Dynamic Reliability Management

Dissertation

A thesis submitted to the
Faculty of Electrical Engineering, Computer Science and Mathematics
of the
Paderborn University
in partial fulfillment of the requirements for the
degree of *Dr. Ing.*

by

Jahanzeb Anwer

Paderborn, Germany
July 2017

Supervisor:
Prof. Dr. Marco Platzner

Reviewers:
Prof. Dr. Marco Platzner
Prof. Dr. Sybille Hellebrand

Additional members of the oral examination committee:
Prof. Dr. Franz Rammig
Prof. Dr. Christian Plessl
Prof. Dr. Christoph Scheytt

Date of submission:
April 3, 2017

Date of public examination:
July 20, 2017

Acknowledgements

First of all, I would like to thank my advisor Prof. Dr. Marco Platzner for supervising, motivating and helping me to solve each of my research problems.
Furthermore, I would like to thank:

- Prof. Dr. Sybille Hellebrand for serving as a reviewer for my dissertation.
- Prof. Dr. Franz Rammig, Prof. Dr. Christian Plessl and Prof. Dr. Christoph Scheytt for serving on my oral examination committee.
- My office-mate Sebestian Meisner for having fruitful discussions with me during numerous phases of my research.
- My colleagues Andreas Agne, Stefan Biedemann, Alexander Boschmann, Stephanie Drzevitzky, Heinrich Giebler, Heiner Giefers, Tobias Graf, Tobias Beisel, Markus Happe, Nam Ho, Server Kasap, Paul Kaufmann, Tobias Kenter, Achim Lösch, Antoniou Paraskewi, Lars Schäfers, Gavin Vaz, Tobias Wiersema and Prof. Dr. Christian Plessl for valuable discussions and making my working environment a pleasant place to conduct research.
- The Paderborn Center for Parallel Computing for providing the compute resources to run my time-consuming simulations.
- The International Graduate School of Dynamic Intelligent Systems (IGS-DIS) for providing funds to conduct this research.

Finally, I would like to thank my family. In particular, I would like to thank my parents for their continuous support and my wife Rija for her encouragement and patience.

Abstract

Radiation-tolerant computing for FPGAs has become an important field of research due to the increased usage of FPGAs in space missions. Various device and design hardening techniques have been used in the past to mitigate errors, particularly single event upsets, that appear due to ionizing radiation particles in aerospace missions. Redundancy is the most commonly used technique to counter such errors. However, the traditional hardware design approaches use statically redundant structures and incur a fixed overhead in performance factors of area consumption, latency and power dissipation. These structures are designed to handle the worst case radiation scenarios. However, it has been shown by experiments, depicting radiation patterns of space, that the radiation strength varies a lot during the operation time span of satellite missions. Therefore, incurring a fixed overhead of static redundant structures for FPGA hardware, results in the wastage of resources as well as performance loss since lower levels of redundancy provide sufficient level of reliability in relatively calm regions of space radiation.

Since high orders of redundancy cost large overheads in performance factors, the best approach of maintaining the reliability-performance tradeoff is to dynamically reconfigure the FPGAs to required reliability levels, based on the radiation strength of the environment. Fortunately, FPGAs provide this level of flexibility since they are run-time reconfigurable. This concept of run-time reconfiguration for reliability has been named Dynamic Reliability Management (DRM) in this research. DRM is a hardware/software co-design approach consisting of design-time and run-time parts. At design-time, several redundant implementations of a hardware design are generated and rated by the performance factors of area, latency, power and achieved reliability. A Pareto-optimization method has been implemented in MATLAB that filters the non-dominated optimal implementations on the basis of three performance factors described above and the reliability magnitude. The run-time tool flow makes use of the filtered non-dominated optimal implementations while storing them as partial bitstreams in its implementation database. During the operation time span of the mission, each of these implementations can be configured on the basis of system constraints and reliability requirements. The decision when to reconfigure the selected implementation can be made on the basis of external, time, cooperative or radiation/error rate based decision mechanisms.

The construction of design-time and run-time tool flows involved using, extending and developing various tools. In particular, the redundancy-insertion tool, i.e., BANL TMR tool

has been extended to generate three more redundancy configurations which was limited to only triplicated voter structure in the past. The reliability computation tool has been developed based on the Boolean difference error calculus model. This model, though taken from the literature, is extended to cover the redundant structures and sequential circuit analysis before automating it as a MATLAB tool. The system-on-chip platform, developed to validate the DRM run-time tool flow consists of standard Xilinx as well as our custom IP cores. However, the DRM applications have been introduced with the concept of parallel computation engines. The parallelism of hardware applications on FPGA made us implement the DRM with an area-bounded FPGA partition while still benefiting from performance improvement in low reliability requirements.
The design-time and run-time parts of DRM have been validated with a number of benchmarks. ISCAS benchmarks having different combinational/sequential circuit architectures have been used to validate design-time tool flow. The variation of performance factors have been observed during this experimentation and various conclusions are drawn on the patterns by which the performance factors vary. To validate the run-time design tool flow, we have developed a system-on chip platform utilizing Xilinx ML605 evaluation board with Virtex 6 FPGA and embedded ReconOS operating system to run benchmark applications. The run-time applications consist of data sorting and matrix multiplication algorithms. These applications are run under a sample radiation profile and the their performance is measured under a run-time reconfigurable platform of ReconOS. Our experiments have shown that the DRM and dynamic run-time reconfiguration concept is up to seven and a half times performance efficient as compared to statically utilized redundant structures.

Zusammenfassung

Strahlungs-tolerantes Rechnen auf FPGAs ist ein wichtiges Forschungsfeld geworden, da die Nutzung von FPGAs in Weltraummissionen stark angestiegen ist. Verschiedene Härtetechniken gegen Strahlung auf Transistor- und Entwurfsebene wurden in der Vergangenheit genutzt um Fehler, insbesondere Single-Event-Upsets welche durch ionisierende Strahlung während Weltraummissionen entstehen, abzuschwächen. Redundanz ist hierbei die am meisten benutzte Technik um solchen Fehlern entgegen zu wirken. Der traditionelle Hardwareentwurfsansatz benutzt statische Strukturen und nimmt somit zusätzliche Kosten in Form von größerer Chipfläche, Latenz und Leistungsaufnahme in Kauf. Diese Strukturen wurden entworfen um auch die ungünstigsten Strahlungsbedingungen zu überstehen. Jedoch haben Experimente zur Intensität der Strahlung im Weltraum ergeben, dass die Strahlungsintensität während einer Satellitenmission starken Schwankungen unterworfen ist. Daher resultiert die Benutzung von statisch allokierten redundanten Strukturen in einer Verschwendung von Ressourcen und Rechenleistung, da während Zeiten mit niedriger Strahlungsintensität geringere Ansprüche an die Fehlertoleranz gestellt werden können.
Da mehrfach redundant ausgelegte Systeme hohe Kosten verursachen, ist der beste Ansatz um den Kompromiss aus Zuverlässigkeit und Leistungsfähigkeit zu erhalten der, den FPGA dynamisch, abhängig von der Strahlungsintensität, auf die benötigte Redundanz zu rekonfigurieren. Glücklicherweise erlauben FPGAs diese Art von Flexibilität durch ihre Fähigkeit zur Laufzeit neu konfiguriert zu werden. Dieses Konzept zur Laufzeit-Rekonfiguration von FPGAs zur Steuerung der Zuverlässigkeit wird in dieser Arbeit "Dynamic Reliability Management"(DRM) genannt. DRM ist ein Hardware/Software Co-Entwurfs Ansatz der aus Entwurfszeit- und Laufzeitteilen besteht. Zur Entwurfszeit werden mehrere redundante Implementierungen eines Hardwareentwurfes generiert und nach ihren Kosten in Fläche, Latenz, Leistungsaufnahme und erreichter Zuverlässigkeit bewertet. Eine Methode basierend auf Pareto-Optimierung wurde in MATLAB entworfen. Diese Methode filtert die nicht-dominierten, optimalen Implementierungen auf Basis der drei oben erwähnten Kostenfaktoren und der Zuverlässigkeit. Die Laufzeitumgebung nutzt nun diese gefilterten, optimalen Implementierungen, in sie sie als partielle Bitströme in der Implementierungsdatebank speichert. Während er Missionslaufzeit kann nun, entsprechend der verfügbaren Systemressourcen und der benötigten Zuverlässigkeit, jede Implementierung in den FPGA geladen werden. Die Entscheidung, welche Implmentierung geladen werden soll, kann auf der Grundlage von externer, zeitbasierter, cooperativer oder strah-

lungsbasierter Mechanismen getroffen werden.
Die Konstruktion der Entwurfs- und Laufzeitwerkzeuge beinhaltete die Benutzung, Erweiterung und Entwicklung verschiedener Werkzeuge. Im besonderen wurde das Werkzeug "BANL TMRtool", welches Redundanz in Schaltungen einfügt, erweitert um noch drei zusätzliche Redundanzkonfigurationen zu erstellen. Zuvor unterstützte es nur triplizierte Voter-Strukturen. Das Werkzeug zur Berechnung der Zuverlässigkeit wurde auf Basis des "Boolean difference error calculus"Models entwickelt. Dieses aus der Referenzliteratur entnommene Model, wurde in MATLAB implementiert und erweitert, und unterstützt nun redundante Strukturen in Schaltungen sowie die Analyse sequentieller Schaltungen. Die System-on-Chip Plattform, welche zur Validierung der DRM Laufzeitumgebung entwickelt wurde, besteht aus standard IP-Cores von Xilinx als auch aus selbst entwickelten IP-Cores. Die DRM Anwendungen wurden für Unterstützung von parallelen Recheneinheiten entworfen und entwickelt. Dank dieser Unterstützung von parallelen Recheneinheiten kann DRM trotz der statischen Größe von FPGA-Partitionen den Durchsatz der Anwendungen in Zeiten niedriger Zuverlässigkeitanforderungen erhöhen.
Die Entwurfszeitwerkzeuge und die Laufzeitumgebung wurden mit einer Reihe von Benchmarks validiert. Die ISCAS Benchmarks implementieren verschiedene kombinatorische wie auch sequentielle Architekturen und wureden benutzt um die Entwurfszeitwerkzeuge zu validieren. Die Veränderung der Kostenfaktoren wurde während der Experimente beobachtet und verschiedene Schlüsse auf die Änderungsmuster wurden gezogen. Um die Laufzeitumgebung zu validieren, wurde eine System-on-Chip Plattform entwickelt, welche das "Xilinx ML605 Evaluation Board", den darauf befindlichen Virtex6 FPGA, sowie das ReconOS Betriebssystem benutzt, um Benchmark Anwendungen ausführen zu lassen. Die Benchmark Anwendungen bestehen aus Sortier- und Matrix-Multiplikations-Algorithmen. Diese Anwendungen werden unter einem beispielhaften Strahlungsprofil ausgeführt und ihre Leistung wird auf der dank ReconOS zur Laufzeit rekonfigurierbaren Plattform ausgeführt. Unsere Experimente haben gezeigt, das DRM und das Konzept der dynamische Laufzeit-Rekonfiguration um einen bis zu siebeneinhalbmal besseren Durchsatz erlauben als statisch allozierte redundante Strukturen.

Contents

CHAPTER 1

Introduction

1.1 Motivation

The integration density of transistors on a single system-on-chip has tremendously increased in the last couple of years [1]. The packing of more transistors on a chip has been made possible due to shrinking of device dimensions and modern fabrication techniques in semiconductor technologies. However, the critical charge required to flip a logic bit, in the advent of an error, has decreased for the newer scaled technologies thereby increasing the probability of transient errors in logic devices. Therefore, the signal integrity of the electronic devices, has to be maintained nowadays, under low signal to noise ratios and power supply voltages.
There are various types of errors encountered in electronic devices, which can be mitigated using semiconductor fabrication techniques or circuit design approaches at the software or hardware level. The major sources of noise or errors are dealt with at the transistor/analog design layer, e.g., charge sharing/leakage, power supply noise or aging mechanisms like negative temperature bias instability (NBTI) or hot carrier injection (HCI), etc. Another specific category of errors is radiation-induced errors which are encountered in high radiation particularly space environments. The common error-mitigation technique for radiation-induced errors is redundancy. Redundancy, as the name implies, computes the functional output via redundant blocks and compares them before providing a reliable voted output. In this way, a possible error in either of the replicas can be mitigated. However, redundancy in a circuit can be implemented using different granularity levels and voter-insertion algorithms. However, each of the redundant implementation differs in performance factors of area consumption, latency, power dissipation and achieved reliability level.
The space computing, nowadays, is largely dominated by the field-programmable gate arrays (FPGA). FPGAs can perform various tasks during different phases of a mission without the need for holding dedicated resources for each task. This, in turn, reduces the

carry-on hardware and weight of the satellite payload, while providing the feasibility to shut-down circuit modules which are not in use to prevent excessive power dissipation. Moreover, increase in on-board processing requirements on space missions, for various image processing applications, go well with the highly parallel architecture of FPGAs [2]. Most importantly, FPGAs allow spacecraft designers to upload new configuration data (or modify the hardware) after launch in case the mission requirements change or an error is found in application/task design [3]. However, when FPGAs are exposed to space environment consisting of high solar and cosmic radiation, involving high energy electrons, alpha particles and heavy ions, errors in the form of logic reversals appear in the digital circuit elements. These errors, which mainly consist of single event upsets (SEU), could be as disastrous as causing a system level failure or as moderate as internally masked errors. In general, high levels of redundancy, with big replication factors, cost large overheads in performance factors. The traditional way of implementing redundancy in a hardware design is to utilize a fixed redundant structure and bear a constant overhead in circuit and system performance. However, the benefit of using FPGAs is that we can modify the redundant structures at run-time due to the *reconfigurability* of FPGAs. Hence, a suitable trade-off between reliability level and system performance can be maintained while the system is in operation. The idea of the run-time modification of redundant structures goes well with the FPGAs used in space computing since the radiation environment varies during the orbital time span of the space mission. In a particular case study of radiation pattern of a highly elliptical orbit, it has been shown that the redundancy requirement is high only for a very small duration of circuit operation when the satellite passes the Van Allen radiation belts [2]. Therefore, during lower levels of radiation, smaller redundancy levels could be used and vice versa so that a suitable trade off between reliability and performance can be maintained during mission time span.
The mechanism of adaptive reconfiguration of reliability has been proposed in various research works in literature, however, the scope of these research works differ. In this thesis, we propose our technique, named Dynamic Reliability Management (DRM). DRM employs a decision mechanism which tells the system when to reconfigure for higher or lower reliability levels. DRM is composed of two parts. At the design-time, several redundant implementations of a circuit are analyzed for performance factors of area, latency, power and achieved reliability. At run-time, ReconOS, which is an operating system for reconfigurable logic cores, is utilized to switch among various redundant implementations at run-time. The circuit design has the liberty to utilize any decision mechanism for reconfiguration with ReconOS, i.e., external, time, cooperative or radiation/error rate measurements.

1.2 Contributions of the Thesis

This thesis contributes by presenting a novel concept of Dynamic Reliability Management (DRM). In order to realize this concept on both hardware and software, we have utilized, extended and developed various tools. The resulting tools are organized as design-time and run-time tool flows. We explain each of the contributions involving these tools as follows.

- In order to generate various redundant implementations of a digital circuit design, we have utilized BYU-LANL TMR tool. The tool, by default, supports only triple modular redundancy (TMR) as the basic redundant implementation. However, we require more configurations involving higher levels of redundancy and different voter structures, to construct a larger design space for analyzing performance parameters of redundant circuits. Therefore, we have modified this tool to generate three more configurations, i.e., one-alternate-voter, two-alternate-voters and cascaded TMR, extending the design space to 32 implementations based on overall four redundancy configurations and eight voter-insertion algorithms.

- While the performance factors of area, latency and power could be evaluated by standard FPGA softwares, there exists no standard tool for calculating magnitude of reliability. Following the need for a generic tool based on FPGA based circuits, we have developed a MATLAB based tool that takes input and error probabilities of FPGA components and provides us the reliability of circuit outputs. Though the original non-automated reliability model exists in literature, we have extended it in two directions. Our tool not only helps us in evaluating redundant implementations for reliability, it makes us analyze the impact of different orders of redundancy on the circuit reliability. The most important contribution of this analysis is the definition of a threshold point. The threshold point refers to the magnitude of component error-probability after which redundancy serves no improvement in circuit reliability, in fact, it degrades it.

- After extending the BYU tool and developing MATLAB reliability tool, we have additionally used Xilinx synthesis, mapping, placement and routing and power analyzer tools to generate DRM design-time tool flow. The tool flow takes FPGA structural netlist as input and generates non-dominated redundant implementations of a circuit based on performance parameters of area, latency, power and reliability.

- The run-time tool flow of DRM has been constructed using ReconOS operating system involving a system-on chip platform on Xilinx ML605 evaluation board equipped with Virtex 6 FPGA. The system-on-chip (SoC) platform supports DRM by utilizing the non-dominated redundant implementations (stored in the external DRAM) and developing decision mechanism in software or hardware.

- The partial reconfiguration feature of FPGAs has been extended with the parallelism concept of hardware threads. This extended feature is required so that the maximum size of the hardware partition, which needs to be fixed to a maximum implementation size in order to exercise partial reconfiguration, can also be utilized by parallel versions of the smaller implementation sizes. In this way, we can avoid the non- or partial-utilization of large reconfigurable hardware partitions when utilized by comparatively smaller hardware threads. Additionally, the parallelism strategy provides us higher performance of hardware threads.

- Both of the design-time and run-time tool flows have been verified by a number of benchmark circuits. The experimentation in design-time tool flow makes us observe the pattern by which the performance factors of the redundant circuit vary in the placed and routed design. The run-time tool flow, while being verified by sorting and matrix multiplication case studies shows that our proposed DRM technique can provide up to seven and a half times higher performance when compared with static reliability management techniques with fixed redundant structures.
- This research has so far resulted in four published international conference publications [4, 5, 6, 7], one published journal publication [8] while another journal publication is under peer-review process of a respective journal.

1.3 Thesis Outline

The thesis is structured as follows:

Chapter 2 provides the background of fault-tolerance in FPGA based hardware designs used in space computing. It starts by explaining the space environment and different radiation scenarios. The radiation based errors are categorized in different types along with their impacts. Afterwards, it provides the literature on approaches used in radiation tolerant computing for FPGAs. Furthermore, a short overview of adaptive reliability techniques are presented which work on the broad research line of FPGA based fault-tolerance though differing from DRM in scope and implementation.

Chapter 3 explains the development of our reliability evaluation tool, i.e., Boolean Difference Error Calculator (BDEC) on the MATLAB software. Firstly, the conventional BDEC model is explained. Afterwards, we explain the limitations of this model and the extensions made to it in order to perform our simulations. The extended model is afterwards automated in MATLAB. This chapter presents the first experimentation of this tool, i.e., parameter variation analysis of control parameters of the BDEC model.

Chapter 4 presents the tool flow of the design-time and run-time parts of DRM. This chapter discusses the tools that we have utilized, extended and developed for the realization of DRM on software and hardware platforms.

Chapter 5 provides the experiments that we have conducted on various benchmarks to validate our DRM tool flow. The design-time tool flow is validated by analyzing area, latency, power and reliability of six ISCAS benchmarks of different circuit architectures. The run-time tool flow, on the other hand, has been validated by two practical case studies on system-on-chip platform for DRM, which is implemented on Xilinx ML605 evaluation board.

Chapter 6 concludes our research as well as explains our future research directions.

CHAPTER 2

Background and Related Work

The signal integrity of electronic devices, reaching nanometer dimensions, has become a serious concern due to numerous device architecture issues, e.g., ground bounce, cross-coupling, charge sharing/leakage, capacitive/inductive coupling, process variations, aging mechanisms, etc [9, 10]. Additionally, ICs operating in space environment suffers from radiation induced errors, being permanent or transient in nature. The understanding of radiation environment is essential in designing reliable space electronic systems.

2.1 Space Radiation Environment

The radiation environment of the earth is composed of various kinds of radiation particles, with different ionizing strengths, and can lead to different effects in electronic devices. They majorly include protons, electrons, heavy ions and electromagnetic radiation (photons). It is important to understand the radiation environment of the space or orbit in which the electronic system is designed to operate, in order to take the reliability measures, e.g., shielding or device-based fault-tolerance methods.

