same number of patterns with an average 39.7% area/power cost reduction. Furthermore, it can improve by 95.1% on average the achieved error coverage with a similar area/power cost. Particularly, when protecting only the 32 most susceptible patterns, an average error coverage improvement of 63.5% and 58.2% against errors induced by stuck-at and transition faults were achieved, respectively, compared to the case where the same number of patterns are protected without any ranking. Additionally, this technique produced an average error coverage improvement of 163% and 96% against temporary erroneous output transition and errors induced by bit-flips, respectively. These error coverage improvements incur in an area/power cost in the range of 18.0-54.2%, which corresponds to a 145.8-182.0% reduction compared to TMR.

Chapter 4 presented a novel low cost probabilistic online error monitoring technique that produces an alarm signal when systematic erroneous behaviour has occurred over a predefined time interval. First, an analysis was presented for the online signal probabilities at the outputs of a circuit when the latter exhibits systematic erroneous behaviour. Based on this analysis, a RTL design methodology was proposed for two monitoring architectures in order to observe the online signal probabilities at the most vulnerable logic cones of a circuit. The two architectures offer various trade-offs between error detection latency and area cost. The technique was applied on a set of EPFL and ISCAS benchmark circuits. The proposed technique was shown to be scalable to large designs, as its area cost depends only on the number of monitored logic cones, not on the size of the monitored circuit. It was demonstrated that the proposed monitoring technique achieves, on the largest circuits, an average error coverage of 84.4% and 73.1% of errors induced by intermittent bit-flips and intermittent stuck-at faults, respectively, with an average area cost of 1.66% and an error detection latency in the range of [0.01,3.3] milliseconds.

Finally, Chapter 5 presented a circuit approximation technique that can be used for low cost non-intrusive concurrent error detection and for selective fault tolerance. The circuit approximation is based on finding functionality at the logic level that behaves similarly to single logic gates or constant values. An algorithm is proposed to select the input subsets to approximate. This algorithm is applied to arithmetic circuits as a proof of concept. This algorithm was applied to arithmetic circuits of different sizes, and to a set of the LGSynth91 benchmark circuits. Results on these benchmarks show an average input coverage of 33.59% of all the input space with an average 7.43% area cost.

Additionally, using the generated approximate logic circuits for selective fault tolerance results in significant area cost reductions compared to existing techniques. In addition to that, it is shown that he generated approximate logic circuits can be used for low cost concurrent error detection.

The contributions aforementioned provide three different low cost fault tolerance & error monitoring techniques. The techniques were validated by a number of experiments carried out on a variety of circuits from different benchmark sets. Qualitative and quantitative comparisons between these techniques and the state of the art were performed. Each of the techniques in this thesis introduces a novel concept, such as determining the workload to protect using a probabilistic fault model, the behaviour of online signal probabilities and how to monitor them, and the circuit approximation based on single gates and logic constants. It is desired that concepts introduced in this thesis contribute towards future research efforts in low cost fault tolerance & error monitoring techniques, as well as provide a basis for developing novel circuit approximation methods.

6.2 Future Research Directions

A number of research challenges have been identified during the development of the work presented in this thesis. Following is a brief description of the challenges and the future research directions that may be taken.

Reconfigurable Workload-Driven Selective Fault Tolerance: The technique presented in Chapter 3 offered a novel approach to identify the most susceptible workload in general logic, as the patterns that are most likely to propagate an error. The functionality produced by those patterns was protected using a partial TMR scheme. The protected pattern set however is static, defined at design time and synthesized along with the circuit. An expansion of this approach is possible in FPGAs by profiling the workloads expected to execute in-the-field and reconfigure the circuit protection to protect the set of patterns that are most vulnerable in that current workload.

Using signal probability monitors to detect aging: The signal probability monitors proposed in Chapter 4 were used to monitor logic errors caused by stuck-at faults and bit-flips. However, with technology scaling, aging has been accelerated leading to increase delay which causes timing errors. The signal probability monitors may be used to detect timing errors by identifying the input patterns that sensitize the critical paths of the circuit. When such patterns are present at the inputs, the signal probability monitors are activated. When all the identified input patterns are executed, the signal probability monitors will compare the collected data against an aging-free signature.

Ranking Faulty Devices in Low cost IoT applications using signal probability monitors: The concept of online signal probabilities introduced in Chapter 4 may be applied in a cross-layer low power error detection technique in order to rank devices used in IoT applications based on the amount of errors they exhibit. The monitors may be used to detect systematic erroneous behaviour in cost constrained error-tolerant applications with relaxed error detection latency requirements. On-Chip Signal probability monitors may be used to collect the online signal probability at the outputs of each device concurrently to normal operation. Different from the technique presented in Chapter 4 however, the online signal probability information is transmitted to the backend of the system via IP where the software calculates the SEB exhibited by each device and ranks them accordingly. Such an approach may assist the maintainability planning of large arrays of similar IoT devices.

Limiting the Input Subset Search for the generation of Approximate Logic Circuits to the most vulnerable patterns: The circuit approximation technique presented in Chapter 5 does not allow the designer to explicitly select the input space to approximate, however an it is possible to limit the search space of the input subset selection algorithm. This limited search may be performed on an input space delimited by using the probabilistic fault model of output deviations, which is used to identify the most susceptible input patterns. The search space can be limited to the patterns that exhibit the highest probability of propagating errors to the output, thus increasing the likelihood of detecting errors and protecting the circuit against them.

Expand the set of possible circuit approximations beyond single gates and logic constants: The circuit approximations introduced in Chapter 5 were restricted to single gates, logic constants and input-output bypass. However, this is a limited set of all the possible approximations. As the circuit approximation premise is based on finding functionality that behaves identically to a basic block (i.e. a single gate), more complex blocks can be used. These more complex blocks can be AOI or OAI complex gates, or any other small block with a predefined functionality. These small blocks can be generic so they can be used to approximate any circuit, or be tailored to a specific functionality depending on the target circuit.

Appendix A

In-House Fault Injection Simulator

This appendix describes the In-House Fault Injection Simulator that was developed to assist in the design and evaluation of the techniques presented in this thesis. The Fault Injection Simulator (FIS) consists of several modules enumerated as follows:

1. **Script-Based Stuck-At Fault Injection :** A bash script written to parse a verilog netlist and list all the nodes. Once all nodes have been listed, the script creates a copy of the verilog netlist and overwrites a single node with either a logic 1 or logic 0, therefore injecting a stuck-at-1 or stuck-at-0 at that node. This process is repeated for all nodes for a full stuck-at fault injection. The script also allows to inject stuck-at faults in a partial list of nodes, selected arbitrarily or randomly.