2.1.1 Earth's Magnetosphere

The magnetic field surrounding the Earth is called as magnetosphere, depicted in Figure 2.1 [11]. The magnetic field of the earth would have been a dipole if it were not affected by the solar wind coming from the sun. The influence of solar wind shapes the Earth's magnetic field with compressed magnetosheath (the part closer to the sun) and lengthened magnetotail (the part furthest from the sun) [12]. The path of the satellite determines which types of radiation the comprising electronic components could encounter. The orbits closer to earth might experience only few low-energy particles since the magnetosphere blocks or attenuates most of the radiation particles. The higher orbits could not only experience radiation particles from solar wind but also from deep space. Furthermore,

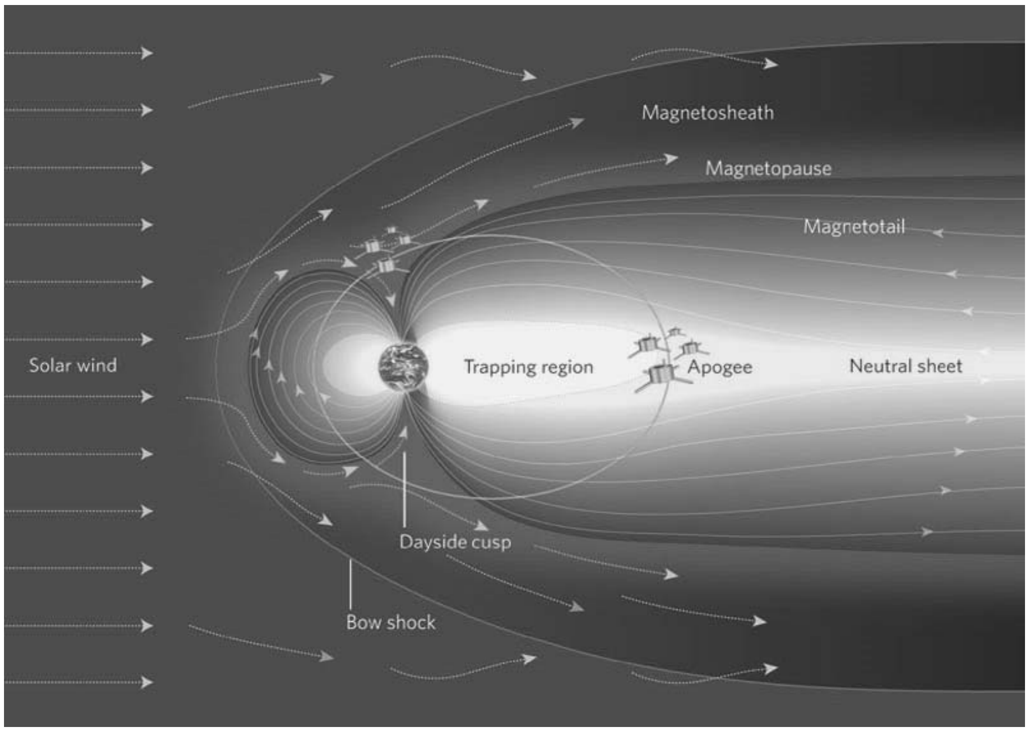


Figure 2.1: Earth's Magnetosphere [11]

the time and duration of the space mission should be considered too since the shape of the Earth's magnetosphere changes due to varying solar wind and radiation patterns.

2.1.2 Ionizing Space Radiation

The ionizing space radiation refers to the radiation particles having sufficient energy to remove electrons from the orbits of atoms thus resulting in charged particles. They can be classified into three domains.

Van Allen Radiation Belts

The earth is encircled by two radiation belts called as inner and outer Van Allen radiation belts. These belts consist of trapped radiation particles; inner belt contains mainly protons (10-100 MeV) while outer belt contains mainly electrons (up to 7 MeV) [13]. These particles are trapped due to magnetospheric force but they keep on entering or ejecting this force due to varying solar activity. In particular, the South Atlantic Anomaly (SAA) [12] is the area where the Van Allen radiation belts penetrate closer to the earth than other regions, hence poses more danger to the on-board electronics as well as a compromise on the reliability of transferred data.

Cosmic Rays

These are the radiation particles which originate from the sun but exists outside our solar system (Galactic). Though these particles have low flux and are much sparser than the Van Allen belt particles, they have very high energy and are difficult to be shielded. They mainly consists of protons and heavy ions.

Solar Particle Events

The sun has an 11 year cycle during which the solar activity greatly varies. During this period, the sun ejects protons, electrons, heavy ions, etc. The strength and number of these particles depends on the solar conditions, e.g., massive amounts of radiation during *solar maximum* period or relatively very quiet during *solar minimum* period.

2.2 Effects of Radiation on Electronic Devices

The radiation particles affect the functionality of electronic devices including FPGAs. Among the radiation particles encountered in space computing, protons (85% of the galactic radiation) and heavy ions (highly energetic particles, up to GeV) are the major sources of errors [12, 13]. In contrast, alpha particles (less penetrative) and gamma rays (lightly ionizing) are no longer a big concern in reliable space computing. Generally, the effects of radiation can be classified into two domains, cumulative and single event effects.

2.2.1 Cumulative Effects

The cumulative effects are the long-term destructive effects on a device and can be further divided into two types.

Total Ionizing Dose

The amount of radiation that a device can absorb before its transistors begin to degrade is called Total Ionizing Dose (TID) [14]. TID refers to the accumulation of charge in transistor's oxide region which causes an increase in current leakage and power dissipation as well as changes in threshold voltage and timing performance of the device. TID is measured in units of rad (radiation absorbed dose) where 1 rad is equal to 10nJ of energy deposited per gram of material, typically Silicon. The radiation hardened FPGAs, are rated by the total ionizing dose to show their immunity to radiation for space missions, e.g., 1 Mrad for Xilinx space grade Virtex-5QV FPGA [15].

Displacement Damage

This phenomenon refers to the damage done to the device when incoming radiation particles displace atoms from their original lattice positions [12]. It often leads to a chain displacement reaction, when one atom, when displaced, further displaces the neighboring atoms unless not enough energy is left to cause further displacement. However, this is an insignificant effect in SRAM based FPGAs.

2.2.2 Single Event Effects (SEE)

The single event effects (SEE) are the electrical disturbances caused by the ionization of the Silicon lattice by the incident charged particle [16]. The minimum amount of charge that a particle must deposit to change the state of a logic element is called as critical charge, Q_{crit}, defined by Equation 2.1.

$$Q_{crit} = C_{node} * V_{node} \tag{2.1}$$

where C_{node} is the capacitance between transistor nodes and V_{node} is the transistor operating voltage. With every new transistor technology, the operating transistor voltage decreases as well as node capacitance due to its smaller size, thus decreasing the critical charge needed to cause a functional error in a logic device [17]. SEEs are instantaneous effects and classified into non-destructive and destructive types [9, 12, 13, 18].

Non-destuctive SEEs

These include single event upsets (SEU), single event transients (SET), single event functional interrupts (SEFI) and multiple bit upsets (MBU).

- **Single Event Upsets (SEU):** These errors appear when a radiation particle hits a storage element, e.g, a latch and deposits sufficient charge, i.e., critical charge, to change the output state from 0 to 1 or vice versa. This is the most frequent source of errors in digital circuits, hence our further work, including error-mitigation techniques, will mainly focus on SEUs.

- **Single Event Transients (SET):** These are temporary logic level glitches in combinational elements of the circuit. They are typically non-effective unless the short surge of the transient latches with the following memory element, where it behaves like an SEU. With increasing clock speed of the logic circuits nowadays, the probability of latching up SETs is increasing too [12].

- **Single Event Functional Interrupts (SEFI):** These are errors appearing in circuit's control logic, e.g., power-on/reset circuitry. Typically, when these critical bits get upset, the FPGA must be re-configured via pulsing the PROG pin or cycling power, thus causing the outage of FPGA operation for tens or hundreds of milliseconds [19]. Typical SEFIs named during radiation testing of Xilinx Virtex-5QV FPGA include power-on-reset (POR), SelectMAP (SMAP), frame address register (FAR) and global signal SEFIs [19].

- **Multiple Bit Upsets (MBU):** A single particle can cause multiple bit SEUs when the incident path is non-orthogonal to the device. A similar scenario happens when an SET is latched into multiple registers, e.g., via select pin of a multiplexer.

Radiation testing of Xilinx FPGAs shows that each successive Virtex FPGA family is more susceptible to MBUs than previous models [20, 21].

Destructive SEEs

These SEEs cause a permanent failure to the device in contrast to the temporary errors in non-destructive category. They are divided into following three types.

- **Single Event Latchup (SEL):** This condition occurs when a radiation particle causes a low-impedance path between power rails of the MOSFET, where the parasitic NPN/PNP transistors are put into positive feedback condition (PNPN). The resulting runaway current damages the device.
- **Single Event Burnout (SEB):** Primarily caused by heavy ions, SEB occurs when an ion passing through the device in its off state generates a plasma filament of electron-hole pairs in its path. The runaway current generated in this path can trigger a secondary breakdown in the parasitic transistor in the MOSFET.
- **Single Event Gate Rupture (SEGR):** This effect occurs when the path of the heavy ion breaks down the gate oxide. This happens as the electrons in the generated electron-hole pairs diffuse faster than holes, whereby the accumulated holes create a transient field which exceeds the breakdown voltage of the gate oxide. This phenomenon is more probable with thin gate oxides.

2.3 Radiation-induced-error Mitigation Techniques

The conventional radiation/error-tolerant technique is shielding, usually of Aluminum sheet/coating, that is good for absorbing low energy particles, up to 30 MeV. Though the heavy ions were unstoppable in previous times too, the high transistor dimensions were less vulnerable to radiation effects. Due to the shrinking of device dimensions and scaling of power supply and threshold voltages, fault-tolerance measures in addition to shielding is the requirement of time. The two major approaches to make digital circuits radiation tolerant are via device and design.

2.3.1 Radiation Hardened by Device

The digital circuits can be made radiation hardened by using different fabrication/process techniques for their transistors. This includes the prominent silicon-on-insulator (SOI) technology [9]. Following Figure 2.2, we can see that the conventional CMOS technology when used with silicon-on-insulator technique, i.e., an insulator on bulk silicon, the number of generated electron-hole pairs and the ion trail length reduces. Hence, the runaway current (or opposing electric fields causing dielectric breakdown) can be reduced. These techniques can particularly increase the TID and lowers SEL effects though they are expensive techniques especially for low-volume IC production. Therefore, while utilizing

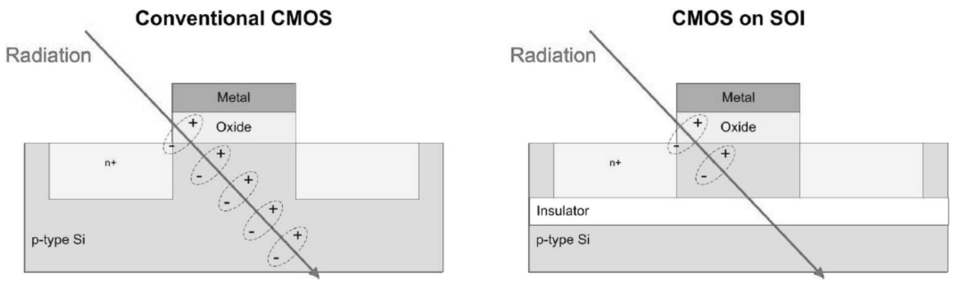


Figure 2.2: Silicon-on-insulator technology [9]

the conventional fabrication process but utilizing architecture-based mitigation techniques makes radiation hardening *by design* an alternative.

2.3.2 Radiation Hardened by Design

The architecture based fault-tolerance detects and corrects single event effects, whether implemented in the software or hardware design. The conventional and popular techniques that fall into this category are redundancy and error detection and correction (EDAC). Though EDAC is mainly used for memory protection against SEEs; in this thesis, we will focus on redundancy and its various implementations in hardware design.

Redundancy

As the name implies, redundancy refers to performing a computing task more than once and compare the outputs of the redundant modules. The most popular form of redundancy is triple modular redundancy (TMR) [22, 23, 24]. In principle, TMR instantiates three copies of an identical circuit and places a voter module at the end to take a majority decision for each output. This concept is illustrated in Fig 2.3(a). The TMR equation for a single voter module is represented as Equation 2.2 where R_1, R_2 and R_3 refer to reliability of the three redundant modules respectively and R_{voter} represents the reliability of the voter itself.

$$R_{out} = R_{voter}[R_1R_2 + R_1R_3 + R_2R_3 - 2R_1R_2R_3] \tag{2.2}$$

The problem with this architecture is the single point of failure, i.e., an error occurring in the voter renders the TMR technique useless. To avoid the single point of failure, the voter can also be triplicated in addition to the triplicated logic modules, as shown in Figure 2.3(b), and the three outputs for each module are run in parallel unless the output has to be merged, to communicate as a single output, via converging voters.
The optimal length of each branch after which a voter can be placed depends of the required granularity level of redundancy. A coarse grained redundancy simply requires triplication of whole circuit and places a single or triplicated voter at the end. A rather fine-grained

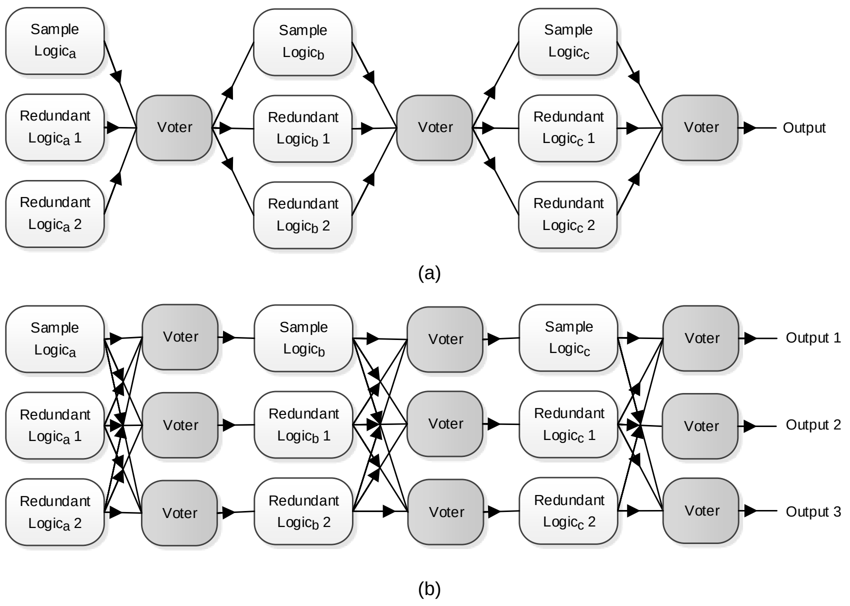


Figure 2.3: Triple modular redundancy with a) single voter b) triplicated voter

redundancy implies triplication of individual logic gates or blocks, e.g., adders or multipliers. The trade-off of TMR reliability is the degradation of circuit's performance factors, i.e., area consumption, latency and power dissipation. The scaling factor of these performance factors depend of the placement and routing of individual circuit elements/blocks in a system design.

Variation in Voting Structures of TMR and Cascaded TMR

Besides the conventional single and triplicated voter configurations, there are configurations proposed in [25] with single/double voters in the alternate stages as shown in Figure 2.4. In contrast to the triplicated voter structure, it takes up to maximum two and one extra voter stages to correct an error for one-alternate and two-alternate voter configurations respectively. Using Monte Carlo simulations, the authors proved that these alternate configurations are slightly less reliable than triplicated voter configuration though they save the overhead of increased number of voters. Hence, in situations where the radiation environment is not very strong and area consumption is an important issue, the alternate configurations are highly useful. Additionally, there is a concept of Cascaded TMR which results in more reliable configurations than TMR, though consuming more area [26]. As an example, we illustrate level-1 CTMR as shown in Figure 2.5. It can be noted that the CTMR configuration has a single point of failure as it is the extension of TMR with a single voter. While CTMR can also be improved by using the triplicated voter strategy, we assume that CTMR is always superior to TMR for the time being. Similarly, the usage of NMR techniques or high levels of CTMR are possible though multiplying the

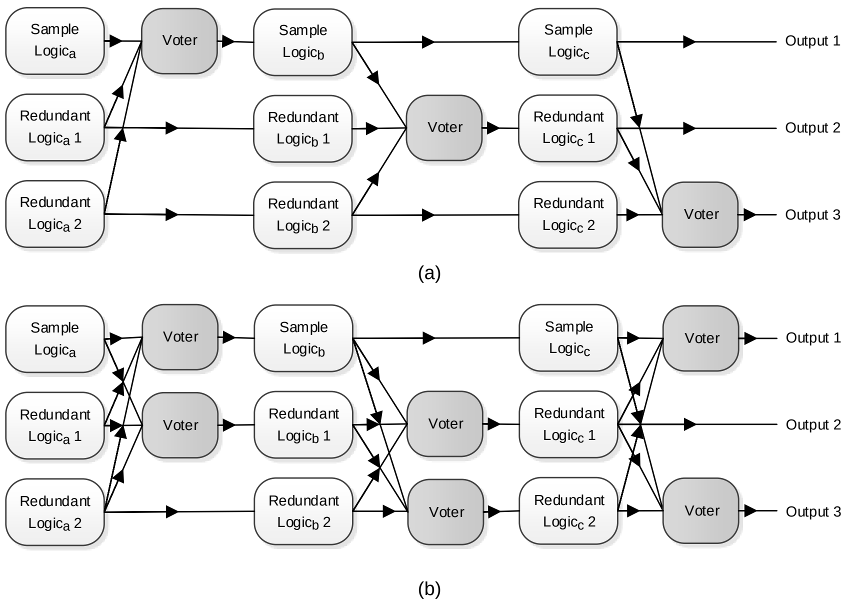


Figure 2.4: Triple modular redundancy with a) one alternate voter b) two alternate voters

cost on the performance factors of area, latency and power. Overall and based on the results presented in [24, 25, 26], we rate the reliability of the discussed configurations in the following ascending order:

- SV: Single Voter ([24])
- OAV: One alternate voter ([25])
- TAV: Two alternate voters ([25])
- TV: Triplicated voter ([24])
- CTMR: Cascaded TMR- Level 1 ([26])

In this thesis, we will use these five redundancy configurations for experimentation and comparison. However, there is no limit to the number of higher redundant configurations when one uses N-modular redundancy or higher cascaded levels of CTMR.

2.4 Field-programmable Gate Arrays in Space Computing

Field-programmable Gate Arrays (FPGAs) are integrated circuits, whose logic blocks and interconnects are configurable in contrast to the Application Specific Integrated Circuits (ASICs) whose functionality is fixed to a particular task. FPGAs consist of a very large number of configurable logic blocks (CLBs) containing a number of basic programmable

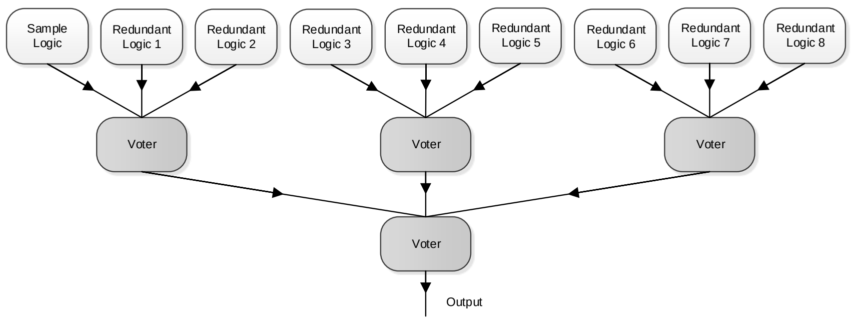


Figure 2.5: Cascaded triple modular redundancy- Level 1

components: flip-flops (FFs) and look-up tables (LUTs). These basic elements are connected by a programmable interconnect network, which allows to configure a routing network on a grid-layout and provides large flexibility in functional design modification on the FPGA. The hardware design on FPGAs is done by hardware description languages, i.e., Verilog or VHDL. Today's FPGA market is largely dominated by manufacturers Xilinx and Altera.
Due to the run-time *reconfigurable design* feature of the FPGA, it is increasingly in demand for spacecraft electronics. FPGAs can perform various tasks during different phases of a mission without the need for holding dedicated hardware for each task. This, in turn, reduces the carry-on hardware and weight of the satellite payload, while providing the feasibility to shut-down circuit modules which are not in use to prevent excessive power dissipation. Moreover, increase in on-board processing requirements on space missions for various image processing applications go well with the highly parallel architecture of FPGAs [2]. Most importantly, FPGAs allow spacecraft designers to upload new configuration data (or modify the hardware) after launch in case the mission requirements change or an error is found in application/task design [3]. As examples, SRAM FPGAs have been utilized in both earth orbits [27, 28, 29] as well as towards Mars [30].

2.4.1 Radiation-tolerance in FPGAs

Since the FPGAs are manufactured by the similar state-of-the-art fabrication technologies as used in all electronics and ASICs, they are equally susceptible to radiation-induced errors. Being in demand in space-computing, their radiation-tolerant design is important as their operating environment is composed of highly energetic radiation particles. In particular, these particles could appear as errors in different resources of an FPGA, e.g., SEUs in sequential logic (registers, memory blocks, DSPs) and configuration memory and SETs in combinatorial logic (LUTs) and clock and global routing [18]. The configuration memory is the most critical part of an FPGA which stores the bitstream representing the functionality of an FPGA including LUT contents as well as routing network, and therefore needs higher protection against radiation particles than other hardware blocks. It has been

validated in radiation-tolerance studies for FPGA, for example in Rosetta experiment [31], that the FPGAs are susceptible to radiation-induced errors, particularly at high altitudes. Therefore, the FPGAs manufacturers, provide radiation-hardened solutions for FPGAs by device as well as by design.
The commonly used FPGAs are SRAM-based. Although the high performance and reconfiguration flexibility are achievable only through SRAM based FPGAs, they are highly susceptible to SEUs. Therefore, the FPGA manufacturers offer space-grade FPGAs with different device technologies on the cost of lower performance compared to SRAM alternatives. They include the foremost Actel (currently known as Microsemi) RTAX Antifuse FPGAs [32]. These devices are one-time programmable and the development of permanent interconnections after configuration make them immune to SEUs. However, being non-reprogrammable, they lose their charm for utilization in multiple design modification scenarios. The second popular category consists of Flash-based FPGAs [33] which offer full reconfiguration though lacks in partial reconfiguration [2]. Moreover, these devices have typically lower TID rating than SRAM or Antifuse FPGAs [34]. Therefore, utilizing the performance of SRAM FPGAs while having built-in radiation-tolerance features leads to space-grade SRAM FPGAs, e.g., Xilinx Virtex 5QV [15], however, not all resources in these FPGAs are radiation hardened, e.g., BRAM and DSP modules [35].
In contrast to inherent radiation-tolerance capabilities of FPGAs, fault-tolerant computation approaches in hardware and software are utilized as well. TMR [9, 24] and scrubbing [36, 37] are two most popular techniques for tolerating SEUs and avoiding their accumulation in FPGA designs. TMR and its different voter implementations were discussed in Section 2.3.2. Scrubbing, on the other hand, involves refreshing the configuration memory contents by reading its golden copy at regular time intervals, to prevent accumulation of errors. Whether external or internal, scrubbing typically compares the calculated cyclic redundancy check (CRC) value of each of the configuration bitstream frames to the known CRC value of that frame [38] or rely on Hamming codes for single-error-correction-double-error-detection (SECDED) error mitigation [39]. Its worth noting that the configuration memory scrubbing do not account for errors in the user memory, i.e., BRAM, which can be made error-resilient by separate BRAM scrubbing, error-correcting codes (ECC) implementation or simply applying triple modular redundancy for each BRAM block. The combination of redundancy and scrubbing is considered a widespread optimal fault-tolerant solution in hardware. Additional approaches in literature include duplication with comparison (DWC) [40], error checking and correcting codes (ECAC) [41] and algorithm-based fault-tolerance (ABFT) [42], as examples. However, in this paper, we focus solely on redundancy and its different variations in hardware.
Hardware redundancy techniques for FPGAs are more involved than basic TMR with respect to partitioning a circuit into submodules, deciding on how many voters to insert, and where to place the voters in the FPGA design. Tools for automating redundancy-insertion in FPGA designs are available, including the TMR tool of Xilinx [43], Precision Hi-Rel software [44], and the BYU-LANL TMR tool [45]. Fault-tolerance mechanisms, particularly modular redundancy, comes with an overhead in terms of excessive area consumption as well as latency and power dissipation. Therefore, while providing fault-tolerance, the

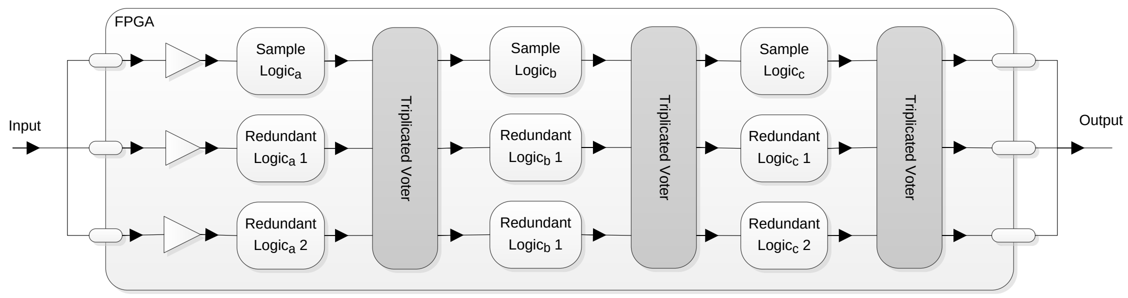


Figure 2.6: TMR implementation in FPGA

design of a mission critical system also has to limit these overheads to given constraints. Generally, space-grade SRAM FPGAs have been used with redundancy and scrubbing features to obtain a combination of radiation-hardening by device as well as by design [35, 46].

Triple Modular Redundancy in FPGAs

The TMR technique can be used in an FPGA by simply triplicating the inputs, outputs and logic modules, inserting buffers and connecting the outputs of logic modules to the triplicated voter [24]. There are some practical considerations due to which this straightforward implementation is not suitable. Firstly, TMR is able to counter one error among the three redundant branches, and a larger length of each branch increases its probability of being erroneous more than once. To deal with this issue, there is a need to break the logic of the branch at regular intervals and place the triplicated voters in intermediate stages of the circuit as shown in Figure 2.6. Thus, an error occurring in one partition of the logic will not be forwarded to the next partition due to the error-mitigation effect of the triplicated voter. However, the minimum size of the logic partition, or granularity level, could be limited to a single component on an FPGA, e.g., a look-up table or a multiplexer. In addition, there are certain locations on an FPGA called illegal-cut locations which should not be triplicated due to the FPGA architecture, e.g., dedicated route connections in a slice [47]. Moreover, voters should not be placed on high-speed carry chains in order to not deteriorate the timing performance of the design. Most importantly, voters should always be added in the feedback paths to avoid data corruption at the outputs of sequential elements being forwarded into the feedback paths [24, 47]. These voters are commonly denoted as synchronization voters. Figure 2.7 shows a fine-grained TMR implementation than Figure 2.6 including combinational (inverter) and sequential (D-type flip-flop) elements as well as depicting a triplicated synchronization voter. Part(a) of this figure represents a state-machine of a simple 1-bit counter which is triplicated, in part(b), and the voters are inserted before the feedback path to follow the synchronization voter concept.

The process of automatic TMR insertion into a circuit design can be done by the automated redundancy-insertion tools, as mentioned previously. However, in this research, we use

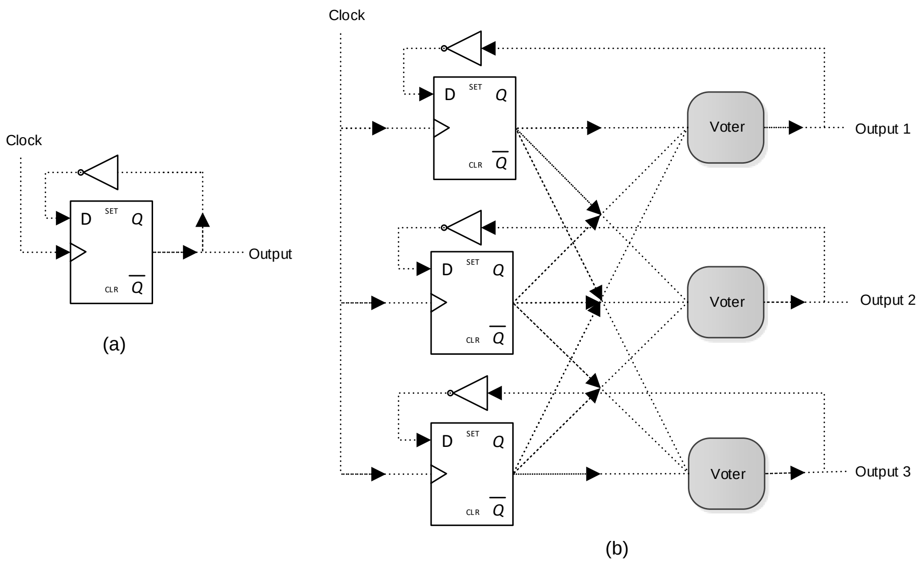


Figure 2.7: TMR implementation of one-bit counter [24]

BYU-LANL (BANL) tool for generation of redundant circuit netlists, as it is the only open-source and modifiable tool compared to other commercial ones. The BANL tool is able to triplicate the design, insert voters and use built-in algorithms to take care of the constraints explained above. It is up to the discretion of the circuit designer to request the desired *redundancy configuration*, e.g., TMR with only single voters or with a mixture of single and triplicated voters. Moreover, there is a choice of eight algorithms that decide the placement of voters in the triplicated design. These algorithms are termed *voter-insertion* algorithms. Depending on the type of an algorithm, the sets of nets are determined where the voters should be inserted, e.g., using feedback edge set (FES) algorithms or decomposition of strongly connected components (SCCs) in the circuit graph [47]. The details of the these algorithms can be found in [48]. The algorithms used in the BANL tool and used in our experimentation are abbreviated as follows:

- CC: Connectivity cutset
- AFC: After flipflop cutset
- BFC: Before flipflop cutset
- BD: Basic decomposition
- HFC: Highest fanout cutset
- HFFC: Highest flipflop fanout cutset
- HFFIC: Highest flipflop fanin input cutset

- HFFOC: Highest flipflop fanin output cutset

Originally, the BANL tool supported only SV and TV configurations while we extended the tool to support the rest of the three configurations, i.e., OAV, TAV and CTMR, as explained in Section 2.3.2. It has to be noted that all configurations have to resort to single voters for illegal-cut locations. This happens as the three redundant branches have to converge to one where the single voter provides the required convergence at the redundancy-forbidden locations. Therefore, the configuration TV which is the default configuration of the BANL tool as well as OAV, TAV and CTMR combines their respective configurations with single voter structures. However, we are not using sole SV configuration for experimentation due to its single point of failure limitation.

2.5 The Need for Adaptive Fault-tolerance in FPGAs

Following different redundancy structures in Section 2.3.2, we learned that by going towards higher levels of redundancy, the hardware usage increases. This in turn increases the area consumption, latency and power dissipation. Moreover, as briefly discussed in Section 2.1, the radiation strength of the space environment is fluctuating, and depends on the path of the space mission. Therefore, there is a need to develop an adaptive system that monitors the radiation strength and varies the redundancy levels accordingly at run-time thereby optimizing the trade-off between reliability and performance in a best possible way. An extended design approach to this idea is to vary the reliability/redundancy levels of individual hardware modules in a system according to their criticality or available resources on the FPGA. This concept is illustrated in Figure 2.8, where each of the n hardware modules have m possible redundancy configurations rated by different performance factors.

2.5.1 Correlating Apative Fault-tolerance with Varying-radiation Environments

In this section, based on the literature, we provide 4 distinct radiation scenarios to explain how the radiation strength varies along the path of a space mission, and hence discuss how

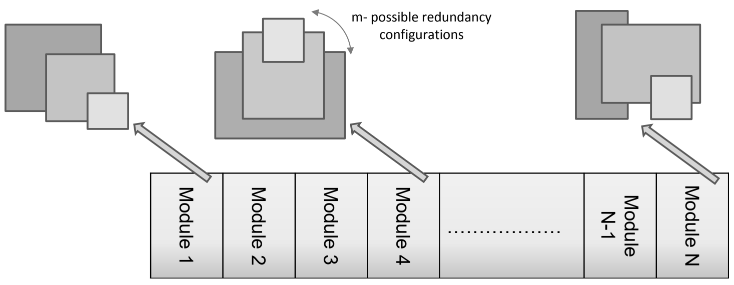


Figure 2.8: The Concept of Adaptive Redundancy

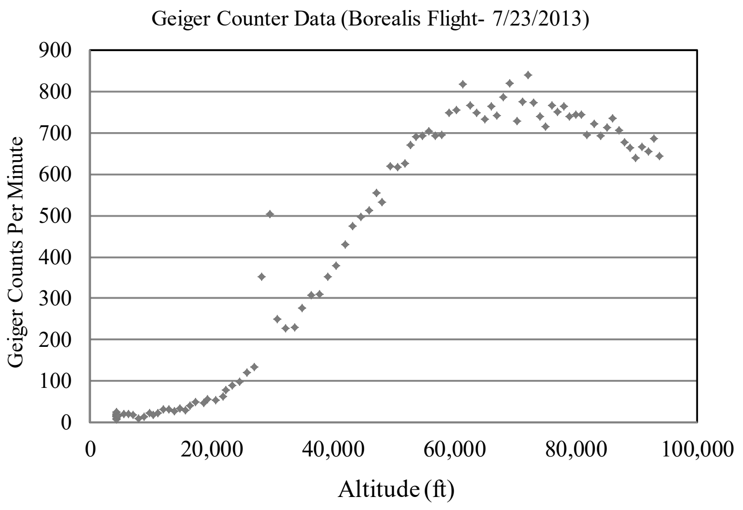


Figure 2.9: Borealis radiation strike-rate profile

varying reliability/redundancy levels fit into such radiation patterns. The first scenario is referenced with height, second and third with respect to time and fourth with solar conditions. The radiation strikes per minute and the soft error-rate are considered proportional measures of radiation strength, however, not all the radiation particles appear as errors in the hardware due to the their low ionizing energy or masking effects.

Borealis Flight

In a hot air balloon testing conducted at University of Montana [49], a custom radiation sensor was built and sent to a high altitude of around 100,000 feet with the time duration of 103 minutes (one-way). The experiment was aimed at observing the total number of particles hitting the sensor in a high energy flux environment, up to 10 MeV. The sensor logged the radiation strikes per minute, detected by a Geiger counter that captures most of the low and high energy particles. Figure 2.9 shows the variation of the recorded radiation strike rate with altitude. This figure reports on an extended experimentation of [49] called as Borealis flight[1]. It is evident from this experiment that the radiation strikes increase sharply with altitude. To implement adaptive fault-tolerance in this scenario, one can divide the radiation data into number of regions equal to the redundancy levels/configurations at hand while utilizing higher redundant structures for high radiation regions. In this case, the more redundancy levels we have at hand, the better a system would be able to exploit the trade-off between reliability and performance. It is also worth noting that the sensor captured two radiation data points at approximately 30,000 feet which are uncorrelated to the trend. This emphasizes the need for a system which is able to adapt the reliability level in order to respond to unexpected deviations from an observed trend.

[1]The data has been obtained via private communication with the authors in [49].

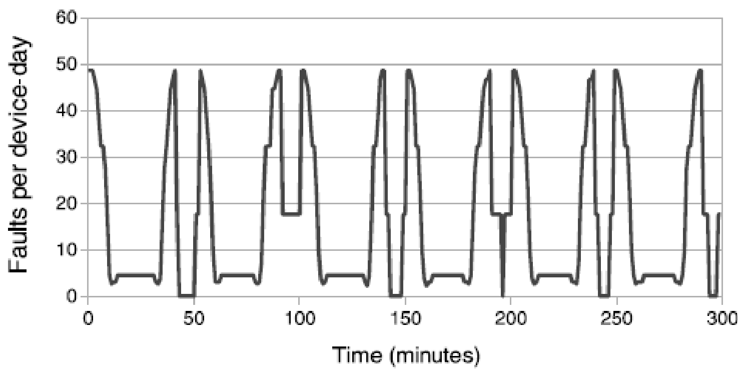


Figure 2.10: Expected fault-rate of LEO [2]

Low-Earth-Orbit Case Study

In the research work presented in [2], a fault-rate model is presented and used to simulate the expected error-rate for a path in low-earth orbit (LEO) with the two-line element (TLE) of EO-1 satellite. This orbit is typically used for Earth-observing science applications. The error-rate is presented in the unit of faults per device day which is useful to represent the cumulative error-rate when a number of devices/FPGAs are monitored in parallel. The orbital track of LEO case study has a mean travel time of 98 minutes due to which we see the repetition of fault pattern, in Figure 2.10, with maximum fault-rates estimated at the Earth's magnetic poles. The fault rate in this orbit is not excessive as the orbit has an altitude of 700 km which is below the Van Allen radiation belts and completely within the Earth's magnetosphere. It can be observed from the figure that there are approximately three discrete radiation/fault-rate levels, hence three reconfigurable reliability levels would be sufficient.

Highly-Elliptical Orbit Case Study

Referring to the research work in [2], an additional case study for expected fault-rate was conducted for an elliptical orbit of perigee 1100 km and apogee 39000 km with a mean travel time of 12 hours. The path is called Molniya orbit and used for the communication satellites in particular. Compared to the LEO case study, the fault-rate is very high especially when the satellite passes the Van Allen radiation belts. As can be seen in the radiation plot of Figure 2.11, the fault rate gets high at the end of the time period when it passes the Van Allen radiation belts. Most of the orbit duration has negligible fault-rate as compared to a short duration of excessive fault-rate at the end of the time period. For such a fault-rate profile, we can expect two reliability levels to be sufficient where the higher reliability level is required only for a very short duration.

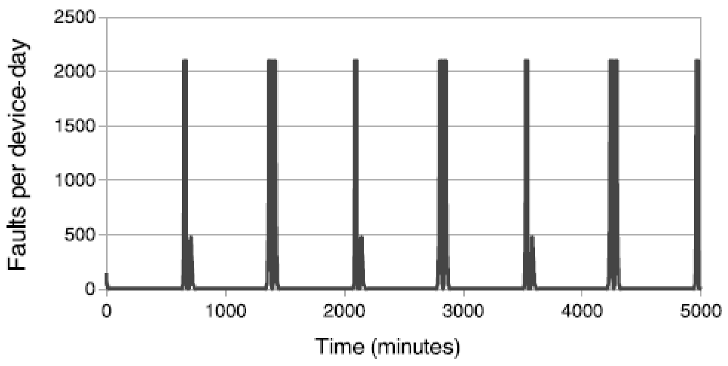


Figure 2.11: Expected fault-rate of HEO [2]

Anticipated Error-Rate for Different Solar Conditions

In the research work conducted in [28], an expected error-rate for another LEO was anticipated for seven different solar conditions as shown in Figure 2.12. While the names of the solar conditions are replaced with numeric numbers in the figure for simplicity; their details can be found in [28]. The graph shows the minimum fault-rate of 0.5 SEUs per device day in solar condition No. 2 and maximum fault-rate of 26 SEUs per device day for solar condition No. 3. The graph can be divided into two regions according to SEU-rate. The first region (solar conditions No. 1, 2, 5, 6 and 7) can be used with low redundancy while second region (solar conditions No. 3 and 4) can be used with high redundancy structures.

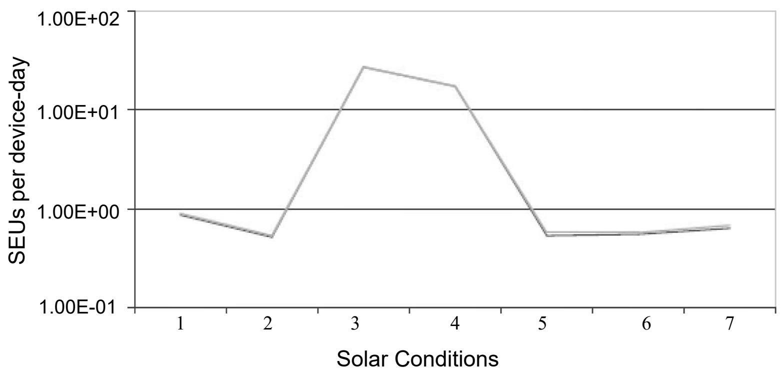


Figure 2.12: Expected fault-rate of LEO orbit under different solar conditions [28]

2.6 Reliability Evaluation of FPGA Designs for Space Computing

The reliability of the FPGA based hardware designs for space applications can be computed by developing methods following the knowledge of space environment and device vulnerability. The different resources of an FPGA, e.g., configuration memory, BRAM, DSP components, etc should be modeled for reliability separately since each of theses resources respond differently to radiation. Additionally, the redundant structures can be evaluated for reliability-enhancement by developing fault-rate models. The traditional way of modeling reliability of electronic systems is to describe the time until the system fails by a random variable. Using an exponential distribution for the time between failures, assuming independent failures, and a constant fault-rate λ we can determine a system's time-dependent reliability $R(t)$ as defined by Equation 2.3.

$$R(t) = e^{-\lambda t} \tag{2.3}$$

Assuming a constant fault-rate is reasonable if we exclude the burn-in and wear-out phases of systems. Thus, based on an estimated or measured fault-rate, $R(t)$ can be calculated and expresses the probability that a system survives, i.e., is without fault, from its start at time 0 until time t. The reciprocal of λ is denoted as mean-time to failure (MTTF) or mean-time between failures (MTBF) in case of systems that can be repaired. MTTF and MTBF are widely used as reliability metrics. Though many research works follow this generic reliability equation, they differ in how they determine the fault-rate λ. In literature, two methods of reliability/fault-rate computation for FPGAs are used in addition to error-injection based simulations and radiation testing methods.

2.6.1 CREME96 based Reliability Computation

Vaderbilt University's online tool CREME96 [50] is specifically used to compute fault-rate of devices used in aerospace missions. The tool takes as input parameters defining a space orbit, weather conditions and fault cross-section of a device and hence computes the fault-rate/SEU-rate of the device in units faults per device-day [12].

Device Characterization

The initial step is to characterize the device that is to be used in space orbit. For this purpose, the device's upset rate is calculated via experimental testing in particle accelerator. Afterwards, the FPGA's static cross-section is determined by calculating the upsets caused in the device for a given fluence of radiation, as shown in Equation 2.4.

$$\sigma = \frac{Number\ of\ errors}{fluence\ (particles/cm^2)} \tag{2.4}$$

The static cross-section refers to the FPGA's vulnerable area to radiation particles at a specific energy. Therefore, the experimental testing is repeated for a range of energy levels including both proton and heavy ions to compute the overall static SEU cross-section.

Path/Orbit Specification

CREME96 uses data from previous satellite orbits to predict an average flux of particles in certain orbits under different solar conditions. Therefore, while choosing orbit conditions and parameters, the specific solar condition has to be selected as well. *Solar Minimum* and *Solar Maximum* refers to the lowest and highest averaged flux during the solar activity of 11 year solar cycle, respectively. The *worst week*, *worst day* and *worst 5 minute peak* solar conditions refers to the specific solar event of October 1989, and used for worst case estimates. Additionally, *peak trapped proton* solar condition refers to the worst proton flux for all proton energies for both solar minimum and maximum portions of solar cycle.