2. **Input Pattern Generator :** A C++ program reads a verilog netlist to determine the number of inputs (*NoInputs*) in the circuit. It then produces a set of random input vectors of size *NoInputs* based on the specified seed. The seed can be either randomised or explicitly defined. Defining a seed and using it allows to repeat experiments with the same generated workload. The generated workloads are passed on to the Logic Circuit Simulator

3. **Logic Circuit Simulator :** A C++ program that reads a verilog netlist and applies the patterns generated by the Input Pattern Generator to the read netlist. The circuit simulator propagates the inputs through the circuit to generate an output. This simulator operates at the logic level, with no technology information. For each input pattern applied, the logic circuit simulator produces a single output vector.

4. **Vulnerable Pattern Generator :** The Vulnerable Pattern Generator uses the input pattern generator and the logic circuit simulator modules. The input pattern generator module produces an input vector, for which one or more bit-flips are forced in a random location. The original input pattern and the bit-flipped pattern are passed to the logic circuit simulator where both output vectors are produced. The error-free output and the bit-flip injected output are compared to determine if there are any differences. If there is a mismatch, the pattern is deemed as vulnerable and is kept for applying during Bit-Flip simulations. Vulnerable patterns are those that completely bypass the logic masking of a circuit, propagating an error from the input to the output.

The FIS was configured in two modes: Stuck-At and Bit-Flip. The Stuck-At mode, shown in Figure A.1 uses the script-based stuck-at fault injection, input pattern generator and logic circuit simulator modules. The stuck-at fault injections is used to injected stuck-at in a node of the netlist. The circuit simulator receives the input vectors from the input pattern generator and simulates both the fault-injected netlists and the original circuit netlists to produce the faulty and fault-free outputs. The output vectors are compared to collect the number of errors observed. This process is repeated for each node of the netlist. The FIS in Stuck-At mode was used to compute the error coverage of errors induced by stuck-at faults presented in Chapter 4.

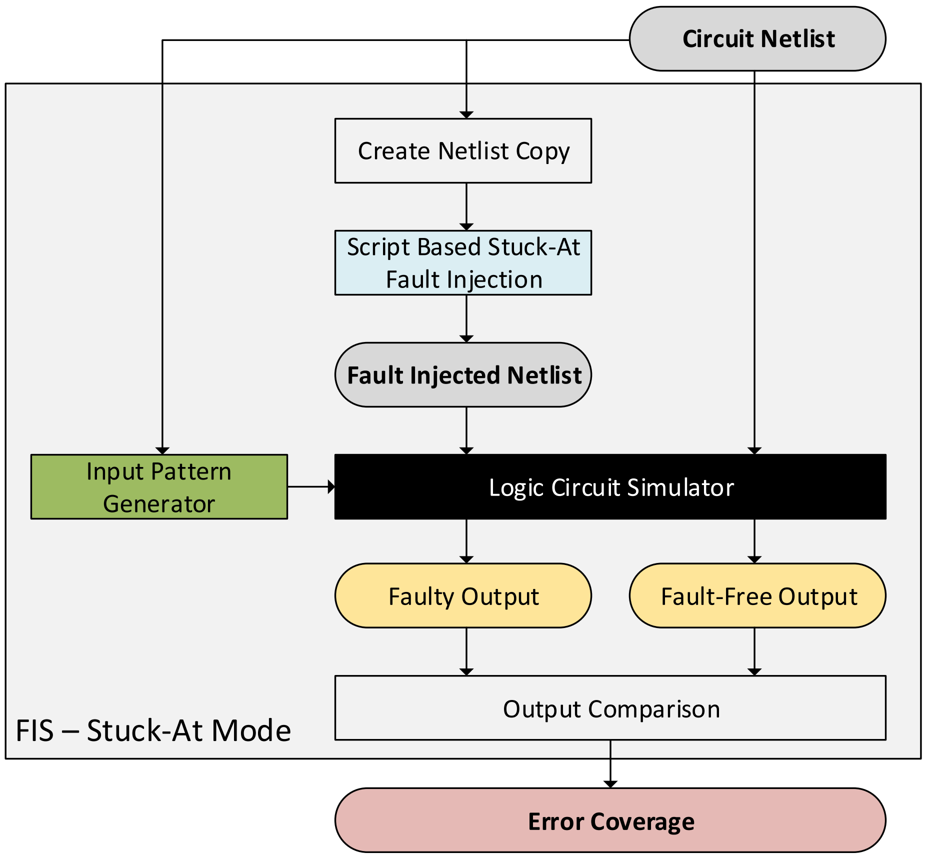


Figure A.1: Fault Injection Simulator for Stuck-At Faults as used in Chapter 4

Figure A.2 shows the Vulnerable Pattern Generator and the FIS in Bit-Flip mode. The vulnerable patterns are passed on to the logic circuit simulator module to be applied to the protected netlist (Section 3.3). Those vulnerable patterns that do not propagate errors on the protected netlist are masked and are counted towards the error coverage. The Vulnerable Pattern Generator was used to compute the error coverage induced by Bit-Flips presented in Chapter 4. The FIS in Bit-Flip mode was used to compute the error coverage induced by Bit-Flips in Chapter 3.