SEU Rate Prediction

Therefore, utilizing the device SEU cross-section, choosing a solar condition and by providing specific parameters of space orbit, the SEU rate due to heavy ions and protons are computed separately and hence averaged. The basic fault-rate model of this tool whether used in its original form or modified for specific system designs can be found in research works of [2, 3, 46]. However, the impact of redundancy can be modeled by Markov fault model [2] or classical TMR equation [46].

2.6.2 Probabilistic Computational Reliability Models

Another category of reliability computation is via probabilistic computational models that take as input error probabilities of individual components and compute the output error probability ε_{out} of the overall system by propagating error probabilities from inputs to outputs. A number of publications relate the output error probability to the system's reliability according to $R = (1 - \varepsilon_{out})$, the most popular works being [51, 52, 53, 54]. It is important to note that this notion of reliability and the underlying notion of a system's output error probability are instantaneous and hence timeless. The related work, however, does not detail how to determine the exact component error probabilities and uses arbitrarily set values. Hence, the probabilistic computational models are in contrast to the time-dependent reliability measure of Equation 2.3, and the fault-rate method in CREME96 that simulates real device and orbit features and targets mainly radiation induced errors. A brief overview of the major probabilistic computational models along with their complexity is provided as follows.

An accurate and powerful model in error probability calculation is the probabilistic transfer matrices model (PTM) [51], though it is an extremely complex and time-intensive method when used for large circuits. For large circuits, the number and size of matrices become huge which needs a lot of storage memory and computation time for calculating regular and tensor matrix products. The major concern is the modelling of a wire swap as a single PTM stage. In FPGAs where wire swaps are based on the placed and routed design by FPGA design softwares, e.g., Xilinx ISE, the reliability differs from one routed design to another. Moreover, the numerous wire swaps in large circuits drastically increase the number of matrix calculations, which is why this approach is unfeasible for modelling large circuits. Algebraic decision diagrams (ADD) have been used to improve the storage and timing performance of PTM method though it is still a non-promising solution for large circuits. An error-analysis for sequential circuits based on PTM is proposed in [55] though the application is considered for a simple adder only.

A major research work in error-modelling has been proposed with the probabilistic gate model (PGM) [52] and the Bayesian networks error modelling [53]. The basic algorithm of PGM is easy to model though the accurate model has a high complexity particularly to evaluate reliability of large circuits. Therefore, a midway solution called modular approach is proposed for large circuits though no details of a toolbox implementation or assumptions considered are presented. Moreover, sequential circuits are not supported. Similarly, Bayesian networks are used efficiently for analyzing small circuits; however approximate techniques are proposed for large circuits due to complexity of the model. Additionally, the results of Bayesian model are highly incomparable to PTM and other popular error probabilistic schemes [56].

Probabilistic decision diagrams (PDD) [57] provide a model scalable for analysis of large circuits by decomposing the circuit based node graph and thus achieving timing efficiency. However, its worst-case complexity is still exponential in the number of inputs of the circuit. Probabilistic model checking [58] is another approach measuring circuit reliability but on the cost of excessive memory requirement and timing complexity. The memory requirement problem was addressed successfully in [26] though high runtimes are still a problem. A hybrid approach combining exact and probabilistic models to evaluate reliability is proposed in [59]. This scheme lowers the complexity of the analysis, however the timing results are not reported. The analysis is performed on small circuits excluding sequential elements though it is a good approach to harden error-sensitive gates in the circuit by upsizing them. Observability-based and single-pass reliability analysis [60] claims to be an extremely fast and scalable approach particularly compared to PTM and Bayesian techniques though the work lacks detailed results and comparisons. A strong focus is given on using observability analysis to improve reliability by manipulating reconvergent fanout while avoiding redundancy-insertion.

A simple and fast approach known as Boolean difference error calculator (BDEC) is proposed in [54, 61]. This technique is independent of the wire swapping, involves simple Boolean calculus and is perfectly analytic which makes it accurate for calculating reliabilities for circuits of any size. Moreover, its complexity is linear in the number of inputs of the circuit. Since the model is applicable to any logic element represented in a form of Boolean equation, it is perfect to use with FPGA based netlists. The authenticity of this approach has been proved by the comparison of results of this scheme with Monte Carlo simulations performed in [54]. A close insight into BDEC can be found in the next chapter where we utilized and modified this technique to calculate reliability values of non-redundant and redundant versions of utilized benchmarks.

The drawback of using these probabilistic methods is that they do not take into account the impact of redundancy. While redundant configurations of a circuit are known to be more reliable than non-redundant version, these methods provide us even lesser reliability of redundant circuits since they take each component as erroneous and bigger size of a redundant circuit automatically means lower output reliability. The premier solution for this interpretation problem will be provided in Section 3.1.2 where the probabilistic theory is merged with the conventional TMR theory [22, 23] to distinguish the reliability-enhancement component in the circuit, i.e., voter from the rest of error-prone components [6].

2.6.3 Fault-injection and Testing-based Reliability Models

A typical way of experimentally checking FPGA reliability is via fault injection into the FPGA bitstream [62]. Since the SEUs can be modeled as bit flips in storage elements, we can revert the bit at the register output and see its effect on the behavior or functionality of the circuit. The bitstream formats of FPGAs are proprietary, which turns the identification of the exact bit locations in the bitstream that need to be flipped to execute a specific fault in the desired circuit element, into a tedious re-engineering job. Therefore, random fault injection is normally exercised using Monte Carlo techniques [25]. Since an FPGA bitstream contains only a small fraction of critical bits, i.e., the bits that cause an error when affected by an SEU, most of the SEUs do not affect the circuit's functionality. Together with the restricted time for simulation this leads to a limited coverage.
The FPGA manufacturers, e.g., Xilinx and Altera, have their custom testing methods and models that provide reliability measures. They typically report reliability in the FIT (failures-in-time) metric, which defines the number of errors in one billion operation hours of a device [63]. The failure rate can be determined for Xilinx devices by Equation 2.5.

$$Failure\ Rate = \frac{x^2 10^9}{2(D)(H)(A)} \tag{2.5}$$

where D, H and A refer to number of devices, number of hours and acceleration factor respectively. x^2 refers to the Chi-squared value at desired confidence level and $(2f+2)$ degrees of freedom, where f is the number of failures. The acceleration factor is calculated using Equation 2.6.

$$Acceleration\ Factor = \exp\left(\frac{E_a}{k}.\left(\frac{1}{T_{J1}} - \frac{1}{T_{J2}}\right)\right) \tag{2.6}$$

where E_a is the thermal activation energy, k is the Boltzman's constant and T_{J1} and T_{J2} are the use junction and stress junction temperatures respectively. The Xilinx results show the calculated failure rates for various process technology nodes [63, 64, 65]. Additionally, the radiation affects are modeled by the SEU device rates measured by accelerated beam testing and real-time atmospheric testing, particularly for configuration memory and BRAM. Moreover, various other failure-rate measurements are conducted by Xilinx and reported in their device reliability reports, though not only radiation induced errors but also high temperature, humidity and stress tests.

2.7 Major Research Works in Adaptive Fault-tolerance

In this section, we briefly discuss related projects, presented in literature, that follow the concept of adaptive fault-tolerance. All have in common a concept for a self-adaptive reconfiguration mechanism for reliability using FPGAs, with focus on radiation-induced errors. Though the adaptive reconfiguration can be used for avoiding hot-spots on FPGA during multicore computation too [66], it is out of scope for this thesis. Additionally, research works [67, 68, 69, 70, 71, 72] focus on the SEE mitigation techniques for FPGAs too but are not discussed in this section since they do not follow the adaptive reconfiguration narrative.

2.7.1 Reconfiguration for Reliability (R4R)

The authors in [73] present a very detailed approach for an adaptive system denoted as reconfiguration for reliability (R4R). The R4R framework proposes an initial circuit analysis to estimate the cost of different versions of a circuit implementation including the default design and more reliable, i.e., hardened, solutions. In the next step, design space exploration is performed including floorplanning to optimize the placement of hardened solutions on the FPGA. The hardened solutions are based on TMR at the system and component levels including the design obtained by the commercial Xilinx TMR tool. The solutions are then Pareto-optimized on the basis of area utilization and reconfiguration time. It is also stressed that different portions of the circuit design can be treated for varying levels of fault-tolerance depending on their criticality.
While this research work describes the initial analysis and design space exploration steps in detail, it stays abstract due to following reasons. R4R is seen as a broad framework for having an intelligent system, exploring different redundant hardware designs, though no implementation of such a system is proposed with run-time support. Moreover, the cost parameters do not consider other important factors, e.g., maximum clock frequency of the circuit and power dissipation of different hardened designs. Additionally, the authors do not discuss concepts for decision mechanisms for the online reconfiguration. Overall, this paper provides the basic knowledge and motivation for adaptive fault-tolerance with short case studies and can be taken as a foundation for further research in this field.

2.7.2 Reconfigurable Fault-tolerance (RFT)

The concept of reconfigurable fault tolerance (RFT) is provided in [2]. This work targets space applications and simulates processing components' availability based on DWC, TMR, ABFT and high-performance (HP) fault-tolerance mechanisms. The switching among different fault-tolerance approaches is supported by architecture-level changes through varying partially-reconfigurable fault-tolerance regions. The RFT hardware architecture, as a system-on-chip design with RFT controller is shown in Figure 2.13. The number of partially reconfigurable regions (PRR) utilized depends on the selected fault-tolerance approach, i.e., 3 PRRs for TMR, 2 PRRs for DWC and single PRR for HP, ABFT or internal TMR. The main architectural components consist of a microprocessor (Microblaze),

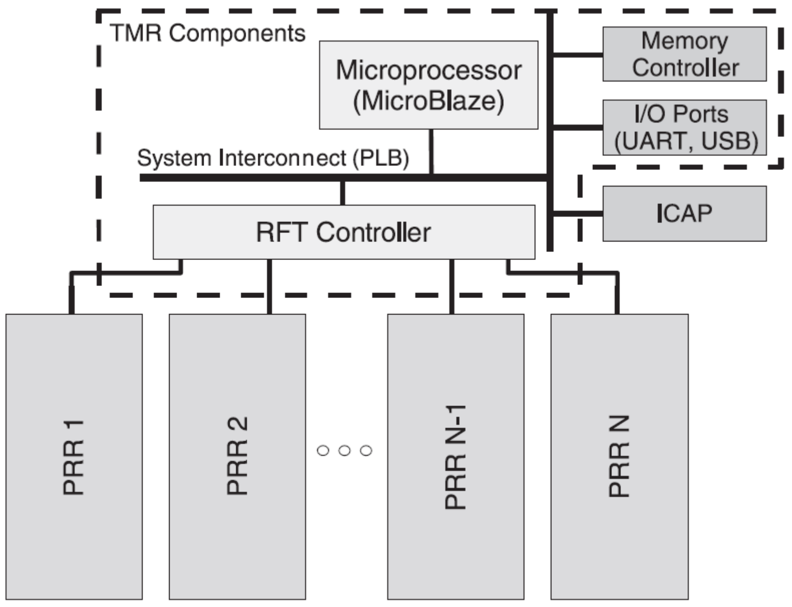


Figure 2.13: System-on-chip design with RFT controller [2]

processor local bus (PLB), memory controller, I/O ports and RFT controller (responsible for routing signals from PLB to PRRs). All architectural components are protected by TMR since their reliability dictates the system reliability.
Using this SoC approach, the authors use two case studies about low and high earth orbits to represent the radiation strength variation in atmosphere, which is useful for taking appropriate reconfiguration decisions. The fault-rate models in this research utilized the existing orbital fault models characterizing the space orbits using CREME96 tool. It has been successfully concluded from the experiments that different fault-tolerance approaches can be useful in different radiation environments, as shown in varying radiation example plots in Figures 2.10 and 2.11. Moreover, it has been clarified how high scrubbing rates can improve system performance. However, the authors neither compare the overheads due to area, latency and power for different fault-tolerance approaches, nor do they consider a design space exploration of different redundant designs. Additionally, RFT focuses on the classical, system-level TMR approach while more efficient component-based redundancy approaches are available.

2.7.3 Reconfiguration via Spare Resources Method

A custom radiation sensor measuring radiation strikes has been built and proposed for adaptive fault-tolerance in [3]. This research employs a radiation sensor whose measured radiation strikes are considered proportional to the errors appearing in the system. The basic principle of detecting a particle-hit is to detect the short surge of current that

is generated by the collection of induced holes and electrons, by the incident radiation particle, on two sides of the sensor. This system is also able to figure out the particular error-hit area and uses partial reconfiguration to mitigate this error. The specialized GUI used for this system consequently shows the particle-hit area and whether it has been affected by an SEU or SEFI.
In addition to TMR and scrubbing, this system even utilizes spare redundant blocks in the event of error-detection. The idea is inspired by the fact that handing over computation to a spare computation block is less time-intensive than reconfiguring the partially reconfigurable area to higher redundancy or scrubbing it. When all the spare computation blocks are exhausted, the computation starts from the initial block while reconfiguring all the blocks. The authors verify their approach and show that the combination of redundancy plus scrubbing plus spare resources technique improves the MTBF. However, this technique comes with the increased cost of holding extra resources for a single computation task. In satellite missions where cost is dominated by weight, latency and power of resources, such a system is indeed fault-tolerant but probably not cost-effective.

2.7.4 BRAM Fault-Detection based Adaptive Reconfiguration

The authors in [46] utilize the integrated block RAMs on the FPGA to monitor the error-rate, which is considered proportional to the radiation strength to which an FPGA is exposed at run-time. Figure 2.14 shows the FPGA based system consisting of BRAM sensor and adaptive subsystems. By checking the parity bits of the BRAM scrubber, the number of BRAM upsets are recorded in the fault memory and accessed by the fault management unit (FMU) to direct the reconfiguration control unit (RCU). The RCU then controls the adaptive subsystem by loading the partial bitstreams via ICAP from the external memory to implement one of three operation modes, i.e., no redundancy, DMR (dual modular redundancy) and TMR. The no-redundancy mode utilizes all the three partially reconfigurable regions to run the system at the maximum throughput.
The integrated BRAM Fault Detectors (BFD), in this system, implement radiation sensors at virtually no cost because the memory is still available for applications. However, the paper does not consider that the BRAM radiation sensor might not only detect SEEs due to radiation, but also faults introduced by other effects like supply voltage instabilities and aging/permanent faults. Additionally, the error counter registers are not protected. If an SEU hits an error counter, this might lead to a wrong error count. The reliability of the voter is not considered as well.

2.8 Chapter Conclusion

This chapter provides a background on the fault-tolerant computing for FPGAs used in high radiation space environments. As a starting point, we have described the radiation particles in space environment that appear as errors in FPGA based hardware designs. These errors, range from temporary effects, e.g., single event upsets and transients to permanent ones, e.g., total ionizing doze. Afterwards, we have explained two approaches

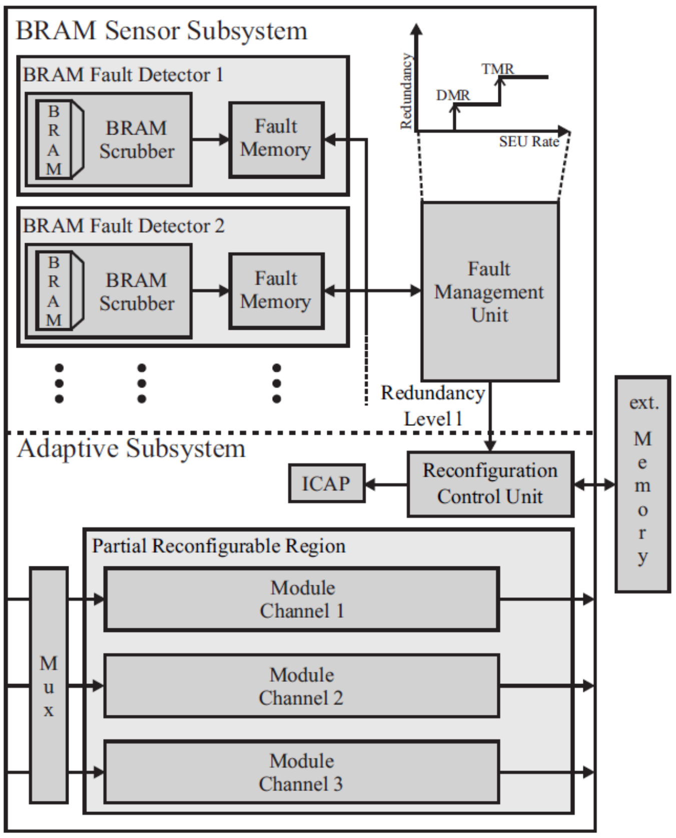


Figure 2.14: An FPGA-based SEU mitigation system with BRAM sensor and adaptive subsystems [46]

to harden the FPGA devices to these errors, i.e., by using device-based or design-based fault-tolerance. The device approach is expensive in terms of developing a separate semi-conductor fabrication process. On the other hand, design-based hardening uses the same device fabrication processes though utilizing the circuit design hardening techniques, the most popular being redundancy. By referring to different redundancy configurations, voter-insertion algorithms and replication factors, we have shown how redundancy can be implemented in FPGA devices resulting in a number of possible structures. Additionally, we have pointed out that the trade-off of the increased reliability comes in the form of degraded circuit performance factors, i.e., area consumption, latency and power dissipation. Therefore, we have highlighted the need for realizing and maintaining a suitable trade-off between reliability and performance that fulfills the system design constraints.
By presenting various radiation pattern case studies, we have concluded that the strength of radiation greatly varies during the time span of a space mission or satellite. Therefore, there is a need to optimize the reliability-performance trade-off at run-time on the basis of predicted/calculated radiation/error rate data. We have summarized this mecha-

nism as adaptive fault-tolerance in this chapter. Afterwards, we have provided the three broad categories of fault-tolerance in FPGA hardware, from the literature, i.e., CREME96 tool based reliability computation, probabilistic models and error-injection/testing based mechanisms. Moreover, we have discussed four research works in literature that focus the broad concept of adaptive fault-tolerance and show how the implementation and scope of each of these techniques differs from each other.

CHAPTER 3

Reliability Computation of Redundant Structures

This chapter focuses on computing and contrasting reliability of different redundant implementations of a hardware design. In Section 2.6, we have seen that three common reliability computation methods exist which, whether used in original form or with modification of their default fault-models, provide us the magnitude of circuit's reliability. However, before using each of these methods, we need to analyze them for their particular application domain. Prior to this analysis, we summarize the applicability of each of the three reliability computation methods as follows.

- CREME96 tool is the standard software that computes the circuit's reliability for aerospace applications with reference to a particular orbit and weather conditions. The fault model for computing reliability of redundant structures, using CREME96, was proposed in [2, 46]. The reliability, as measured by this software is typically in the form of single-event-upset (SEU) rate.

- The probabilistic computation schemes work on the static input error probabilities that must be provided to the simulation setup if one wants to utilize this approach. Therefore, these schemes can be utilized for comparing different circuit structures without having the exact knowledge of real-time component error probabilities, i.e., by using arbitrary or closely estimated values. Since the output reliability of a circuit depend on the flow of error-probability from primary inputs towards outputs, it can comprehend the exact difference in reliability among different redundant structures as every component in the circuit will have an impact on the overall computed reliability. This technique is particularly useful for comparing redundant configurations of a circuit which vary in small number of components, e.g., single, triplicated or alternate voters, as shown in Section 2.3.2. However, to make these models work with redundant structures, they need to be extended so that the redundancy effect can be comprehended differently than the error-prone effect of individual logic modules of the circuit.

- The third reliability computation method is fault-injection and hardware testing. The fault-injection whether done by injecting errors into bitstream or during hardware design, is extremely time-consuming since the number of fault combinations are huge and simulating each of the error scenarios is usually impractical. The hardware testing, on the other hand, works by implementing various redundant configurations on FPGA and check the accuracy of outputs under a radiation environment, e.g., in a particle accelerator. The testing technique is, in particular, hard to use for comparing redundant structures which vary in slightly different structures, e.g., alternate voter configurations. The reliability of this method could be measured as an SEU rate or as a percentage of time for an error-free computation.

The choice of a specific reliability model depends on the need of the circuit/system designer. For a practical aerospace application, the CREME96 software should be employed to gain realistic reliability values. Testing and simulation-based methods are mostly useful to provide a level of confidence in commercial products. The probabilistic techniques, whether used with actual or arbitrary input error probability values, excel at comparing different redundant circuit configurations. We have decided to base our work on probabilistic computational reliability schemes due to the following reasons:

- The mathematical models of these schemes are mature and have been thoroughly analyzed in literature [6].
- The probabilistic schemes are able to comprehend very small differences in reliability among slightly different redundant structures.

Among the probabilistic computation schemes, we decided to use the Boolean difference error calculator (BDEC) method based on the analysis and merits/de-merits of various computation schemes explained in Section 2.6.2. The basic BDEC model was not ready to use with redundant circuits' reliability computation due to certain limitations and non-availability of an automated tool based on this method. In the following section, we will describe in detail, the theory of BDEC model, its limitations and our extensions to this model.

3.1 Boolean Difference Error Calculator

The Boolean difference error propagation model when applied to a logic element, represented in the form of Boolean equation, takes signal and error probabilities of inputs along with the error probability of the logic element, i.e., n-input logic gate, and computes the error probability of the output. This concept is illustrated in Figure 3.1 by a BDEC model of a faulty logic element with Boolean equation f. The signal probability, i.e., p_n, refers to the probability of a signal being at logic 1. ε_n and ε_g refers to input and gate error probabilities respectively. The details of this model can be found in [54, 61], however to highlight the fundamental concept, we provide the mathematical implementation of this model by using a 2-input AND gate as an example. The generic equation for the output

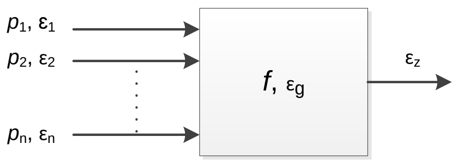


Figure 3.1: BDEC model of faulty logic element [54]

error probability, i.e., ε_z, of an n-input logic gate is represented in Equation 3.1:

$$\varepsilon_z = \varepsilon_g + (1 - 2\varepsilon_g)\varepsilon_{in} \tag{3.1}$$

For a 2-input logic gate, the total input error ε_{in} is computed via Equation 3.2:

$$\varepsilon_{in} = \varepsilon_1(1 - \varepsilon_2)Pr\{\frac{\partial f}{\partial x_1}\} + \varepsilon_2(1 - \varepsilon_1)Pr\{\frac{\partial f}{\partial x_2}\} + \varepsilon_1\varepsilon_2 Pr\{\frac{\Delta f}{\Delta(x_1x_2)}\} \tag{3.2}$$

where x_i represents the corresponding input of the logic gate. The first and the second terms of the equation model the error in the first and the second inputs respectively, while the final term represents the simultaneous error in the two inputs. $\frac{\partial f}{\partial x_i}$ and $\frac{\Delta f}{\Delta(x_1x_2)}$ represent partial and full Boolean derivatives of the logic function respectively. The signal probability function $Pr\{.\}$ returns the probability of its Boolean argument to be '1'. As an example, the computation of signal probability functions containing partial and full derivatives for a 2-input AND gate ($f = x_1x_2$) is shown in Equations 3.3, 3.4 and 3.5 though the details on computation of partial and full derivatives is found in [54].

$$Pr\{\frac{\partial f}{\partial x_1}\} = Pr\{x_2\} \quad = p_2 \tag{3.3}$$

$$Pr\{\frac{\partial f}{\partial x_2}\} = Pr\{x_1\} \quad = p_1 \tag{3.4}$$

$$\begin{aligned} Pr\{\frac{\Delta f}{\Delta(x_1x_2)}\} &= Pr\{\overline{x_1}\ \overline{x_2} + x_1x_2\} \\ &= (1 - p_1)(1 - p_2) + p_1p_2 \\ &= 1 - (p_1 + p_2) + 2p_1p_2 \end{aligned} \tag{3.5}$$

By combining Equations 3.3, 3.4 and 3.5 with 3.1 and 3.2, we obtain the total input error and output error probability of 2-input AND gate in Equations 3.6 and 3.7 respectively.

$$\varepsilon_{in} = \varepsilon_1(1 - \varepsilon_2)p_2 + \varepsilon_2(1 - \varepsilon_1)p_1 + \varepsilon_1\varepsilon_2(1 - (p_1 + p_2) + 2p_1p_2) \tag{3.6}$$

$$\varepsilon_{AND2} = \varepsilon_g + (1 - 2\varepsilon_g)(\varepsilon_1p_2 + \varepsilon_2p_1 + \varepsilon_1\varepsilon_2(1 - 2(p_1 + p_2) + 2p_1p_2)) \tag{3.7}$$

Therefore, starting from a Boolean equation and while using Boolean difference calculus, we obtain an algebraic equation of output error probability of an n-input logic gate. Likewise, we can calculate the generic equations of all n-input gates used and utilize them for different logic functions; which accounts for the simplicity of BDEC implementation. Since not all the errors which appear in a circuit propagate to the final outputs; this masking effect of errors is identified in the model by the p_n values used in conjunction with the Boolean equation of the logic element.

3.1.1 Limitations of the BDEC Model

The BDEC model has the following limitations due to which its default version cannot be used to compute reliability of redundant circuits.

BDEC v/s TMR Model of Voter

The BDEC model treats every circuit component as erroneous and accumulates the error effect of each component in the output reliability as the error probability flows from inputs to outputs. Redundancy adds extra components to the default circuit where these extra components will be treated as erroneous by the BDEC theory as well. Hence, a redundant circuit, when evaluated for reliability by BDEC provides even lower output reliability as the non-redundant one. It can be noted that BDEC model is timeless and accounts for the reliability computation at an instantaneous time value.
On the other hand, the reliability of redundant systems is traditionally evaluated by classic TMR model [22, 23]. Equation 3.8 describes the reliability of a TMR system with a single voter module, based on the binomial theorem. Using this equation we compute the output reliability R_{out} of a voter as a function of the modules' reliabilities R_1, R_2 and R_3 and the voter reliability R_{voter}. Contrasting the input parameters with BDEC, TMR model does not take p_n values into account. Moreoever, the TMR model does not discuss or support the notion of error-probability (ε). The reliabilities in this model are time-dependent, however, we present here an instantaneous time snapshot of this model to align with BDEC model. Figure 3.2 shows the difference between the the BDEC and TMR models of the voter element.

$$R_{out} = R_{voter}[R_1R_2 + R_1R_3 + R_2R_3 - 2R_1R_2R_3] \tag{3.8}$$

The reliability evaluation by these two theories differ for the voter component only which serve as the basis for redundant systems. The TMR model, which holds a non-linear relationship of output with input reliabilities, has the capability of improving output reliability provided that the input and voter reliabilities stay lower than certain bounds, which will be proved in Subsection 3.3.4. The BDEC model, in contrast to TMR model, treats input reliabilities as mutually independent, and does not have the capability to interpret the redundant behaviour. Additionally, redundant systems have not been evaluated by BDEC or any other probabilistic reliability evaluation scheme in the literature. The difference between BDEC and TMR reliabilities for five input reliability combinations are shown in

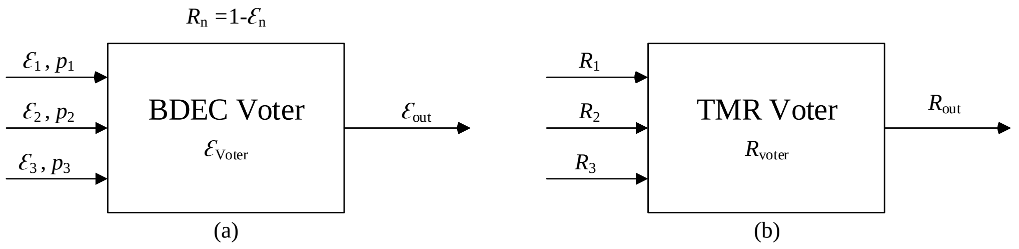


Figure 3.2: BDEC v/s TMR model of voter

Rvoter	R1	R2	R3	Rout- BDEC	Rout- TMR
99%	95%	95%	95%	92.0%	98.3%
99%	99%	98%	97%	96.1%	98.9%
99%	99%	90%	70%	80.5%	95.7%
99%	95%	90%	85%	85.6%	96.4%
95%	95%	95%	95%	88.6%	94.3%

Table 3.1: BDEC v/s TMR Reliability of Voter

Table 3.1. It can be seen that BDEC model, due to its non-interpretation of redundant behaviour always result in lower reliability than the TMR model. However, the TMR model is highly dependent on the voter reliability and the output reliability falls rapidly with the decreasing input reliabilities.

Reliability Computation of Sequential Circuits

The BDEC theory as well as other probabilistic computation techniques are limited to analysis of combinational circuits only. It has been claimed in the related research works that these techniques, including BDEC, can be used for sequential components by computing error probability for multiple iterations. However, to the best of our knowledge, the literature does not provide any results.

Reliability Evaluation Tool

The literature on BDEC model provides the analysis of only combinational circuits which are small and limited in size. Without the automation of the mathematical model, the BDEC model cannot be used for the analysis of large circuits which comprise of both combinational and sequential elements. Therefore, an automation of the BDEC model is required in the form of a tool to make the analysis of large circuits possible.

3.1.2 Extensions to the BDEC Model

To address the limitations in BDEC model, as explained in the previous section, we extended its conventional model in two ways followed by the automation of the revised BDEC model. Firstly, our revised model is able to treat redundant portions of the circuit according to classical TMR theory instead of faulty BDEC elements. Secondly, the model supports sequential circuit analysis by iteratively calculating reliability of sequential elements in feedback. Finally, the revised model is automated so the circuits of any size can be evaluated for reliability by only providing the circuit netlist as input to the automated BDEC tool.

Interpretation of Redundant Behaviour

The default BDEC model, as shown in the previous section, is not capable of evaluating reliability of redundant circuits. In other words, the reliability of voter elements, calculated by the BDEC model, are not correct. Therefore, we need to use TMR model to calculate reliability of voters. Hence a merger of BDEC and TMR model is exercised to interpret the redudant behaviour of circits. Based on the naming conventions of components in the circuit netlist we can easily distinguish the voters from other components and evaluate them for reliability via the classical TMR theory; all other components are evaluated by the BDEC model. Figure 3.3 illustrates this concept on a TMR subsystem. The three redundant circuit modules are evaluated in the BDEC domain, which results in reliabilities $R_1 = 1 - \varepsilon_1, R_2 = 1 - \varepsilon_2$, and $R_3 = 1 - \varepsilon_3$. The voter elements, on the other hand, are evaluated by the TMR equation 3.8. The swapping between the two domains continues

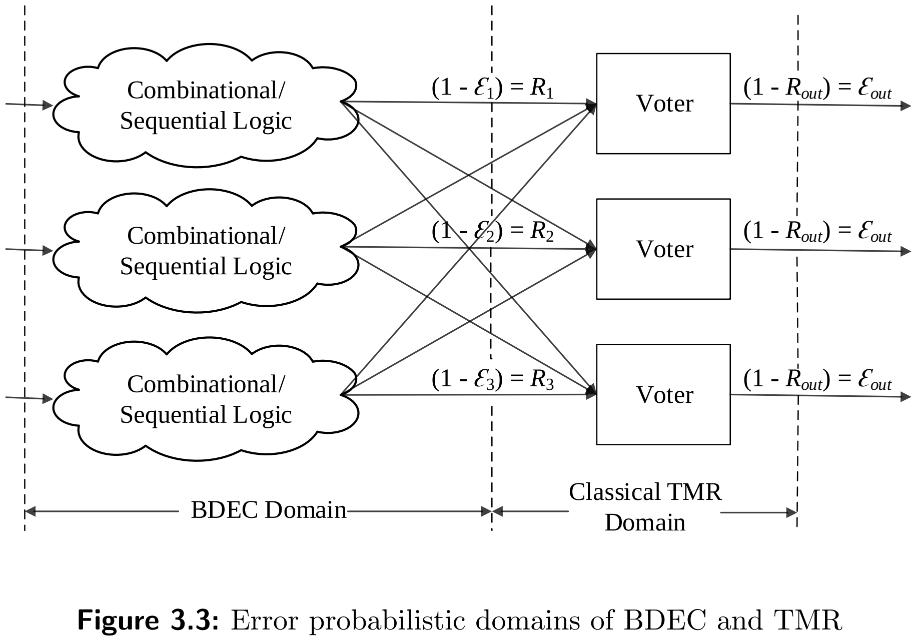

Combinational/ Sequential Logic
Combinational/ Sequential Logic
Combinational/ Sequential Logic
$(1 - \mathcal{E}_1) = R_1$
$(1 - \mathcal{E}_2) = R_2$
$(1 - \mathcal{E}_3) = R_3$
Voter
Voter
Voter
$(1 - R_{out}) = \mathcal{E}_{out}$
$(1 - R_{out}) = \mathcal{E}_{out}$
$(1 - R_{out}) = \mathcal{E}_{out}$
BDEC Domain
Classical TMR Domain

Figure 3.3: Error probabilistic domains of BDEC and TMR

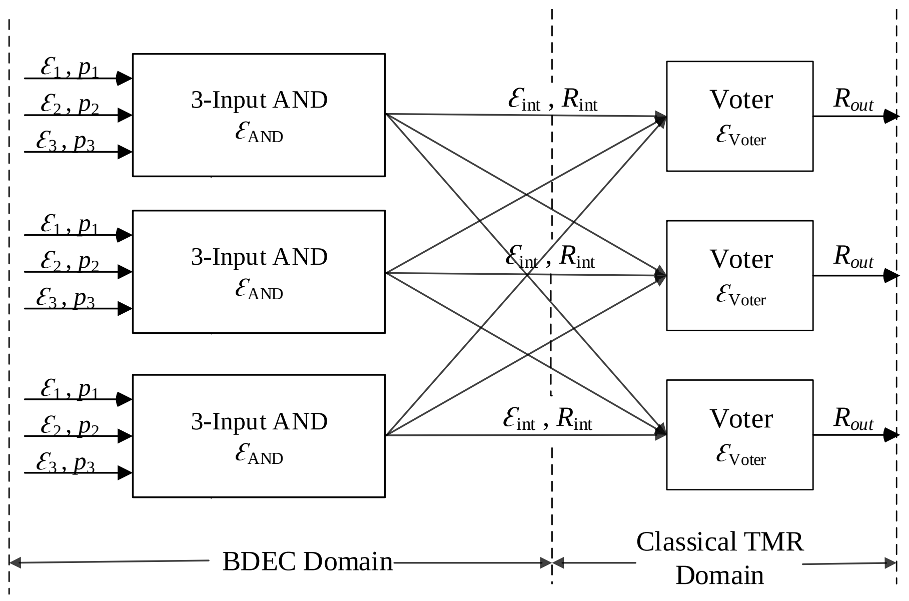


Figure 3.4: A redundant 3-input AND gate

unless the final output stage is reached.
The merger of the BDEC and instantaneous-time TMR model results in a combined timeless model with no notion of failure rate. Therefore, one must not use R_{out} to compute the probability of the system being error-free after a certain time period. In the combined model, the overall error propagation behaviour of the circuit is now different from the conventional BDEC in that the cumulative error does not always increase as we move forward from inputs to outputs. Instead, there exist voter elements which improve the reliability of the circuit at various stages inside the circuit structure. To illustrate this concept, we present an example of a 3-input AND gate as the combinational logic and calculate the reliability of its voted output, according to BDEC-only model and the combined BDEC-TMR model. The example circuit is shown in Figure 3.4 where ε_{int} and R_{int} represent the intermediate values of gate error probability and reliability respectively. This 3-input AND gate, for reference, is realized as a 3-input LUT in this example. The input error-probabilites of three inputs of the AND gate are assumed to be 1%, i.e., $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 0.01$. The p_n values are assumed to be 50%, i.e., $p_1 = p_2 = p_3 = 0.5$ whereas the gate error probabilities of AND gate and voter are taken 1%, i.e., $\varepsilon_{AND} = \varepsilon_{Voter} = 0.01$. The BDEC domain calculation results in $\varepsilon_{int} = 0.0173$ and $R_{int} = 98.3\%$. Finally, the BDEC-only model results in output reliability (R_{out}) of 96.5% whereas the BDEC-TMR model results in 98.9%. Compared to the intermediate reliability value, we can see that the combined model, in contrast to the BDEC-only model, is able to comprehend the voter concept by improving the output reliability. Therefore, it can also be concluded that the existence of more voters and redundant stages improve the overall circuit reliability.

Supporting Sequential Behaviour of Circuits

The original BDEC model supports only combinational circuits. We have extended BDEC to cover also sequential circuits in this work. To calculate the reliability of a sequential circuit, we have applied the loop-breaking and time-frame expansion technique for feedback paths as used in analysis of sequential circuits in [74, 75, 76]. The basic idea behind the time-frame expansion of a sequential circuit is to the represent it as time-referenced combinational portions connected in series mode. Loop breaking refers to breaking the feedback path in a sequential circuit and treating it as primary input in the first portion of series circuit. For the succeeding portions of the circuit, the output from the previous portion serves as input. To implement this concept in the error computation algorithm of BDEC, we initialize the error of the feedback paths with the same error probability values as we use for primary inputs of the circuit. Afterwards, we let the algorithm update the error of the feedback paths by propagating it through the circuit elements in feedback.
The obvious inference from the BDEC model is that the error will keep on increasing with each iteration, giving us the impression that the feedback path error will eventually approach to 1, i.e., 100%. However, it has been shown in [75] that for some components, the error converges to a high output reliability value after many clock cycles whereas it drops to dangerously low reliability values at other nodes in the circuit. In contrast, our revised BDEC model improves the reliability when the signal is passed through redundant portions in a circuit. Therefore, even at the nodes where the reliability drops at the outputs of sequential elements, we can use redundancy at the feedback paths to make the reliability converge to high values or make it oscillate between bounds. For this reason, it is always suggested to insert the voters in the feedback paths to avoid the propagation of errors in the feedback. In this thesis, we update the feedback error once. In future work, we intend to observe the behaviour of updating the feedback error by propagating the error for multiple iterations and check whether it converges to any fixed value or oscillates between bounds, and try to include results from current research in modeling reliability of sequential elements [77].

3.2 Automation of the BDEC Model

The default model of BDEC is extended with redundant circuits and sequential behaviour analysis as explained in the last section. For simplicity, we will continue to call the revised model as BDEC. Afterwards, the model will be automated in a form of MATLAB tool that can calculate reliability of a circuit of any size on the provision of input error probabilities. Note that since we have used the Xilinx FPGAs, we will refer to the softwares and netlist formats of Xilinx FPGAs only.

3.2.1 Inputs to the BDEC Tool

The BDEC tool has to be provided with the FPGA component netlist as well as control parameters, i.e., input and signal probabilities of BDEC model. In FPGA based circuits, a component (or gate) refers to all resources of an FPGA, e.g., lookup table (LUT), multiplexers (MUX), random access memory (RAM) blocks, etc.

FPGA Component Netlist

The BDEC model works by flow of probability from inputs to outputs of circuits represented in the form of logic components. However, the FPGA circuits are designed using behavioral HDL coding, i.e., via Verilog or VHDL. The synthesis software of the FPGA translates the behavioral description to the specific resources of the FPGA utilized to develop the hardware. The resource description file is called as .ngc netlist which is the Xilinx propriety netlist format. The .ngc netlist is converted to .edf format and later to structural Verilog format using Xilinx ngc2edif and netgen tools respectively. The structural netlist describes each of the component via names of its instance, inputs, outputs and instantiation string (if the component is a LUT). Hence, each component in this netlist could be analyzed separately.

Input Error and Signal Probabilities

The input error probabilities (ε_i) of only primary components are provided to the tool since input error probabilities of following components are equal to the output error probabilities of their preceding components. The gate or component error probability (ε_g) is transistor technology-dependent parameter and is common to all the components. The voter error probability (ε_{voter}) is taken equal to the gate error probability assuming similar device characteristics of voters compared to other components. The signal probability (p_i) of each component refers to the percentage of time (clock cycles) in which its output stays at logic 1. p_i has to be provided for each component in the netlist.

3.2.2 Programming Mathematical Model of BDEC

The BDEC tool calculates reliability of a single logic component in five stages programmed into MATLAB.

1. **Formulate the Boolean equation:** The Boolean equations of generic FPGA resources are known in advance, e.g., 2x1 multiplexer, and can be provided as hard inputs to the tool. The flipflops are treated as buffers. The remaining category of LUTs need more computation. The Boolean equations of LUTs can be generated by reading their instantiation strings and formulating their Boolean equations as sum of minterms. The bigger, custom and composite blocks in FPGA, e.g., DSP blocks are out of scope in this research.

2. **Compute the Partial and Full Derivatives:** The partial and full derivatives are computed according to the equations defined by Boolean difference calculus in [54]. The generic equations of each type of resource has to be hard-coded in the tool. As the number of inputs of the component increases, the number of derivatives that need to be evaluated increases, e.g., a 6-input LUT has 6 partial and 57 full derivatives.

Therefore, a library of all derivatives of each type of component has be developed for rapid calculation of component reliabilities.

3. **Compute the Signal Probability Functions:** The signal probability functions convert the Boolean variable equations into corresponding equations dependent on signal probabilities. Therefore, the signal derivatives formulated in the previous step are evaluated by substitution of the the Boolean equation and signal probability values (p_i) of each input of the component.

4. **Calculate Total Input Error:** The total input error, as demonstrated in Equation 3.6 is computed by utilizing the derivatives evaluated in the previous step and the input error probabilities.

5. **Calculate Total Output Error Probability:** The total output error probability of each component is evaluated by Equation 3.1 and the overall output error probability is calculated by averaging the error probabilities of all outputs of the circuit.

These programming steps can be enumerated by using the 2-input AND gate example, reproduced from the beginning of Section 3.1.