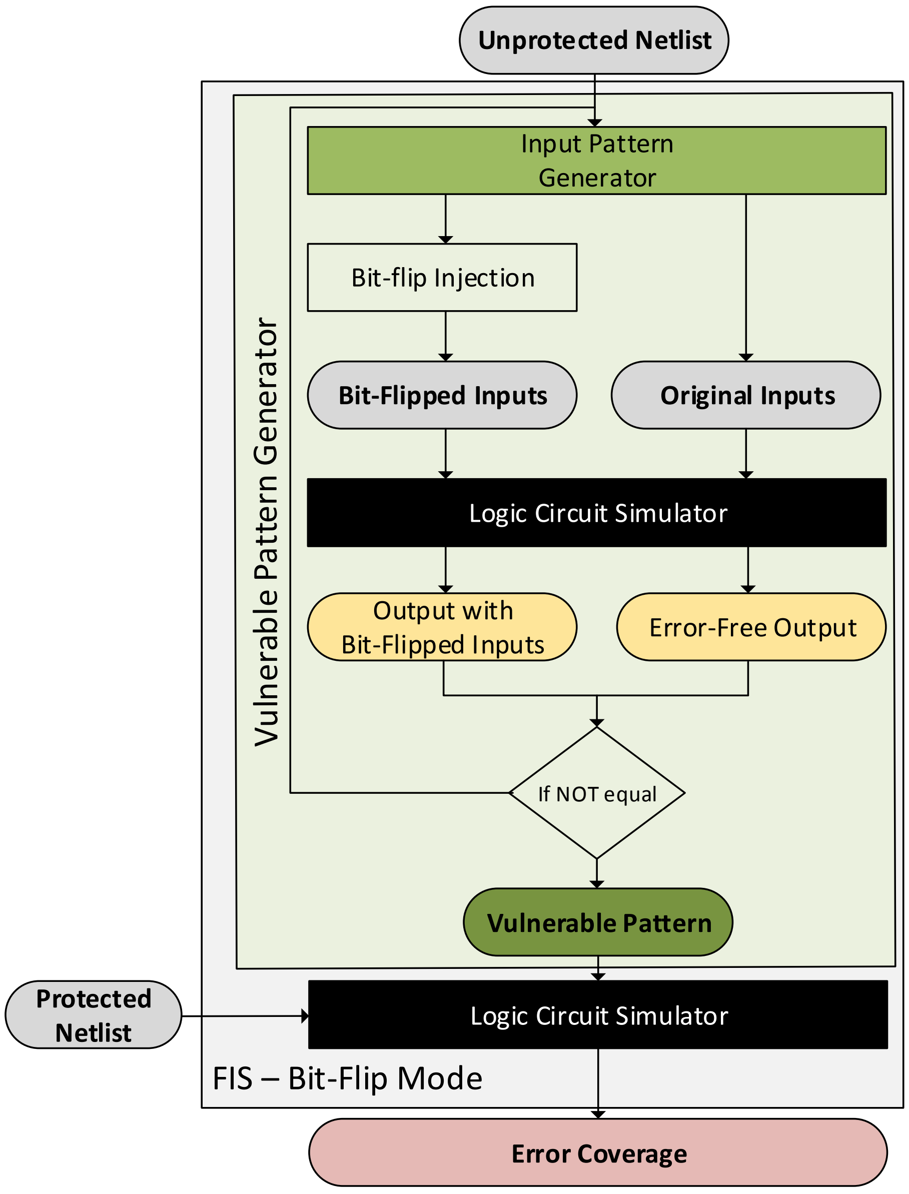


FIGURE A.2: Vulnerable Pattern Generator as used in Chapter 4 and Fault Injection Simulator for Bit-Flips injections as used in Chapter 3

References

[1] S. Borkar, “Design challenges of technology scaling,” *Micro, IEEE*, vol. 19, no. 4, pp. 23–29, Jul. 1999.

[2] G. Moore, “Cramming more components onto integrated circuits,” *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, Jan. 1998.

[3] V. Chandra and R. Aitken, “Impact of technology and voltage scaling on the soft error susceptibility in nanoscale cmos,” in *Defect and Fault Tolerance of VLSI Systems, 2008. DFTVS ’08. IEEE International Symposium on*, Oct. 2008, pp. 114–122.

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

[5] J. Suran, “Effect of circuit design on system reliability,” *Reliability and Quality Control, IRE Transactions on*, vol. RQC-10, no. 1, pp. 12–18, 1961, ISSN: 0097-4552.

[6] S. Pant and D. Blaauw, “Static timing analysis considering power supply variations,” in *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, 2005, pp. 365–371.

[7] R. Baumann, “Soft errors in advanced computer systems,” *Design Test of Computers, IEEE*, vol. 22, no. 3, pp. 258–266, May 2005.

[8] R. Lyons and W. Vanderkulk, “The use of triple-modular redundancy to improve computer reliability,” *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962, ISSN: 0018-8646.

[9] N. Deo, “Generalized parallel redundancy in digital computers,” *Computers, IEEE Transactions on*, vol. C-17, no. 6, pp. 600–600, 1968, ISSN: 0018-9340.

[10] K. J. Gurzi, “Estimates for best placement of voters in a triplicated logic network,” *Electronic Computers, IEEE Transactions on*, vol. EC-14, no. 5, pp. 711–717, 1965, ISSN: 0367-7508.

[11] A. Golnari, Y. Vizel, and S. Malik, "Error-tolerant processors: formal specification and verification," *2015 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2015*, pp. 286–293, 2016. DOI: 10.1109/ICCAD.2015.7372582.

[12] H. Cho, L. Leem, and S. Mitra, "Ersa: error resilient system architecture for probabilistic applications," *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS*, vol. 31, no. 4, pp. 546–558, 2012.

[13] C. Bottoni, B. Coeffic, J. M. Daveau, L. Naviner, and P. Roche, "Partial triplication of a sparc-v8 microprocessor using fault injection," *2015 IEEE 6th Latin American Symposium on Circuits and Systems, LASCAS 2015 - Conference Proceedings*, 2015. DOI: 10.1109/LASCAS.2015.7250415.

[14] I. A. C. Gomes, M. Martins, A. Reis, and F. L. Kastensmidt, "Using only redundant modules with approximate logic to reduce drastically area overhead in tmr," in *2015 16th Latin-American Test Symposium (LATS)*, 2015, pp. 1–6. DOI: 10.1109/LATW.2015.7102522.

[15] T. Arifeen, A. S. Hassan, H. Moradian, and J. A. Lee, "Probing approximate tmr in error resilient applications for better design tradeoffs," *2016 Euromicro Conference on Digital System Design (DSD)*, pp. 637–640, 2016. DOI: 10.1109/DSD.2016.57. [Online]. Available: http://ieeexplore.ieee.org/document/7723610/.

[16] S. Yang, S. Khursheed, B. M. Al-Hashimi, D. Flynn, and S. Idgunji, "Reliable state retention-based embedded processors through monitoring and recovery," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 12, pp. 1773–1785, 2011, ISSN: 0278-0070. DOI: 10.1109/TCAD.2011.2166590.

[17] S. Yang, S. Khursheed, B. M. Al-Hashimi, D. Flynn, and G. V. Merrett, "Improved state integrity of flip-flops for voltage scaled retention under pvt variation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 11, pp. 2953–2961, 2013, ISSN: 1549-8328. DOI: 10.1109/TCSI.2013.2252640.