Step 1:

$$f = x_1 x_2 \tag{3.9}$$

Step 2:

$$\frac{\partial f}{\partial x_1} = x_2 \tag{3.10}$$

$$\frac{\partial f}{\partial x_2} = x_1 \tag{3.11}$$

$$\frac{\Delta f}{\Delta(x_1 x_2)} = \overline{x_1}\ \overline{x_2} + x_1 x_2 \tag{3.12}$$

Step 3:

$$Pr\{\frac{\partial f}{\partial x_1}\} = p_2 \tag{3.13}$$

$$Pr\{\frac{\partial f}{\partial x_2}\} = p_1 \tag{3.14}$$

$$Pr\{\frac{\Delta f}{\Delta(x_1 x_2)}\} = (1-p_1)(1-p_2) + p_1 p_2 = 1 - (p_1 + p_2) + 2p_1 p_2 \tag{3.15}$$

Step 4:

$$\varepsilon_{in} = \varepsilon_1(1-\varepsilon_2)Pr\{\frac{\partial f}{\partial x_1}\} + \varepsilon_2(1-\varepsilon_1)Pr\{\frac{\partial f}{\partial x_2}\} + \varepsilon_1\varepsilon_2 Pr\{\frac{\Delta f}{\Delta(x_1 x_2)}\} \tag{3.16}$$

$$\varepsilon_{in} = \varepsilon_1(1-\varepsilon_2)p_2 + \varepsilon_2(1-\varepsilon_1)p_1 + \varepsilon_1\varepsilon_2(1 - (p_1 + p_2) + 2p_1 p_2) \tag{3.17}$$

Step 5:

$$\varepsilon_{AND2} = \varepsilon_g + (1 - 2\varepsilon_g)\varepsilon_{in} \tag{3.18}$$

$$\varepsilon_{AND2} = \varepsilon_g + (1 - 2\varepsilon_g)(\varepsilon_1 p_2 + \varepsilon_2 p_1 + \varepsilon_1 \varepsilon_2 (1 - 2(p_1 + p_2) + 2p_1 p_2)) \tag{3.19}$$

The formulation of error probability equation has been further explained by three more examples in Appendix A.

3.2.3 BDEC Reliability Computation Algorithm

In this section, we describe the tool flow of the revised BDEC model implemented in MATLAB and represented as a flow diagram in Figure 3.5. The error propagation in this model is sequential, i.e., the error propagates from the primary inputs and passes through each element in the circuit unless it reaches the final outputs of the design. The input of the tool, i.e., Verilog structural netlist is interpreted in MATLAB as a simple text file. For analyzing this netlist, we use basic text-reading functions of MATLAB and use arrays and cells to store information about each of the component. The algorithm starts by splitting the used FPGA components into arrays according to different resource types, e.g., LUT3, RAM16x1, etc. Afterwards, we initialize the input and gate error probabilities. If the circuit contains any sequential elements, we treat the feedback paths as primary inputs and initialize their error probabilities in the same way as inputs. The inputs of all the components which are not connected by primary inputs are treated as wires. A general wire array holds the error and signal probabilities of each wire in the circuit design.
The algorithm is composed of multiple nested loops. The outermost loop checks if all the wires have their error probability values calculated or not. If it finds an unprocessed wire, i.e., an unprocessed component, it sequentially picks up a resource type array and parse it unless it finds a component whose output error probability has not been evaluated. In order to calculate the reliability of any circuit component, all of its input error and signal probabilities should be available which can be checked by the wire entry in wire array. If one or more of the input error or signal probabilities of this component are not available, then the component is left pending to be processed later. Upon availability of all the input requirements for error computation, the Boolean equation of this component is formulated. The Boolean equation is later used to calculate partial and full derivatives for this component and finally the output error probability, as explained by the steps in Section 3.2.2. This output error is updated in wire array and this component will not be processed again unless the resource-type is a flipflop whose output error probability will be updated once more. When the array for each resource type is wholly parsed, the next resource-type array is parsed similarly.
Once all the resource type arrays are parsed, the algorithm checks whether all the wires in the wire array have their error probabilities updated or not. Otherwise, the algorithm keeps running. This phenomenon is due to sequential error-propagation of this model where one component in a resource-type could be attached to the primary inputs while another element of the same resource-type could be connected to the circuit outputs.

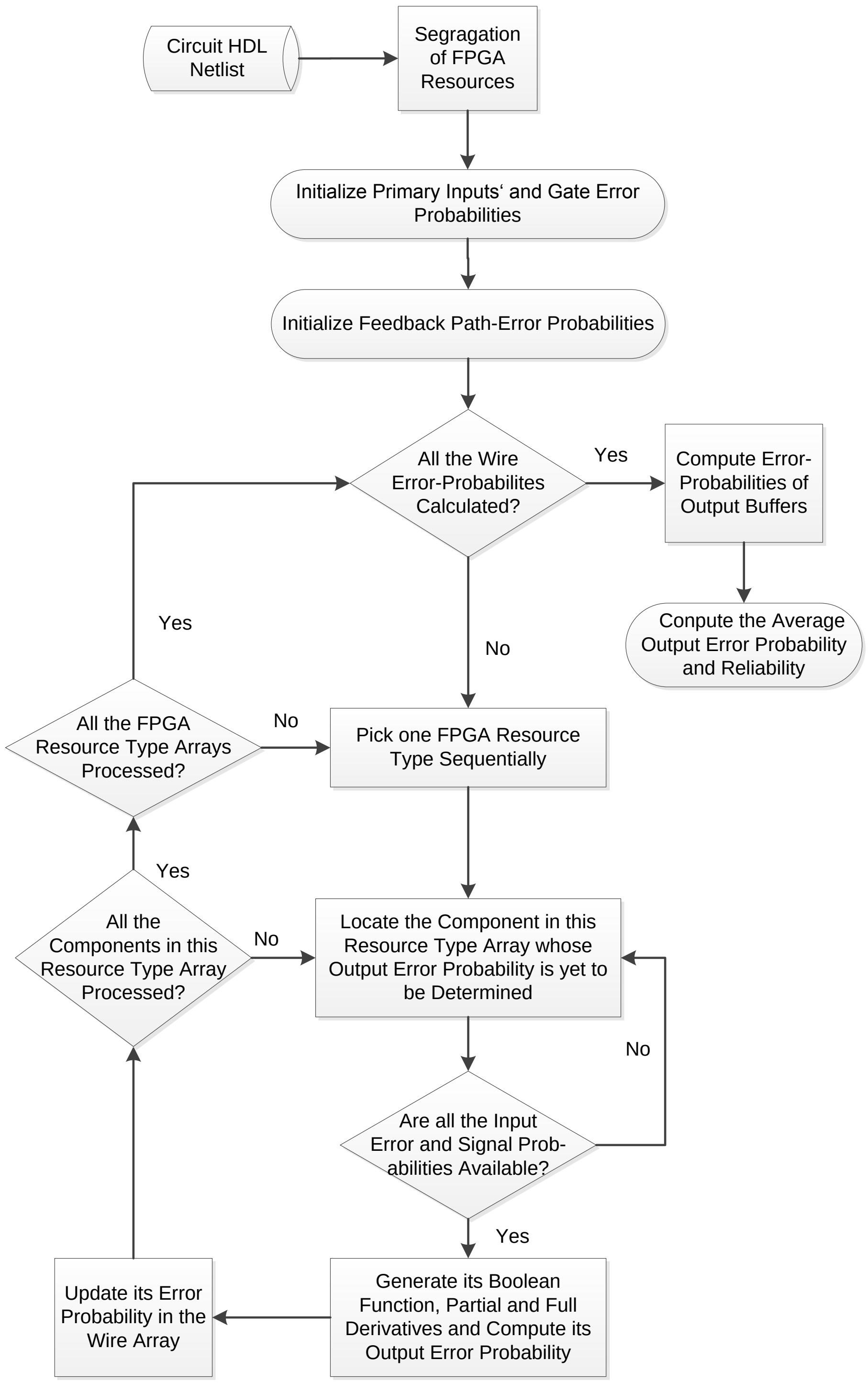

Circuit HDL Netlist
Segragation of FPGA Resources
Initialize Primary Inputs' and Gate Error Probabilities
Initialize Feedback Path-Error Probabilities
All the Wire Error-Probabilites Calculated?
Yes
Compute Error-Probabilities of Output Buffers
Conpute the Average Output Error Probability and Reliability
No
Yes
All the FPGA Resource Type Arrays Processed?
No
Pick one FPGA Resource Type Sequentially
Yes
All the Components in this Resource Type Array Processed?
No
Locate the Component in this Resource Type Array whose Output Error Probability is yet to be Determined
No
Are all the Input Error and Signal Prob-abilities Available?
Yes
Update its Error Probability in the Wire Array
Generate its Boolean Function, Partial and Full Derivatives and Compute its Output Error Probability

Figure 3.5: BDEC reliability computation algorithm

Finally, we calculate the output error probabilities of output buffers whose inputs are wires as well. Hence, the error probability of each output is computed and the result is averaged. The reliability is calculated by subtracting the average output error probability from unity.
The timing analysis of this algorithm shows that the time taken by the condition-check loops, e.g., whether a specific wire has its error probability updated or not is extremely fast in MATLAB. The time taken by the BDEC model is the real concern which highly increases with the number of large input components, e.g., 5- and 6-input LUTs. The total computation time of the algorithm varies from few seconds to few days depending on the circuit size though complexity of BDEC algorithm is still lower than major reliability evaluation techniques discussed in Section 2.6.2.

3.3 Parameter Variability Analysis

The output reliability of circuits with their default and redundant versions can be computed with the MATLAB tool we developed in this research. Before utilizing this tool, we perform a variability analysis of the control parameters of the BDEC model, i.e., gate error probability (ε_g), input error probability (ε_i), voter error probability (ε_{voter}) and signal probability (p_i) to observe the variation in output reliability. This analysis is absent in the literature due to unavailability of an automated tool. However, since each control parameter experiment requires computing reliability of a circuit for its multiple values, the overall experiment is extremely time-consuming when performed for large circuits. Therefore, we have chosen a basic combinational circuit, i.e., c17 from ISCAS85 benchmarks suite.
For each parameter variation experiment, we fix the other three variables. Unless otherwise stated, the default values of ε_g, ε_i and ε_{voter} used are 5%, i.e., 0.05 and p_i at 50%, i.e., 0.5. Following the single parameter analysis, we observe the joint affect of ε_g and ε_i on the output reliability. To check the variation of output reliability with redundancy insertion, we have used three circuit implementations, i.e., no-redundancy (NR), triple modular redundancy (TMR) and cascaded TMR (CTMR). The redundant implementations of benchmarks are generated from the BYU-LANL TMR tool [45]. Note that the BYU-LANL tool supports conversion of target HDL design to TMR equivalent only though we enhanced its capability to generate a cascaded TMR (CTMR) version of the circuit as well. The details of the enhanced BYU-LANL TMR toolflow can be found in Section 4.1.1. Note that in a practical radiation-based scenario, the error probabilities of gate, input and voter lie in a very small region, typically less than 1%, but in our analysis we vary these probabilities to 100% to see the possible effects of extreme error-rate on the output reliability. Moreover, high variations of error probability helps us to find the threshold point after which the redundancy offers no benefit on reliability.

3.3.1 Variation of Gate-Error Probability ε_g

Figure 3.6 shows the variation of output reliability with gate error probability variation from 0-100% keeping other parameters at default setting. The general observation is that the reliability decreases sharply with the gate error increasing from 0 to 20%. From 20-80% the decrease in reliability is almost only 15% while it again starts a sharp decrease afterwards. To check the redundancy affect, it is clear that CTMR remains most reliable up to gate error probability of 80% after which high redundancy has a reverse affect. This gate error probability value is named as threshold point, with respect to ε_g. The maximum difference in reliability from non-redundant to maximum redundancy, i.e., CTMR, is found to be around 10%. From 20-80% variation of ε_g, the reliability-difference among non-redundant and redundant implementations decreases though it increases in the reverse direction after 80% which proves that redundancy has an adverse affect in extreme radiation environments. Note that difference in reliability among redundant implementations is highly dependent on the circuit structure and the number of nodes in a circuit which are made redundant. In our case, we use the default redundancy decisions taken by the BYU-LANL tool though this variation can be increased by forced redundancy insertion and usage of different voter-insertion algorithms [47].

3.3.2 Variation of Input Error-Probability ε_i

Variation in output reliability by changing input error probability is shown in Figure 3.7. Compared to the ε_g plot, the decrease in output reliability is less sharp though the threshold point after which redundancy has an adverse affect on reliability approaches earlier. The maximum positive affect due to redundancy is again around 10%. Moreover, after the threshold point, the negative effect of redundant configurations is high which calls

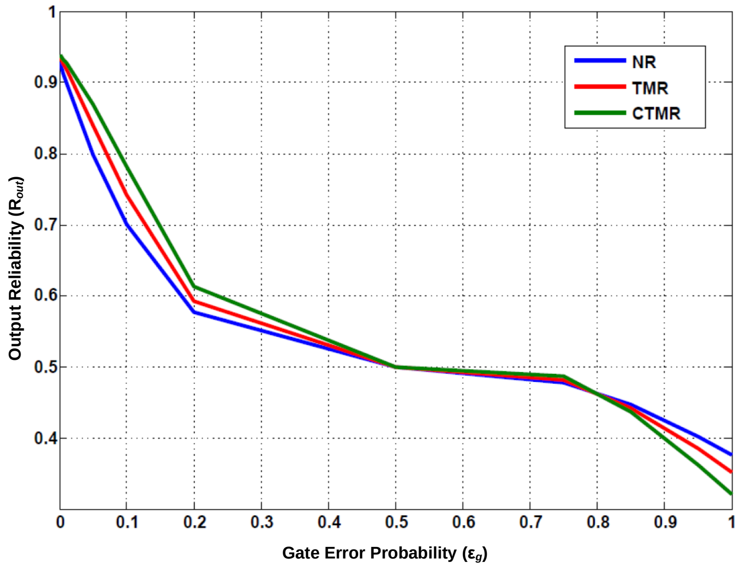


Figure 3.6: Variation of output reliability of c17 benchmark with gate error probability (ε_g)

for having as low input error as possible. Hence, a high input error is more severe to be handled by redundancy as compared to the case when individual circuit components are erroneous or have high ε_g.

3.3.3 Variation of Signal Probability p_i

Following Figure 3.8, this is the least vulnerable parameter affecting the circuit output reliability as the total variation of reliability for a single circuit implementation is less than 1%. The maximum reliability, however is obtained for p_i to be at 60%, i.e., for probability of each wire being at logic 1 at 60%. Note that though we have assumed a 50% signal probability values for all nodes, as practiced in previous research works [54, 61], this strategy does not wholly represent a real simulation scenario. In reality, the signal probability of every node/wire in a circuit can only be recorded by simulating the circuit with a testbench. Due to numerous nodes in large circuits, the signal probability of each node can be measured only via developing another automated tool that interprets the testbench simulation file, i.e., value change dump (VCD), which is an ASCII propriety format of Xilinx. The development of this additional MATLAB module is subject to our future work due to complexity and time needed for its formation.

3.3.4 Variation of Voter Error-Probability ε_{voter}

Voter error probability is the most vulnerable parameter of the variability analysis. As can be seen from Figure 3.9, the reliability of the circuit drops sharply with voter error

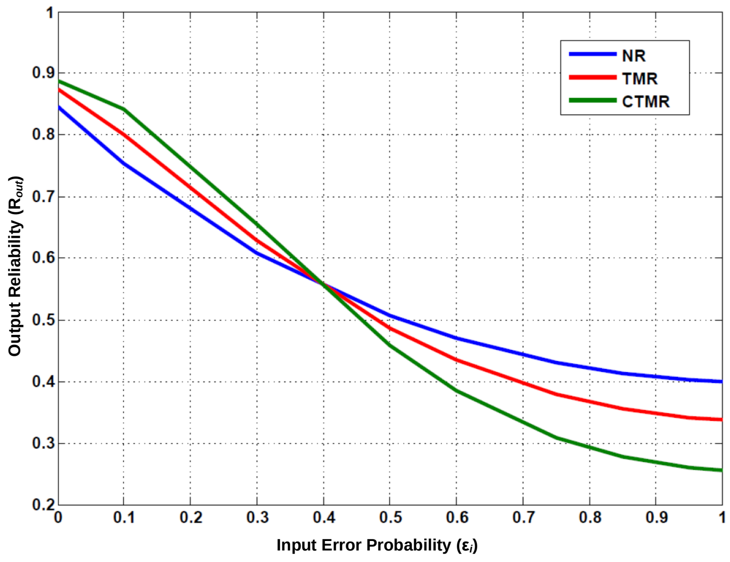


Figure 3.7: Variation of output reliability of c17 benchmark with input error probability (ε_i)

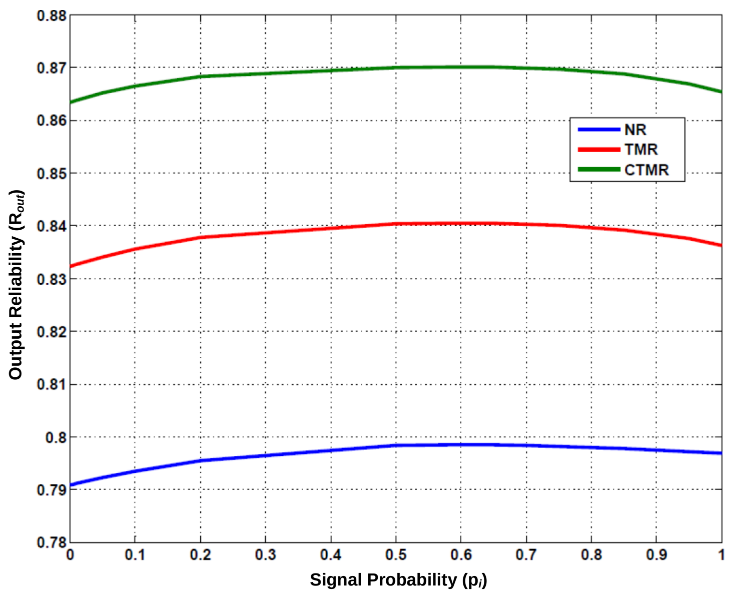


Figure 3.8: Variation of output reliability of c17 benchmark with signal probability (p_i)

probability exceeding 10%. The redundancy plays an opposite role with higher orders of redundancy being more disastrous as they have multiple stages of voters. Therefore, its highly recommended to have more reliable voters than other circuit components in particular. Using the concept of differential reliability, we can selectively use higher power supply and threshold voltages for voters which is more tricky to be done on FPGAs than ASICs.

3.3.5 Joint Variation of Input and Gate Error-Probabilities

Figure 3.10 shows the joint effect of gate error and input error on the output reliability. The variation is observed for NR, TMR and CTMR implementations of the c17 benchmark separately. It can be observed that the graphs are symmetric, i.e., the output reliability tends to increase when ε_g and ε_i simultaneously approach either 0% or 100%. This is due to the masking effect of gate error on input error and vice versa. Comparing the three implementations, reliability of 80% or higher is achieved for joint variation of ε_g and ε_i in approximately 0-10% or 90-100% intervals where these intervals increase in span for higher orders of redundancy. Similarly, the output reliability variation of 60-80% lasts for a longer duration for high redundancy levels. Therefore, the mid-level output reliability range of 40-60% decreases with increase in redundancy. On the contrary, the reliability decreases when one of the parameters approaches 0% error and the other towards 100% where the masking effect of these parameters on each other is minimal. Moreover, the low reliability region, i.e., 20-40%, also increases for higher redundancy levels.