[18] M. A. Kochte, A. Dalirsani, A. Bernabei, M. Omana, C. Metra, and H.-J. Wunderlich, "Intermittent and transient fault diagnosis on sparse code signatures," *2015 IEEE 24th Asian Test Symposium (ATS)*, pp. 157–162, 2015, ISSN: 10817735. DOI: 10.1109/ATS.2015.34. [Online]. Available: http://ieeexplore.ieee.org/document/7422252/.

[19] *Semiconductor industry association, international technology roadmap for semiconductors 2013*. [Online]. Available: http://www.itrs2.net/2013-itrs.html.

[20] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power cmos digital design," *Solid-State Circuits, IEEE Journal of*, vol. 27, no. 4, pp. 473–484, Apr. 1992.

[21] M. Keating, *Low Power Methodology Manuela For System-On-Chip Design.* Springer, 2007.

[22] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Lifetime reliability: Toward an architectural solution," *IEEE Micro*, vol. 25, no. 3, pp. 70–80, 2005, ISSN: 0272-1732. DOI: 10.1109/MM.2005.54.

[23] S. Yang, W. Wang, T. Lu, W. Wolf, N. Vijaykrishnan, and Y. Xie, "Case study of reliability-aware and low-power design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 7, pp. 861–873, Jul. 2008.

[24] J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor sizing issues and tool for multi-threshold cmos technology," in *Design Automation Conference, 1997. Proceedings of the 34th*, Jun. 1997, pp. 409–414.

[25] T. Sakurai and A. R. Newton, "A simple mosfet model for circuit analysis," *IEEE Transactions on Electron Devices*, vol. 38, no. 4, pp. 887–894, 1991, ISSN: 0018-9383. DOI: 10.1109/16.75219.

[26] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, "Robust system design with built-in soft-error resilience," *Computer*, vol. 38, no. 2, pp. 43–52, Feb. 2005.

[27] L. R. Harriott, "Limits of lithography," *Proceedings of the IEEE*, vol. 89, no. 3, pp. 366–374, Mar. 2001.

[28] S. Bhunia, S. Mukhopadhyay, and K. Roy, "Process variations and process-tolerant design," in *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07)*, 2007, pp. 699–704. DOI: 10.1109/VLSID.2007.131.

[29] A. Sreedhar and S. Kundu, "On modeling impact of sub-wavelength lithography on transistors," in *Computer Design, 2007. ICCD 2007. 25th International Conference on*, Oct. 2007, pp. 84–90.

[30] W. Zhao, F. Liu, K. Agarwal, D. Acharyya, S. Nassif, K. Nowka, and Y. Cao, "Rigorous extraction of process variations for 65-nm cmos design," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 22, no. 1, pp. 196–203, Feb. 2009.

[31] M. Omana, D. Rossi, N. Bosio, and C. Metra, "Low cost nbti degradation detection and masking approaches," *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 496–509, 2013, ISSN: 0018-9340. DOI: 10.1109/TC.2011.246.

[32] R. S. Oliveira, J. Semião, I. C. Teixeira, M. B. Santos, and J. P. Teixeira, "On-line bist for performance failure prediction under aging effects in automotive safety-critical applications," *LATW 2011 - 12th IEEE Latin-American Test Workshop*, vol. 1, pp. 5–10, 2011, ISSN: 15461998. DOI: 10.1109/LATW.2011.5985919.

[33] Y. Wang, S. Cotofana, and L. Fang, “A unified aging model of nbti and hci degradation towards lifetime reliability management for nanoscale mosfet circuits,” *Proceedings of the 2011 IEEE/ACM International Symposium on Nanoscale Architectures, NANOARCH 2011*, pp. 175–180, 2011. DOI: 10.1109/NANOARCH.2011.5941501.

[34] E. Takeda, Y. Nakagome, H. Kume, and S. Asai, “New hot-carrier injection and device degradation in submicron mosfets,” *IEE Proceedings I - Solid-State and Electron Devices*, vol. 130, no. 3, pp. 144–150, 1983, ISSN: 0143-7100. DOI: 10.1049/ip-i-1.1983.0026.

[35] L. Benini, A. Bogliolo, S. Cavallucci, and B. Ricco, “Monitoring system activity for os-directed dynamic power management,” in *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, Aug. 1998, pp. 185–190.

[36] V. Tiwari, R. Donnelly, S. Malik, and R. Gonzalez, “Dynamic power management for microprocessors: A case study,” in *VLSI Design, 1997. Proceedings., Tenth International Conference on*, Jan. 1997, pp. 185–192.

[37] Y.-S. Hwang and K.-S. Chung, “Dynamic power management technique for multicore based embedded mobile devices,” *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 3, pp. 1601–1612, Aug. 2013.

[38] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez, “Reducing power in high-performance microprocessors,” in *Design Automation Conference, 1998. Proceedings*, Jun. 1998, pp. 732–737.

[39] R. Bhanuprakash, M. Pattanaik, S. Rajput, and K. Mazumdar, “Analysis and reduction of ground bounce noise and leakage current during mode transition of stacking power gating logic circuits,” in *TENCON 2009 - 2009 IEEE Region 10 Conference*, Jan. 2009, pp. 1–6.

[40] H. Mahmoodi, V. Tirumalashetty, M. Cooke, and K. Roy, “Ultra low-power clocking scheme using energy recovery and clock gating,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 1, pp. 33–44, Jan. 2009.

[41] A. Ejlali, B. Al-Hashimi, M. Schmitz, P. Rosinger, and S.-G. Miremadi, “Combined time and information redundancy for seu-tolerance in energy-efficient real-time systems,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 4, pp. 323–335, Apr. 2006.

[42] M. Schmitz, B. Al-Hashimi, and P. Eles, “Energy-efficient mapping and scheduling for dvs enabled distributed embedded systems,” in *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*, 2002, pp. 514–521.

[43] M. Schmitz and B. Al-Hashimi, “Considering power variations of dvs processing elements for energy minimisation in distributed systems,” in *System Synthesis, 2001. Proceedings. The 14th International Symposium on*, 2001, pp. 250–255.

[44] Y. Zhang and K. Chakrabarty, “A unified approach for fault tolerance and dynamic power management in fixed-priority real-time embedded systems,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 1, pp. 111–125, Jan. 2006.

[45] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, “A dynamic voltage scaled microprocessor system,” in *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, Feb. 2000, pp. 294–295.

[46] H. Al-Asaad, “Efficient techniques for reducing error latency in on-line periodic built-in self-test,” *IEEE Instrumentation and Measurement Magazine*, vol. 13, no. 4, pp. 28–32, 2010, ISSN: 10946969. DOI: 10.1109/MIM.2010.5521863.

[47] M. A. B. M. Abramovici and A. D. Friedman, *Digital Systems Testing and Testable Design*. IEEE Press, 1994.

[48] P. Liden and P. Dahlgren, “Switch-level modeling of transistor-level stuck-at faults,” in *Proceedings 13th IEEE VLSI Test Symposium*, 1995, pp. 208–213. DOI: 10.1109/VTEST.1995.512639.

[49] Y. Levendel and P. Menon., “Transition faults in combinational circuits: Input transition test generation and fault simulation,” in *Fault Tolerant Computing Symposium. Proceedings*, 1986.

[50] C. T. Glover and M. R. Mercer, “A method of delay fault test generation,” in *25th ACM/IEEE, Design Automation Conference.Proceedings 1988.*, 1988, pp. 90–95. DOI: 10.1109/DAC.1988.14740.

[51] D. Siewiorek, “Fault tolerance in commercial computers,” *Computer*, vol. 23, no. 7, pp. 26–37, Jul. 1990.

[52] K. Gala, D. Blaauw, J. Wang, V. Zolotov, and M. Zhao, “Inductance 101: Analysis and design issues,” in *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, 2001, pp. 329–334. DOI: 10.1145/378239.378501.

[53] J Gracia, L. J. Saiz, J. C. Baraza, D Gil, and P. J. Gil, “Analysis of the influence of intermittent faults in a microcontroller,” in *Design and Diagnostics of Electronic Circuits and Systems, 2008. DDECS 2008. 11th IEEE Workshop on*, 2008. DOI: 10.1109/DDECS.2008.4538761.

[54] J. Gracia-Moran, D. Gil-Tomas, L. J. Saiz-Adalid, J. C. Baraza-Calvo, and P. J. Gil-Vicente, “Defining a representative and low cost fault model set for intermittent faults in microprocessor buses,” *Proceedings - 6th Latin-American Symposium on Dependable Computing, LADC 2013*, pp. 98–103, 2013. DOI: 10.1109/LADC.2013.19.

[55] S. Mitra and E. J. McCluskey, “Which concurrent error detection scheme to choose ?” In *Proceedings International Test Conference 2000 (IEEE Cat. No.00CH37159)*, 2000, pp. 985–994. DOI: 10.1109/TEST.2000.894311.

[56] A. Uhl and J. Becker, “Concurrent error detection in multipliers by using reduced wordlength multiplication and logarithms,” *Proceedings - 16th Euromicro Conference on Digital System Design, DSD 2013*, no. 2, pp. 129–135, 2013. DOI: `10.1109/DSD.2013.22`.

[57] N. a. Touba and E. J. McCluskey, “Logic synthesis techniques for reduced area implementation of multilevel circuits with concurrent error detection,” vol. 16, no. 7, pp. 651–654, 1994, ISSN: 1063-6757. [Online]. Available: `http://dl.acm.org/citation.cfm?id=191326.191609`.

[58] G. Zervakis, N. Eftaxiopoulos, K. Tsoumanis, N. Axelos, and K. Pekmestzi, “A high radix montgomery multiplier with concurrent error detection,” pp. 199–204, 2014. DOI: `10.1109/IDT.2014.7038613`.

[59] M. Maniatakos and P. Kudva, “Low-cost concurrent error detection for floating-point unit (fpu) controllers,” *. . . , IEEE Transactions on*, vol. 62, no. 7, pp. 1376–1388, 2013.

[60] E. Idzikowska, “Concurrent error detection scheme for haf hardware,”

[61] R. M. Sedmak and H. L. Liebergot, “Fault tolerance of a general purpose computer implemented by very large scale integration,” *IEEE Transactions on Computers*, vol. C-29, no. 6, pp. 492–500, 1980, ISSN: 0018-9340. DOI: `10.1109/TC.1980.1675608`.

[62] K. Mohanram and N. Touba, “Cost-effective approach for reducing soft error failure rate in logic circuits,” *International Test Conference, 2003. Proceedings. ITC 2003.*, vol. 1, 2003, ISSN: 1089-3539. DOI: `10.1109/TEST.2003.1271075`.

[63] J. L. A. Hughes, E. J. McCluskey, and D. J. Lu, “Design of totally self-checking comparators with an arbitrary number of inputs,” *IEEE Transactions on Computers*, vol. C-33, no. 6, pp. 546–550, 1984, ISSN: 0018-9340. DOI: `10.1109/TC.1984.1676478`.

[64] E. J. McCluskey, “Design techniques for testable embedded error checkers,” *Computer*, vol. 23, no. 7, pp. 84–88, 1990, ISSN: 0018-9162. DOI: `10.1109/2.56855`.

[65] A. Dutta and A. Jas, “Combinational logic circuit protection using customized error detecting and correcting codes,” in *Proceedings of the 9th International Symposium on Quality Electronic Design, ISQED 2008*, 2008, pp. 68–73, ISBN: 0769531172. DOI: `10.1109/ISQED.2008.4479700`.

[66] M. Nicolaidis, R. O. Duarte, S. Manich, and J. Figueras, “Fault-secure parity prediction arithmetic operators,” *IEEE Design Test of Computers*, vol. 14, no. 2, pp. 60–71, 1997, ISSN: 0740-7475. DOI: `10.1109/54.587743`.

[67] J. M. Berger, “A note on error detecting codes for asymmetric channels,” *Information and Control*, vol. 4, pp. 68–73, 1961, ISSN: 00199958.

[68] P. H. Bardell and W. H. McAnney, “Self-testing of multichip logic modules,” in *ITC*, 1982.

[69] C. E. Stroud, "An automated bist approach for general sequential logic synthesis," in *25th ACM/IEEE, Design Automation Conference.Proceedings 1988.*, 1988, pp. 3–8. DOI: 10.1109/DAC.1988.14726.

[70] A. Steininger and C. Scherrer, "On the necessity of on-line-bist in safety-critical applications-a case-study," *Digest of Papers. Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing (Cat. No.99CB36352)*, pp. 208–215, 1999, ISSN: 07313071. DOI: 10.1109/FTCS.1999.781052. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=781052.

[71] A. Jas, C. V. Krishna, N. a. Touba, and C. Engineering, "Hybrid bist based on weighted pseudo-random testing: a new test resource partitioning scheme," *Program*, pp. 2–8, 2001.