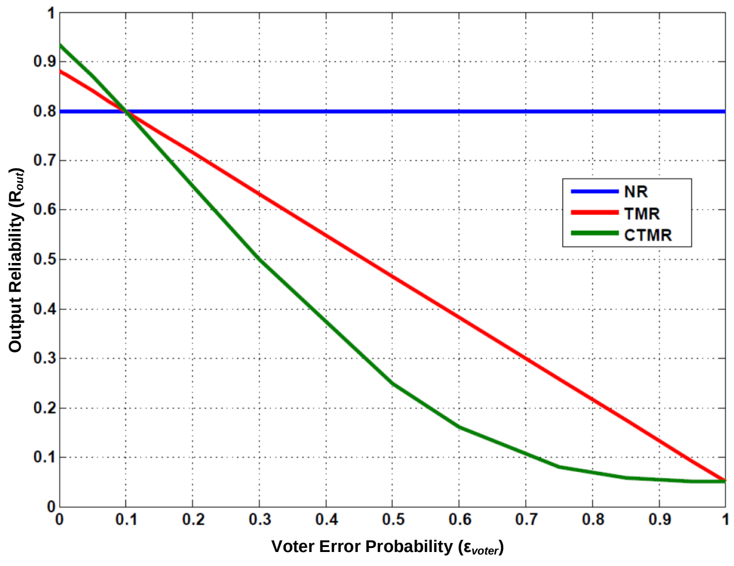
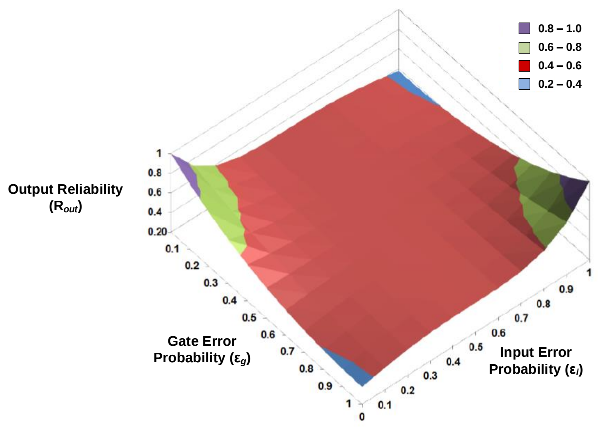


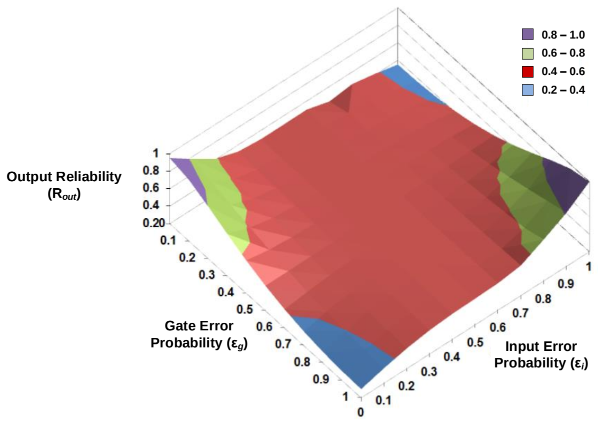
Figure 3.9: Variation of output reliability of c17 benchmark with voter error probability, ε_{voter}

3.4 Chapter Conclusion

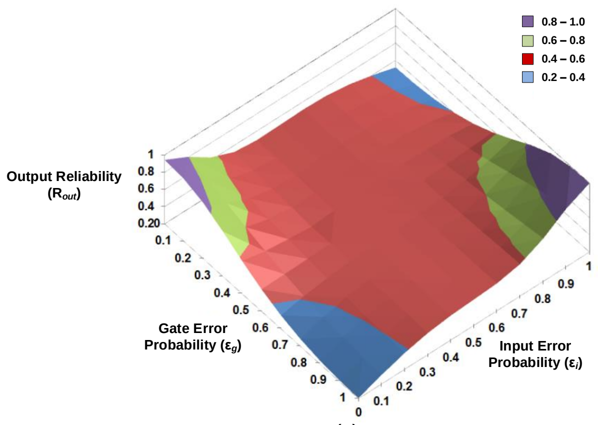
In this chapter, we have analyzed different reliability computation methods according to their application domain. Based on the analysis, we have chosen Boolean difference error calculation method (BDEC) to analyze, evaluate and compare different redundant structures implemented on the FPGA. However, we have found that the default BDEC model lacks in two domains, i.e., interpreting the redundant circuit behavior and analysis of sequential circuits. Therefore, we have extended this model in these two directions. The revised BDEC model is then automated in the form of a MATLAB tool that is able to compute reliability of any FPGA based circuit available in the form of structural netlist. Using this tool, we have performed a variability analysis of output reliability based on the error probabilities of logic components, inputs and voters in addition to the signal probabilities of the wires and inputs. We have learned from this analysis that the voter error-probability is the most vulnerable parameter that could affect the output reliability. We have also concluded that the benefit of redundancy could be achieved up to only a certain limit of error probability, after which the higher redundancy levels result in decrease in circuit's reliability. The transition point of this redundancy-impact behavior is termed as threshold point in this research. Though the variability analysis in this chapter has been limited to a single benchmark circuit, in Section 5.3, we compare the threshold point variation for a complex sequential circuit as well.



(a)



(b)



(c)

Figure 3.10: Variation of output reliability of c17 benchmark with input and gate error probabilities for (a) NR (b) TMR and (c) CTMR circuit implementations

CHAPTER 4

Dynamic Reliability Management

In this chapter, we present our novel technique of Dynamic Reliability Management (DRM). DRM is based on the adaptive fault-tolerance concept explained in Section 2.5. Revisiting this concept, adaptive fault-tolerance refers to the reconfiguration of FPGA hardware based on the reliability requirements and thus optimizing the trade-off between performance and reliability of an application at run-time. The major research works following the concept of adaptive fault-tolerance are discussed in Section 2.7. In contrast to other adaptive fault-tolerance schemes, DRM considers more performance parameters and is completely a self-adaptive approach. The structure of DRM is composed of two parts, i.e., design-time and run-time.

4.1 DRM Design-Time Circuit Analysis

We have learned in Section 2.3.2 that there are multiple variations of redundant architecture of a hardware design, where each implementation differs in performance factors of area, latency and power as well as the achieved reliability. Since the reliability of an implementation increases on the cost of additional area, latency and power consumption, this trade-off needs to be analyzed before utilizing a specific redundant implementation. The design-time tool flow of DRM focuses on analyzing this trade-off for four redundancy configurations (discussed in Section 2.5.1) and additionally the default non-redundant hardware design, as abbreviated below.

- NR: No redundancy
- OAV: One alternate voter
- TAV: Two alternate voters
- TV: Triplicated voter

- CTMR: Cascaded TMR- Level 1

Furthermore, we have discussed eight voter-insertion algorithms in Section 2.4.1 that we are going to utilize in the design-time analysis of DRM. They are abbreviated as follows.

- CC: Connectivity cutset
- AFC: After flipflop cutset
- BFC: Before flipflop cutset
- BD: Basic decomposition
- HFC: Highest fanout cutset
- HFFC: Highest flipflop fanout cutset
- HFFIC: Highest flipflop fanin input cutset
- HFFOC: Highest flipflop fanin output cutset

Hence, our design-time analysis consist of 32 redundant implementations of a hardware design based on four redundancy configurations and eight voter-insertion algorithms. Each of the implementation is rated by the area (slice usage), latency (max. clock frequency), dynamic power consumption and reliability. The resulting design space is Pareto-optimized on the basis of four factors described above. Before presenting our merged tool flow, we give a brief overview on the BANL TMR tool which is used to generate redundant implementations of the target HDL design.

4.1.1 BYU-LANL TMR Tool

The BYU-LANL (BANL) TMR tool converts a hardware design, in an EDIF format, to two redundant implementations, i.e., duplication with comparison (DWC) [40] and triple modular redundancy (TMR) [9]. There are multiple options with this tool that gives the freedom to user for implementing error detectors, make individual components redundant or use partial redundancy [78]. We briefly describe the functionality of BANL tool flow including the optional sub-tools for additional functionalities mentioned above.

JEdifBuild

JEdifBuild creates merged netlists in a .jedif file format from multiple .edf files. This tool, by default, also flattens the design, performs shift-register LUT (SRL) replacement and half-latch removal. The .jedif file format is an intermediate file format used by the following replication tools.

JEdifAnalyze

JEdifAnalyze performs basic circuit analysis necessary for subsequent executables. In particular, it performs feedback and IOB analysis. The results of JEdifAnalyze are saved in a circuit description file (.cdesc) required by the following tools.

JEdifNMRSelection

JEdifNMRSelection determines which parts of a design will be replicated. This executable can be run in multiple passes to select different parts of a design for different kinds of replication. Each run of JEdifNMRSelectionn can select portions of a design for a single replication type, i.e., duplication or triplication. Design portions can be selected for replication based on available space or specific cell types, instances, ports, and clock domains specified by the user. The results of JEdifNMRSelection are saved in a replication description (.rdesc) file.

JEdifVoterSelection

JEdifVoterSelection determines the locations where voters will be inserted into a triplicated design. Voter locations are determined using a choice of eight voter-insertion algorithms. The results are added into the replication description (.rdesc) file.

JEdifNMR

JEdifNMR performs the replication selected by previously run tools. Information about what to replicate and where to insert voters/detectors is obtained from the replication description (.rdesc) file created by the previous steps.

JEdifMoreFrequentVoting (Optional)

JEdifMoreFrequentVoting inserts extra voters for more frequent voting within a design based on a logic levels threshold or a total number of desired partitions.

JEdifDetectionSelection (Optional)

JEdifDetectionSelection determines error detector locations for both triplicated and duplicated design portions using user-specified options. Results are saved in the replication description file (.rdesc).

JEdifPersistenceDetection (Optional)

JEdifPersistenceDetection determines additional error detector locations necessary for classifying persistent/nonpersistent errors detected in a design. Results are saved in the replication description (.rdesc) file.

4.1.2 DRM Design-time Tool Flow

Our design-time tool flow of DRM is illustrated in Figure 4.1. In summary, it converts a benchmark HDL design into a set of 4-dimensional Pareto-filtered implementations rated by the reliability magnitude, area consumption, latency and dynamic power consumption. This overall tool flow is constructed by utilizing, extending and creating various tools as described below.

- Tools Utilized: Xilinx ISE (Mapping, Placement and Routing, Power Analyzer), MATLAB Pareto filter
- Tool Extended: BANL TMR Tool
- Tool Created: MATLAB BDEC Tool

The extended sub-tools of BYU-LANL TMR tool, i.e., JEdifNMRSelection and JEdifVoterSelection and the newly created tool, i.e., Revised BDEC tool are marked as dark shaded blocks in Figure 4.1 to highlight our contribution areas in the tool chain. The steps used in tool flow are discussed as follows.

Xilinx ISE Synthesis

In the first stage, we synthesize the benchmark HDL design, whether behavioural or structural, with Xilinx ISE. The synthesis generates an EDIF netlist file containing the specific FPGA components needed to realize the HDL description of the application. The EDIF netlist is passed to the BANL TMR tool.

Replication via the BANL TMR Tool

The original BANL TMR tool (based on the Java programming language) was extended to support additional features which we explain as follows.

Original BANL Tool Flow The default version of the tool performs logic replication which we categorize in four major stages:

1. The first stage comprising *JEdifBuild* and *JEdifAnalyze* performs the technical steps of design flattening, circuit and IOB analysis, etc., and saves the information in intermediate files to be used by the following tools.
2. The second stage comprising *JEdifNMRSelection* and *JEdifVoterSelection* determines the type of configuration and replication to be used, replicates the instances, determines the voter-insertion locations by the specific algorithm used and makes the necessary wire connections.
3. The third stage can be run if more voters and error-detectors are desired. For our research, we are excluding this stage as an optional one.

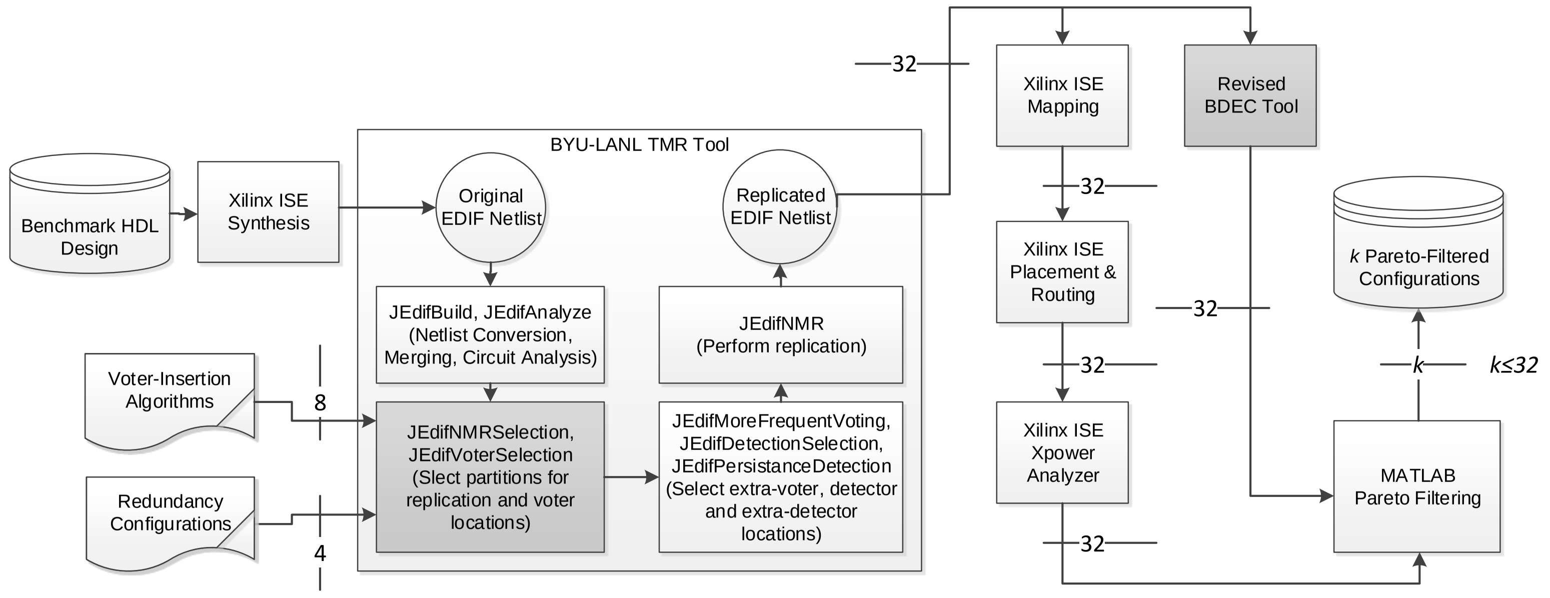

Benchmark HDL Design
Xilinx ISE Synthesis
BYU-LANL TMR Tool
Original EDIF Netlist
Replicated EDIF Netlist
JEdifBuild, JEdifAnalyze (Netlist Conversion, Merging, Circuit Analysis)
JEdifNMR (Perform replication)
Voter-Insertion Algorithms
Redundancy Configurations
8
4
JEdifNMRSelection, JEdifVoterSelection (Slect partitions for replication and voter locations)
JEdifMoreFrequentVoting, JEdifDetectionSelection, JEdifPersistanceDetection (Select extra-voter, detector and extra-detector locations)
32
Xilinx ISE Mapping
Revised BDEC Tool
32
Xilinx ISE Placement & Routing
32
32
Xilinx ISE Xpower Analyzer
32
k Pareto-Filtered Configurations
k
k≤32
MATLAB Pareto Filtering

Figure 4.1: DRM deisgn-time tool flow

4. The final stage does the actual replication by reading the intermediate file formats written by the previous tools and generating the replicated netlist.

Extension of BANL Tool We have made two extensions to the original BANL tool in order to support the additional redundancy configurations. By default, the tool supported only TMR configuration which we extended to three more configurations namely one alternate voter (OAV), two alternate voters (TAV) and cascaded-TMR (CTMR). Additionally, we have changed the command-line interface of the *JEdifNMRSelection* tool from replication type to configuration type because there is more than one configuration that uses triplication. As a result, we now can decide on a configuration instead of a replication type. Note that the duplication with comparison (DWC) configuration is out of scope for this research.

Performing Replication

After running the first stage of the tool, we need to choose one of the four redundancy configurations via the *JEdifNMRSelection* tool and one of the eight voter-insertion algorithms via the *JEdifVoterSelection* tool. The final stage of *JEdifNMR* reads the intermediate file formats of previously run tools and writes the final replicated EDIF netlist. Overall, we run the extended-BANL TMR tool for 32 possible combinations of redundancy configurations and voter-insertion algorithms, which sums up the design implementation set.

Xilinx ISE Mapping, Placement/Routing and Power Analysis

All of the 32 generated implementations of the design are passed through Xilinx ISE mapping, placement/routing and XPower analyzer tools to obtain slice utilization (for area consumption), pad-pad delay/max. clock frequency (for latency) and dynamic power consumption, respectively. We resort to the dynamic power consumption since the static power almost remains the same throughout the analysis of a single benchmark.

Reliability Evaluation

For reliability evaluation, we utilized the automated version of BDEC tool developed in this research work, and explained in Chapter 3. All the 32 redundant implementations of the HDL design are evaluated for reliability based on the input error probabilities of inputs and voters with signal probabilities of wires. The details of the BDEC, our extensions to the BDEC mathematical model and an automated reliability tool development using BDEC can be found in chapter 3.

Pareto Filtering of Implementation Points via MATLAB

Using the Pareto Front function of MATLAB [79], we reduce the set of implementation points to non-dominated ones to make the selection of trade-off points easier. Each implementation is characterized on the basis of area, latency, power and measured reliability.

The resulting non-dominated implementation points can be utilized by the system designer on the basis of one or more constraint requirements of area, latency, power and reliability.

4.2 DRM Run-Time Circuit Analysis

The final outcome of the design-time tool flow is the set of non-dominated redundant configurations in addition to the original hardware design of the application module. The run-time system, on the other hand, stores and utilizes these configurations for balancing the reliability-performance trade-off using an operating system support. Another component of the run-time system is the decision module which can be implemented in software or hardware. Before explaining our composite run-time tool flow, we give an insight into the operating system, ReconOS, used for the DRM implementation.

4.2.1 ReconOS

The operating system used for DRM on a platform FPGA is ReconOS [80, 81, 82, 83]. ReconOS is an operating system for reconfigurable system-on-chip that extends the multi-threaded programming model from software to reconfigurable logic cores. Its programming model and system architecture offers unified operating system services for functions executing in software and hardware and a standardized interface for integrating custom hardware accelerators. ReconOS leverages the well-established multi-threading programming model and extends a host operating system with support for hardware threads. These extensions allow the hardware threads to interact with software threads using the same, standardized operating system mechanisms, for example, semaphores, mutexes, condition variables, and message queues. From the perspective of an application it is thus completely transparent whether a thread is executing in software or hardware.
The ReconOS run-time system architecture provides the structural foundation to support the multi-threading programming model and its execution on CPU/FPGA platforms. Figure 4.2 shows a conceptual view of a typical system that is decomposed into application software, OS kernel and hardware architecture. The application software threads are usually executed on the main CPU alongside the host OS kernel that encapsulates APIs, libraries, and all programming model objects as well as lower level functions such as memory management and device drivers. The ReconOS run-time environment consists of hardware components that provide interfaces, communication channels, and other functionality such as memory access and address translation to the hardware threads. Additionally, the runtime system comprises software components in the form of libraries and kernel modules that offer an interface to the hardware, the operating system, and the application software threads.
A key component for multi-threading across the hardware/software boundary is the delegate thread, which is a light-weight software thread that interfaces between the hardware thread and the operating system. When a hardware thread needs to execute an operating system function, it relays this request through the operating system interface (OSIF) to the delegate thread using platform-specific (but application-independent) communication

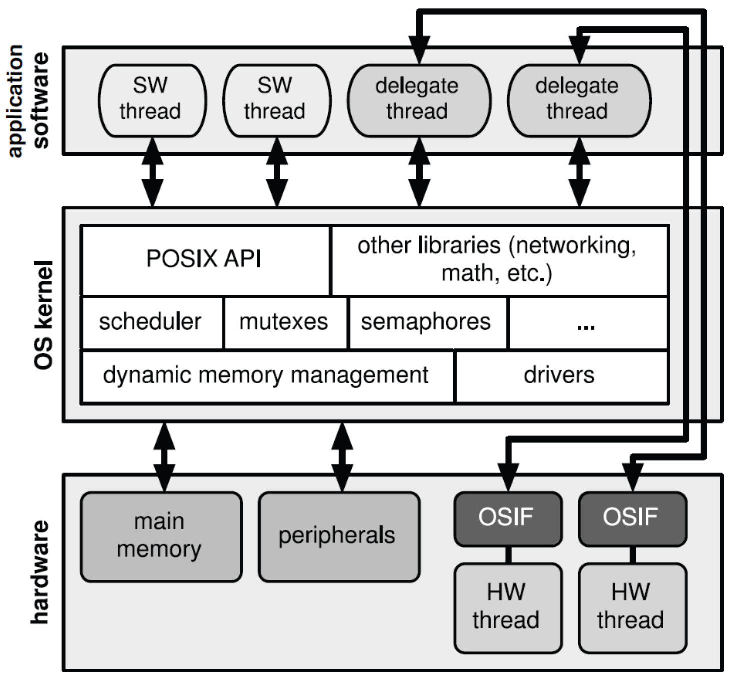


Figure 4.2: Conceptual overview of ReconOS system architecture [83]

interfaces. The delegate thread then executes the desired operating system functions on behalf of its associated hardware thread. Hence, from the OS kernel point of view, only software threads exist and interact, while the hardware threads are completely hidden behind their respective delegate threads. From the application programmer point of view, however, the delegate threads are hidden by the ReconOS runtime environment and only the application hardware and software threads exist. This delegate mechanism together with the unified thread interfaces gives ReconOS exceptional transparency regarding the execution mode of a thread, i.e., whether it runs in software or hardware.
Hardware threads reside in reconfigurable slots, which are predefined areas of reconfigurable logic equipped with the necessary communication interfaces. Besides communicating with the OS kernel on the host CPU, hardware threads residing in reconfigurable slots can also access the system memory. To that end, a hardware thread uses its memory interface (MEMIF) to connect to the ReconOS memory subsystem. The memory subsystem arbitrate and aligns the hardware threads memory requests and can handle single word as well as burst accesses. Hardware threads use FIFOs to communicate with the memory subsystem; one outgoing and one incoming FIFO per hardware thread. Requests for memory transactions are encoded and written to the outgoing FIFO followed by data in the case of a write request. In the case of a read request, data become available on the incoming FIFO upon completion of the memory transfer. A library of VHDL procedures is developed to conveniently handle memory operations. These procedures encode the requests, synchronize with the memory FIFOs, and automatically transfer data from/to

local memory elements within the hardware thread. Since ReconOS supports partial reconfiguration, both hardware and software threads can be instantiated, loaded and started at run-time.

4.2.2 Decision Mechanisms for Changing Reliability Levels

In this section, we discuss the possible criterion on the basis of which the adaptive system could switch among different redundancy/reliability levels. We envision four possible *decision mechanisms*. The mechanisms are denoted as i) external, ii) time, iii) cooperative and iv) radiation/error-rate based mechanisms. Table 4.1 lists the four decision mechanisms and shows whether the system, e.g., the satellite, needs a radiation sensor and a so-called decision module that decides on changing the required level of reliability.

External

With an external decision mechanism, a remote user has the control over reconfiguring the reliability configurations. For example, a ground control center transmits a signal to reconfigure the reliability levels of the satellite application, based on available information about space weather. In this way, the radiation data is recorded by a source external to the satellite and the decision is made by the control station. Hence, both of the components of the decision mechanism are external to the satellite.