[72] N Mukherjee and R Karri, "Versatile bist: an integrated approach to on-line/off-line bist for data-dominated architectures," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 13, no. 2, pp. 189–200, 1998, ISSN: 10893539.

[73] R. Wang, Z. Zhang, X. Kavousianos, Y. Tsiatouhas, and K. Chakrabarty, "Built-in self-test, diagnosis, and repair of multimode power switches," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 8, pp. 1231–1244, 2014, ISSN: 02780070. DOI: 10.1109/TCAD.2014.2314303.

[74] A. Sanyal, S. M. Alam, and S. Kundu, "A built-in self-test scheme for soft error rate characterization," *Proceedings on 14th IEEE International On-Line Testing Symposium, 2008*, DOI: 10.1109/IOLTS.2008.26.

[75] M. Yoshikawa, S. Okumura, Y. Nakata, Y. Kagiyama, H. Kawaguchi, and M. Yoshimoto, "Block-basis on-line bist architecture for embedded sram using word-line and bitcell voltage optimal control," *Proceedings of the 12th International Symposium on Quality Electronic Design, ISQED 2011*, pp. 322–325, 2011. DOI: 10.1109/ISQED.2011.5770744.

[76] E. K. Moghaddam and S. Hessabi, "An on-line bist technique for stuck-open fault detection in cmos circuits," *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*, no. Dsd, 2007. DOI: 10.1109/DSD.2007.4341532.

[77] G Xenoulis, D Gizopoulos, N Kranitis, and A Paschalis, "Low-cost, on-line software-based self-testing of embedded processor cores bt - 9th ieee international on-line testing symposium, iolts 2003, july 7, 2003 - july 9, 2003," pp. 149–769 519 687, 2003. DOI: 10.1109/OLT.2003.1214382.

[78] I. Voyiatzis, a. Paschalis, D. Nikolos, and C. Halatsis, “R-cbist: an effective ram-based input vector monitoring concurrent bist technique,” *Proceedings International Test Conference 1998 (IEEE Cat. No.98CH36270)*, pp. 918–925, 1998, ISSN: 10893539. DOI: `10.1109/TEST.1998.743284`. [Online]. Available: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=743284`.

[79] I. Voyiatzis, A. Paschalis, D. Gizopoulos, C. Halatsis, F. S. Makri, and M. Hatzimihail, “An input vector monitoring concurrent bist architecture based on a precomputed test set,” *IEEE Transactions on Computers*, vol. 57, no. 8, pp. 1012–1022, 2008, ISSN: 00189340. DOI: `10.1109/TC.2008.49`.

[80] I. Voyiatzis, C. Sgouropoulou, and C. Efstathiou, “A concurrent bist scheme for read only memories,” *2015 10th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, pp. 1–2, 2015. DOI: `10.1109/DTIS.2015.7127366`.

[81] B. Divyapreethi and T. Karthik, “Input vector monitoring concurrent bist architecture using modified sram cells,” *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 9, pp. 4042–4046, 2015, ISSN: 18196608. DOI: `10.1109/TVLSI.2013.2278439`.

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

[83] D.-M. Kwai and B. Parhami, “Fault-tolerant processor arrays using space and time redundancy,” in *Algorithms amp; Architectures for Parallel Processing, 1996. ICAPP 96. 1996 IEEE Second International Conference on*, Jun. 1996, pp. 303–310.

[84] P. Mazumder, “An on-chip ecc circuit for correcting soft errors in drams with trench capacitors,” *IEEE Journal of Solid-State Circuits*, vol. 27, no. 11, pp. 1623–1633, 1992, ISSN: 0018-9200. DOI: `10.1109/4.165344`.

[85] G. E. Sacks, “Multiple error correction by means of parity checks,” *IRE Transactions on Information Theory*, vol. 4, no. 4, pp. 145–147, 1958, ISSN: 0096-1000. DOI: `10.1109/IRETIT.1958.6741947`.

[86] H. Pishro-Nik, N. Rahnavard, and F. Fekri, “Nonuniform error correction using low-density parity-check codes,” *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2702–2714, 2005, ISSN: 0018-9448. DOI: `10.1109/TIT.2005.850230`.

[87] R. W. Hamming, “Error detecting and error correcting codes,” *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950, ISSN: 0005-8580. DOI: `10.1002/j.1538-7305.1950.tb00463.x`.

[88] J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Melliar-Smith, R. E. Shostak, and C. B. Weinstock, "Sift: Design and analysis of a fault-tolerant computer for aircraft control," *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1240–1255, 1978, ISSN: 0018-9219. DOI: 10.1109/PROC.1978.11114.

[89] N. Oh, P. P. Shirvani, and E. J. McCluskey, "Control-flow checking by software signatures," *IEEE Transactions on Reliability*, vol. 51, no. 1, pp. 111–122, 2002, ISSN: 0018-9529. DOI: 10.1109/24.994926.

[90] I. Wali, A. Virazel, A. Bosio, L. Dilillo, and P. Girard, "An effective hybrid fault-tolerant architecture for pipelined cores," in *2015 20th IEEE European Test Symposium (ETS)*, 2015, pp. 1–6. DOI: 10.1109/ETS.2015.7138733.

[91] G. Chen, O. Ozturk, G. Chen, and M. Kandemir, "Energy-aware code replication for improving reliability in embedded chip multiprocessors," in *SOC Conference, 2006 IEEE International*, Sep. 2006, pp. 77–78.

[92] W. Hong, R. Modugu, and M. Choi, "Efficient online self-checking modulo 2n̂+1 multiplier design," *Computers, IEEE Transactions on*, vol. 60, pp. 1354–1365, Sep. 2011.

[93] V. Izosimov, P. Pop, P. Eles, and Z. Peng, "Design optimization of time- and cost-constrained fault-tolerant distributed embedded systems," in *Design, Automation and Test in Europe, 2005. Proceedings*, Mar. 2005, pp 864–869 Vol. 2.

[94] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, vol. January, pp. 29–40, 2003, ISSN: 10724451. DOI: 10.1109/MICRO.2003.1253181.

[95] C. Metra, D. Rossi, M. Omaña, A. Jas, and R. Galivanche, "Function-inherent code checking: a new low cost on-line testing approach for high performance microprocessor control logic," *Proceedings - 13th IEEE European Test Symposium, ETS 2008*, pp. 171–176, 2008. DOI: 10.1109/ETS.2008.24.

[96] I. Polian and J. P. Hayes, "Selective hardening: toward cost-effective error tolerance," *IEEE Design and Test of Computers*, vol. 28, no. 3, pp. 54–62, 2011, ISSN: 07407475. DOI: 10.1109/MDT.2010.120.

[97] K. Mohanram and N. A. Touba, "Partial error masking to reduce soft error failure rate in logic circuits," in *Proceedings 18th IEEE Symposium on Defect and Fault Tolerance in VLSI Systems*, 2003, pp. 433–440. DOI: 10.1109/DFTVS.2003.1250141.

[98] M. Maniatakos and Y. Makris, "Workload-driven selective hardening of control state elements in modern microprocessors," in *2010 28th VLSI Test Symposium (VTS)*, 2010, pp. 159–164. DOI: 10.1109/VTS.2010.5469589.

[99] M. Maniatakos, M. Michael, C. Tirumurti, and Y. Makris, “Revisiting vulnerability analysis in modern microprocessors,” *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2664–2674, 2015, ISSN: 0018-9340. DOI: `10.1109/TC.2014.2375232`.

[100] C. G. Zoellin, H. J. Wunderlich, I. Polian, and B. Becker, “Selective hardening in early design steps,” *Proceedings - 13th IEEE European Test Symposium, ETS 2008*, pp. 185–190, 2008, ISSN: 1530-1877. DOI: `10.1109/ETS.2008.30`.

[101] S. N. Pagliarini, L. A. d. B. Naviner, and J. F. Naviner, “Selective hardening methodology for combinational logic,” in *2012 13th Latin American Test Workshop (LATW)*, 2012, pp. 1–6. DOI: `10.1109/LATW.2012.6261262`.

[102] M. Augustin, M. Gossel, and R. Kraemer, “Reducing the area overhead of tmr-systems by protecting specific signals,” in *On-Line Testing Symposium (IOLTS), 2010 IEEE 16th International*, 2010, pp. 268–273.

[103] M. Augustin, M. Gössel, and R. Kraemer, “Implementation of selective fault tolerance with conventional synthesis tools,” *Proceedings of the 2011 IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2011*, pp. 213–218, 2011.

[104] M. D. Gutierrez, V. Tenentes, and T. Kazmierski, “Susceptible workload driven selective fault tolerance using a probabilistic fault model,” in *22nd IEEE International Symposium on On-Line Testing and Robust System Design 2016. IOLTS. Proceedings. 22th IEEE International*, 2016.

[105] S. Bhunia, S. Mukhopadhyay, and K. Roy, “Process variations and process-tolerant design,” in *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID’07)*, 2007, pp. 699–704. DOI: `10.1109/VLSID.2007.131`.

[106] J. M. Cazeaux, D. Rossi, and C. Metra, “New high speed cmos self-checking voter,” in *On-Line Testing Symposium, 2004. IOLTS 2004. Proceedings. 10th IEEE International*, 2004, pp. 58–63. DOI: `10.1109/OLT.2004.1319660`.

[107] C. G. Zoellin, H. J. Wunderlich, I. Polian, and B. Becker, “Selective hardening in early design steps,” in *2008 13th European Test Symposium*, 2008, pp. 185–190. DOI: `10.1109/ETS.2008.30`.

[108] M. Fazeli, S. N. Ahmadian, S. G. Miremadi, H. Asadi, and M. B. Tahoori, “Soft error rate estimation of digital circuits in the presence of multiple event transients (mets),” *2011 Design, Automation & Test in Europe*, pp. 1–6, 2011, ISSN: 1530-1591.

[109] I. Wali, B. Deveautour, A. Virazel, A. Bosio, P. Girard, and M. Sonza Reorda, “A low-cost reliability vs. cost trade-off methodology to selectively harden logic circuits,” *Journal of Electronic Testing*, 2017, ISSN: 0923-8174. DOI: `10.1007/s10836-017-5640-6`.

[110] Z. Wang and K. Chakrabarty, “Test set enrichment using a probabilistic fault model and the theory of output deviations,” *Proceedings of the Design Automation & Test in Europe Conference*, vol. 1, 2006.

[111] F. J. Ferguson and J. P. Shen, “A cmos fault extractor for inductive fault analysis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 11, pp. 1181–1194, 1988, ISSN: 0278-0070. DOI: 10.1109/43.9188.

[112] Z. Wang and K. Chakrabarty, “Test-quality/cost optimization using output-deviation-based reordering of test patterns,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 2, pp. 352–364, 2008.

[113] ——, “An efficient test pattern selection method for improving defect coverage with reduced test data volume and test application time,” *2006 15th Asian Test Symposium*, 2006, ISSN: 1081-7735.

[114] S. Balatsouka, V. Tenentes, X. Kavousianos, and K. Chakrabarty, “Defect aware x-filling for low-power scan testing,” in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, 2010. DOI: 10.1109/DATE.2010.5456928.

[115] X. Kavousianos, K. Chakrabarty, E. Kalligeros, and V. Tenentes, “Defect coverage-driven window-based test compression,” in *2010 19th IEEE Asian Test Symposium*, 2010, pp. 141–146. DOI: 10.1109/ATS.2010.33.

[116] X. Kavousianos, V. Tenentes, K. Chakrabarty, and E. Kalligeros, “Defect-oriented lfsr reseeding to target unmodeled defects using stuck-at test sets,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 12, pp. 2330–2335, 2011, ISSN: 1063-8210. DOI: 10.1109/TVLSI.2010.2079961.

[117] V. Tenentes and X. Kavousianos, “Low power test-compression for high test-quality and low test-data volume,” in *2011 Asian Test Symposium*, 2011, pp. 46–53. DOI: 10.1109/ATS.2011.75.

[118] ——, “High-quality statistical test compression with narrow ate interface,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 9, pp. 1369–1382, 2013, ISSN: 0278-0070. DOI: 10.1109/TCAD.2013.2256394.

[119] L. Agnola, M. Vladutiu, M. Udrescu, and L. Prodan, “Simplified selective fault tolerance technique for protection of selected inputs via triple modular redundancy systems,” in *Applied Computational Intelligence and Informatics (SACI), 2012 7th IEEE International Symposium on*. DOI: 10.1109/SACI.2012.6250014.

[120] K. Parker and E. McCluskey, “Probabilistic treatment of general combinational networks,” *IEEE Transactions on Computers*, vol. C-24, no. 6, pp. 668–670, 1975, ISSN: 0018-9340.

[121] *Abc: A system for sequential synthesis and verification*, `http://www.eecs.berkeley.edu/~alanmi/abc/`.

[122] J. Gracia-Moran, J. C. Baraza-Calvo, D. Gil-Tomas, L. J. Saiz-Adalid, and P. J. Gil-Vicente, "Effects of intermittent faults on the reliability of a reduced instruction set computing (risc) microprocessor," *IEEE Transactions on Reliability*, vol. 63, no. 1, pp. 144–153, 2014, ISSN: 00189529. DOI: `10.1109/TR.2014.2299711`.

[123] D. Gil-Tomas, J. Gracia-Moran, J. C. Baraza-Calvo, L. J. Saiz-Adalid, and P. J. Gil-Vicente, "Injecting intermittent faults for the dependability assessment of a fault-tolerant microcomputer system," *IEEE Transactions on Reliability*, vol. 65, no. 2, pp. 648–661, 2015, ISSN: 00189529. DOI: `10.1109/TR.2015.2484058`.

[124] M. D. Gutierrez, V. Tenentes, D. Rossi, and T. Kazmierski, "Low power probabilistic online monitoring of systematic erroneous behaviour," in *22nd IEEE European Test Symposium 2017. ETS. Proceedings.*, 2017.

[125] J. Gracia-Moran, D. Gil-Tomas, L. J. Saiz-Adalid, J. C. Baraza-Calvo, and P. J. Gil-Vicente, "Experimental validation of a fault tolerant microcomputer system against intermittent faults," in *2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)*, 2010, pp. 413–418, ISBN: 9781424475018.

[126] D. Gil-Tomas, J. Gracia-Moran, J. C. Baraza-Calvo, L. J. Saiz-Adalid, and P. J. Gil-Vicente, "Analyzing the impact of intermittent faults on microprocessors applying fault injection," *IEEE Design and Test of Computers*, vol. 29, no. 6, pp. 66–73, 2012, ISSN: 07407475. DOI: `10.1109/MDT.2011.2179514`.

[127] J. Paradells, C. Gómez, I. Demirkol, J. Oller, and M. Catalan, "Infrastructureless smart cities. use cases and performance," *2014 International Conference on Smart Communications in Network Technologies, SaCoNeT 2014*, 2014. DOI: `10.1109/SaCoNeT.2014.6867772`.

[128] N. Touba and E. McCluskey, "Logic synthesis techniques for reduced area implementation of multilevel circuits with concurrent error detection," vol. 16, no. 7, pp. 651–654, 1997, ISSN: 1063-6757.

[129] N. Karimi, M. Maniatakos, A. Jas, C. Tirumurti, and Y. Makris, "Workload-cognizant concurrent error detection in the scheduler of a modern microprocessor," *IEEE Transactions on Computers*, vol. 60, no. 9, pp. 1274–1287, 2011, ISSN: 00189340. DOI: `10.1109/TC.2010.265`.

[130] Y. Qassim and M. E. Magana, "Error-tolerant non-binary error correction code for low power wireless sensor networks," *The International Conference on Information Networking 2014 (ICOIN2014)*, pp. 23–27, DOI: `10.1109/ICOIN.2014.6799477`.

[131] C. Zhao, S. Dey, and X. Bai, "Soft-spot analysis: targeting compound noise effects in nanometer circuits," *IEEE Design and Test of Computers*, vol. 22, no. 4, pp. 362–375, 2005, ISSN: 07407475. DOI: `10.1109/MDT.2005.95`.

[132] *The EPFL combinational benchmark suite.* [Online]. Available: `http://lsi.epfl.ch/benchmarks`.

[133] V. Tenentes and X. Kavousianos, "High-quality statistical test compression with narrow ate interface," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 9, pp. 1369–1382, 2013, ISSN: 0278-0070. DOI: `10.1109/TCAD.2013.2256394`.

[134] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," *Proceedings - 2013 18th IEEE European Test Symposium, ETS 2013*, pp. 1–27, 2013. DOI: `10.1109/ETS.2013.6569370`.

[135] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: a survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2016, ISSN: 2168-2356. DOI: `10.1109/MDAT.2015.2505723`. [Online]. Available: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7348659`.

[136] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *2008 Design, Automation and Test in Europe*, 2008, pp. 1250–1255. DOI: `10.1109/DATE.2008.4484850`.

[137] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 48–54. DOI: `10.1109/ICCAD.2013.6691096`.

[138] H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[139] K. Y. Kyaw, W.-L. Goh, and K.-S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in *Electron Devices and Solid-State Circuits (EDSSC), 2010 IEEE International Conference of*, Dec. 2010, pp. 1–4.

[140] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014, pp. 1–4. DOI: `10.7873/DATE.2014.108`.

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

[142] a. Sanchez-Clemente, L. Entrena, M. Garcia-Valderas, and C. Lopez-Ongil, "Logic masking for set mitigation using approximate logic circuits," *Proceedings of the 2012 IEEE 18th International On-Line Testing Symposium, IOLTS 2012*, pp. 176–181, 2012. DOI: `10.1109/IOLTS.2012.6313868`.

[143] A. Thayse and J. Deschamps, “Logic properties of unate discrete and switching functions,” *Computers, IEEE Transactions on*, vol. C-26, no. 12, pp. 1202–1212, Dec. 1977.

[144] A. Sanchez-Clemente, L. Entrena, and M. Garcia-Valderas, “Error masking with approximate logic circuits using dynamic probability estimations,” *Proceedings of the 2014 IEEE 20th International On-Line Testing Symposium, IOLTS 2014*, pp. 134–139, 2014. DOI: 10.1109/IOLTS.2014.6873685.

[145] A. J. Sanchez-Clemente, L. Entrena, and M. Garcia-Valderas, “Partial tmr in fpgas using approximate logic circuits,” *IEEE Transactions on Nuclear Science*, vol. 63, no. 4, pp. 2233–2240, 2016, ISSN: 00189499. DOI: 10.1109/TNS.2016.2541700.

[146] M. Choudhury and K. Mohanram, “Approximate logic circuits for low overhead, non-intrusive concurrent error detection,” in *Design, Automation and Test in Europe, 2008. DATE ’08*, Mar. 2008, pp. 903–908.

[147] ——, “Low cost concurrent error masking using approximate logic circuits,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 8, pp. 1163–1176, Aug. 2013.