Time

The decision mechanism based on time can be used for missions where the radiation pattern is already known. For example, in Figures 2.10 and 2.11, the pattern of radiation can be used to plan the reliability reconfiguration after fixed time intervals, calculated on the basis of the travel time of the satellite. With this mechanism, the time-based decision module stays within the application system though there is no sensor involved.

Cooperative

Since the decision module is the most critical part of decision mechanism, either we could protect it by excessive hardening or decide to keep it out of the application system. In this way, the decision is taken out of the system but the radiation data or proportional information, e.g., online error-rate measurement, is taken from the system which can be

Decision Mechanism	Sensor	Decision Module
External	No	No
Time	No	Yes
Cooperative	Yes	No
Radiation/Error	Yes	Yes

Table 4.1: Decision Mechanisms

cross-verified before taking the decision for a reconfiguration. Hence the term *cooperative* refers to the cooperation between the application system and the control station.

Radiation/Error

With this technique, the sensor and the decision module both lie within the application system. Moreover, this is the only self-adaptive decision mechanism which is responsible for collecting radiation data, its interpretation and the reliability reconfiguration. In comparison to a time-based reconfiguration, that would be sufficient in cases when the radiation plot is known in advance, the radiation/error-based approach can also handle unexpected situations or uncertainty of the radiation plot as, for example, shown in the radiation sensor data plotted in Figure 2.9. If one wants to maximize system reliability, a self-adaptive decision mechanism is to be adopted even if the probability of such unexpected radiation changes is very low.

4.2.3 DRM Run-time Tool Flow

The DRM concept requires two additional components over a standard ReconOS system, the decision module running as a software thread on the main processor and the database containing all the implementation variants for the hardware designs. Figure 4.3 depicts the exemplary ReconOS architecture with the additional DRM components as dark shaded blocks. The implementation of decision module in hardware and alternate storage possibilities of redundant configurations apart from DRAM are also possible though investigating the difference in performance due to these alternate strategies correspond to our future work. Since we focus on hardware redundancy only in this work, we will implement our

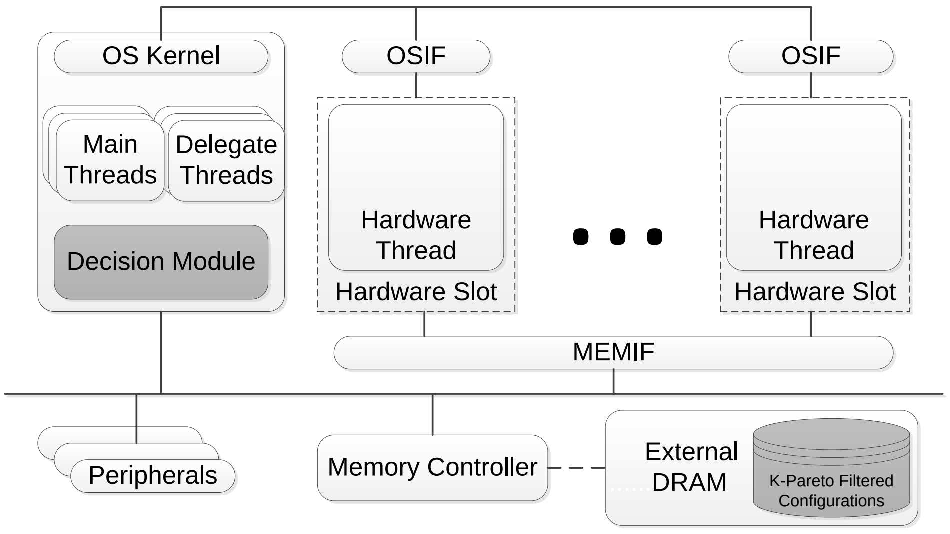


Figure 4.3: Exemplary ReconOS architecture

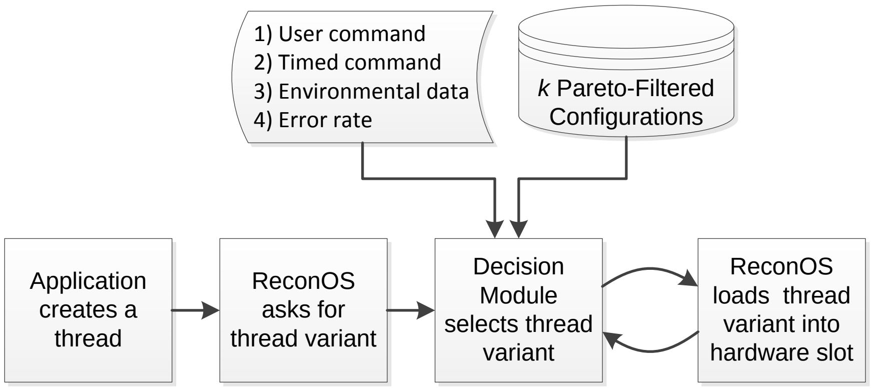

1) User command
2) Timed command
3) Environmental data
4) Error rate
k Pareto-Filtered Configurations
Application creates a thread
ReconOS asks for thread variant
Decision Module selects thread variant
ReconOS loads thread variant into hardware slot

Figure 4.4: ReconOS runtime flow with DRM

target applications as only hardware threads. We use the ReconOS version 3.0 in this work with supports the Xilkernel/Linux platform for the Microblaze processor.
The decision mechanisms for reliability can be used with ReconOS as shown in Figure 4.4. When an application starts and instantiates a hardware design, it creates the corresponding hardware thread and requests ReconOS to load the hardware thread into one of the hardware slots. Subsequently, ReconOS calls the DRM decision module to decide for an actual implementation variant for that thread. Depending on the used approach, the decision module is driven by user commands, time events, or measurements of the radiation level or actual error rates, as explained in the previous subsection. Afterwards, ReconOS retrieves the selected implementation variant from the database, i.e., external DRAM in our case, and configures it into a hardware slot. Any time during operation, the decision module may request ReconOS to reconfigure a hardware thread with an alternative implementation variant. In this research, we focus on reliability of hardware designs which are mapped to ReconOS hardware threads. Eventually, the reliability of the overall ReconOS system will have to be considered including the main CPU, the threads' operating system and memory interfaces (OSIF and MEMIF), buses, and memory controllers. The simplest way is to configure the ReconOS system to highest reliability implementation to ensure reliable operation of the switching mechanism, which adds a fixed cost of redundancy of the base system. The database of implementation variants is stored in external DRAM to allow for a fast reconfiguration. Reliability for external DRAM can be provided through error correction codes.

4.3 Chapter Conclusion

In this chapter, we have described the tool flow for our adaptive fault-tolerance technique, i.e., Dynamic Reliability Management (DRM). The design time and run-time portions of the DRM tool flow are described along with details of the tools utilized, extended and developed. It has been shown in this chapter how to use the design-time DRM tool flow to generate a Pareto-optimized set of non-dominant redundant circuit implementations, on the basis of performance factors of area, latency, power and reliability. The run-time tool flow, utilizes these Pareto implementations, by reconfiguring them into reconfigurable hardware slots. The decision on which redundant implementation to configure, and when to configure it, is taken by a decision module. The implementation of a sensor or radiation/error-rate data calculation is out of scope for this work, however, we will utilize a radiation profile from literature to take the respective decisions, as presented in the next chapter.

CHAPTER 5

Validating Dynamic Reliability Management Tool Flow

To validate the conceptual model of DRM and its tool flow, we conduct experiments on various benchmarks in this chapter to validate the design-time and run-time tool flows of our technique. For design-time part, we use six ISCAS benchmarks, with different architectures, with a 32-point implementation set to observe how the performance factors scale with the circuit size and architecture. In contrast, the run-time part is verified with two practical case studies with a 3-point implementation set to make the reconfiguration process easy for the reader to comprehend.

5.1 Validating Design-time Tool Flow

In order to analyze the variation in performance factors, i.e., area consumption, latency, dynamic power consumption and reliability with respect to changes in the redundancy configuration and voter-insertion algorithm, we report on and analyze six benchmark HDL designs from three classes of benchmarks with different circuit architectures [84]: c17 and c3540 from ISCAS'85, s713 and s838 from ISCAS'89, and b8 and b12 from ISCAS'99 benchmark suites.

5.1.1 Experimental Setup

We have experimented with overall four redundancy configurations including OAV (one alternate voter), TAV (two alternate voters), TV (triplicated voters), and CTMR (cascaded TMR level 1) from Subsection 2.3.2, as well as NR, a non-redundant design for comparison. The target device used is a Virtex 5 FPGA, XCVTX150T, with package FF1156. We have set the optimization method for design mapping to *balanced* to ensure the best combination of area and speed efficiency. For latency comparison, we use the maximum pad-pad delay and maximum clock frequency for combinational and sequential circuits, respectively. To determine dynamic power consumption, we have applied random

testbench signals to each benchmark while trying to maximize the signal activity among the circuit intermediate nodes. Due to lack of standard testbenches available for these benchmarks, our randomly applied testbenches do not necessarily account for the maximum possible power consumption. However, for the sake of comparison among redundant configurations, they serve the purpose.
The reliability results in this experimentation are interpreted as the average reliability of the outputs of the circuit. The output reliabilities are based on the error-probabilites of input, component (LUT, MUX, RAM, etc) and voter taken as 1%. The overall output error-probability is an indicative of each component and input to be probabilistically affected by an error at a rate of 1%. In reality, this is an extremely high over-estimate of component uncertainty which can be verified from device reliability reports having error-rates in the range of 1-10 errors in one billion hours [46, 63]. Using such low error-rates, the reliability is normally represented in MTBF units in conventional reliability theories. In contrast, the representation of reliability in probabilistic methods is the probability of error-free computation at a certain snapshot of time during the operation time span of the circuit. However, when such low error-probabilities are used in BDEC model, the reliability results gets complicated to understand by the user, i.e., the reliabilities among different implementations vary at smaller decimal places. Therefore, we resort to a component error of 1% to make the results comprehensible by the reader. Its worth mentioning here that the size of the benchmarks chosen are moderate in this experimentation (max. 800 slices) since the current version of our BDEC-reliability tool is very time-intensive due to the sequential flow of probability in this model. While we are making efforts to improve the performance of this tool by parallel programming and multi-core processing in the future; the reader can still observe the analysis on performance parameters (excluding reliability) for large benchmarks in our previous work [5].

5.1.2 c17 and c3540 (ISCAS'85)

These benchmark designs are purely combinational circuits and, by default, the BANL TMR tool inserts only single voters for combinational circuits. Without sequential elements the configurations OAV, TAV and TV produce the same results since no triplicated or alternate-triplicated voters are used. Also, the choice of voter-insertion algorithm does not matter because voter insertion also varies only with sequential elements. However, the default decisions taken by the tool can be overridden by forced redundancy insertion by the user though we resort to default decisions taken by BANL tool in this research.
Table 5.1 presents the results for the two ISCAS'85 benchmark designs. The reliability configurations OAV, TAV and TV have identical performance factors. The slice usage scale by a factor of 3 and 5 for c3540 for triplicated (TMR) and nine-plicated (CTMR) versions instead of theoretical scale of 3 and 9 respectively. The variations in latency, i.e., maximum pad-to-pad delay, is less pronounced for c17 than for c3540, which is a much larger design. For c17, there is even a slight decrease in latency with increasing reliability configurations. This decrease in latency as well as lower area scale factor for CTMR versions is due to automatic placement and routing which finds more optimization potential in larger

		# Slices	Max Pad-Pad Delay (ns)	Dynamic PD (W)	Reliability
c17	NR	2	5.461	0.165	0.9521
	OAV/TAV/TMR	3	5.337	0.176	0.9760
	CTMR	6	5.171	0.176	0.9796
c3540	NR	90	15.137	2.499	0.8796
	OAV/TAV/TMR	253	17.283	3.115	0.9323
	CTMR	447	22.135	3.415	0.9582

Table 5.1: Design space exploration results for the benchmarks c17 and c3540 from the ISCAS'85 benchmark suite

designs. The difference in dynamic power consumption is also less pronounced for the small benchmark. The reliability of these circuits improved moderately from non-redundant to redundant versions though for higher redundancy scales, even beyond CTMR, the increase in reliability gets lower due to saturation effect close to maximum reliability of 100%. Note that the Pareto filtering is not required for these benchmarks as Table 5.1 already lists the minimum set of non-dominated points.

5.1.3 s713 and s838 (ISCAS'89)

This series of HDL benchmark designs consists of combinational as well as sequential elements due to which the voter placement algorithms result in large variation of the performance factors. Table 5.2 lists the results for the two benchmarks and highlights the non-dominated, i.e., Pareto-optimal, implementations of the HDL design. Those implementation points having similar parameter values are highlighted only once. The benefit of Pareto-filtering is obvious as the 32-point set is reduced to 12 and 7 points for s713 and s838, respectively. The NR version is not compared since it will always be a Pareto-optimal point. It can be observed from the table that the span of parameters for OAV, TAV and TV is not high since they only differ in number of voters as compared to NR and CTMR which vary in number of modules as well. Moreover, we can observe that different voter-insertion algorithms can greatly vary the trade-off points.
While one would assume that the area utilization always increases in ascending order from NR to CTMR, the experiments prove this assumption wrong. As can be observed for the HFC algorithm and the s713 benchmark, the configuration TV consumes less slices than the configuration TAV, albeit the number of voters for TV is higher than for TAV. The explanation for such anomalies lies again in the automatic mapping, placement and routing tools. Sometimes, resources such as flip-flops remain unused in slices to balance the timing constraints and, more generally, the optimization possibilities vary from one design to another. Furthermore, the choice of voter-insertion algorithm within each configuration has a high impact on the area consumption, e.g., for the s838 benchmark the slice utilization varies from 113 to 124 (9.7% variation) for OAV configurations in contrast to 133 to 276 slices (107.5% variation) for CTMR configurations. Similarly, due to different optimization possibilities, the maximum clock frequency also does not always decrease with configurations using higher degrees of replication. For example, the

			CC	AFC	BFC	BD	HFC	HFFC	HFFIC	HFFOC
s713	NR	# Slices	42							
		Max Freq (MHz)	336							
		Dynamic PD (W)	1.476							
		Reliability	0.9568							
	OAV	# Slices	93	78	86	93	81	85	85	85
		Max Freq (MHz)	233	262	249	233	263	273	255	273
		Dynamic PD (W)	1.997	1.988	1.986	1.997	1.993	1.993	1.988	1.993
		Reliability	0.9767	0.9763	0.9763	0.9767	0.9765	0.9763	0.9763	0.9763
	TAV	# Slices	101	97	97	101	98	106	93	106
		Max Freq (MHz)	205	265	218	205	220	231	246	231
		Dynamic PD (W)	1.993	1.993	2.006	1.993	1.990	1.999	1.986	1.999
		Reliability	0.9772	0.9767	0.9767	0.9772	0.9769	0.9767	0.9766	0.9767
	TV	# Slices	102	117	94	102	86	117	92	117
		Max Freq (MHz)	227	207	222	227	240	207	233	207
		Dynamic PD (W)	2.002	1.996	2.001	2.002	1.988	1.996	1.998	1.996
		Reliability	0.9774	0.9769	0.9767	0.9774	0.9771	0.9769	0.9766	0.9769
	CTMR	# Slices	211	214	223	211	173	214	223	214
		Max Freq (MHz)	219	232	228	219	219	232	228	232
		Dynamic PD (W)	2.083	2.073	2.079	2.083	2.050	2.073	2.079	2.073
		Reliability	0.9790	0.9789	0.9789	0.9790	0.9789	0.9789	0.9789	0.9789
s838	NR	# Slices	33							
		Max Freq (MHz)	260							
		Dynamic PD (W)	0.287							
		Reliability	0.9384							
	OAV	# Slices	113	117	124	113	117	117	124	117
		Max Freq (MHz)	222	205	228	214	205	205	228	205
		Dynamic PD (W)	0.317	0.305	0.316	0.317	0.305	0.305	0.316	0.305
		Reliability	0.9735	0.9736	0.9732	0.9735	0.9736	0.9735	0.9732	0.9736
	TAV	# Slices	147	131	133	147	131	131	133	131
		Max Freq (MHz)	217	212	205	217	212	212	205	212
		Dynamic PD (W)	0.317	0.320	0.322	0.317	0.320	0.320	0.322	0.320
		Reliability	0.9746	0.9746	0.9740	0.9746	0.9746	0.9746	0.9740	0.9746
	TV	# Slices	166	173	140	166	173	173	140	173
		Max Freq (MHz)	203	227	208	203	227	227	208	227
		Dynamic PD (W)	0.328	0.327	0.325	0.328	0.327	0.327	0.325	0.327
		Reliability	0.9752	0.9752	0.9745	0.9752	0.9752	0.9752	0.9745	0.9752
	CTMR	# Slices	276	133	212	276	133	133	212	133
		Max Freq (MHz)	203	219	203	203	219	219	203	219
		Dynamic PD (W)	0.336	0.306	0.321	0.336	0.306	0.306	0.321	0.306
		Reliability	0.9793	0.9793	0.9793	0.9793	0.9793	0.9793	0.9793	0.9793

Table 5.2: Design space exploration results for the benchmarks s713 and s838 from the ISCAS'89 benchmark suite

maximum clock frequency increased for the s713 benchmark design, for CC algorithm, when going from TAV to TV. However, switching from the non-redundant to redundant configurations drastically impacts the maximum clock frequency. For example, when going from NR to CTMR we observe a 45% decrease for s713. The maximum clock frequency also varies considerably with variation of voter-insertion algorithm, e.g., 29% variation for s713 and the TAV configuration. The dynamic power consumption varies minimally for these benchmarks, with a maximum variation of 9.8% observed for s838 and the CTMR configuration. Generally, the reliability always increase from NR to CTMR for a single voter-insertion algorithm though when comparison is made among different algorithms, it can be easily observed that lower redundancy configuration of one algorithm may be more reliable than higher redundancy version of another algorithm. For example, for s713 benchmark, the TV configuration with BFC algorithm achieves less reliability than TAV configuration with BD algorithm. The reliability of CTMR versions is always higher than other configurations even when using different voter-insertion algorithms. However their variation is very minimal using different algorithms. A very noticeable advantage we gained from this design space exploration is the implementation point obtained with CTMR and HFFOC algorithm for s838 benchmark where the performance parameters are very comparable to the triplicated configuration though having higher reliability. Hence, the placement and routing of the design based on different voter-insertion algorithms can make highly redundant designs cost-effective.

5.1.4 b8 and b12 (ISCAS'99)

This series of HDL benchmarks contains complex state machine designs with immense feedback loops. Table 5.3 shows a high variation in all the performance factors compared to the last two benchmark categories. The slice utilization varies by only 4.5 and 2.3 times for b8 and b12 benchmarks respectively from NR to CTMR. The difference in slice utilization from OAV to TV is however very small. Interestingly, for the b12 benchmark, the minimum number of slices consumed for TV is almost equal to OAV. Looking at the importance of the voter-insertion algorithm for this benchmark category, we observe a variation of the maximum clock frequency from 204.1 MHz to 280.3 MHz (37%) for b8 with OAV configuration. The dynamic power consumption for b8 does not have any unexpected variations due to the small size of this benchmark, however, it varies by 35% for b12 from NR to CTMR. Moreover, varying the voter-insertion algorithm for the CTMR configuration of b12 can vary the power dissipation by 62% as well. The reliability varies approximately by 6% and 8% from NR to CTMR for b8 and b12 respectively. Pareto-filtering reduces the number of reasonable designs from 32 to 10 and 11 for b8 and b12, respectively, which covers nearly all the available voter-insertion algorithms in one or more redundancy configurations. As seen for the previous benchmarks, this category shows again how the optimization of resources increase from small to large circuit designs. Additionally, for the b12 benchmark, we observed that none of the implementations generated by any voter-insertion algorithm was repeated by another algorithm in contrast to our experience with previous benchmarks. This is due to the architecture of the circuits

			CC	AFC	BFC	BD	HFC	HFFC	HFFIC	HFFOC
b8	NR	# Slices	7							
		Max Freq (MHz)	290							
		Dynamic PD (W)	0.006							
		Reliability	0.9131							
	OAV	# Slices	35	32	35	35	32	32	35	32
		Max Freq (MHz)	204	280	252	204	280	280	252	280
		Dynamic PD (W)	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008
		Reliability	0.9647	0.9607	0.9649	0.9647	0.9607	0.9607	0.9649	0.9607
	TAV	# Slices	38	30	35	38	30	30	35	30
		Max Freq (MHz)	230	231	215	230	231	231	215	231
		Dynamic PD (W)	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008
		Reliability	0.9665	0.9615	0.9676	0.9673	0.9615	0.9615	0.9676	0.9615
	TV	# Slices	35	37	38	35	37	37	38	37
		Max Freq (MHz)	221	253	232	221	253	253	232	253
		Dynamic PD (W)	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009
		Reliability	0.9688	0.9629	0.9689	0.9688	0.9629	0.9629	0.9689	0.9629
	CTMR	# Slices	75	32	88	75	32	32	88	32
		Max Freq (MHz)	220	209	208	220	209	209	208	209
		Dynamic PD (W)	0.011	0.008	0.014	0.011	0.008	0.008	0.014	0.008
		Reliability	0.9702	0.9629	0.9702	0.9702	0.9629	0.9629	0.9702	0.9629
b12	NR	# Slices	173							
		Max Freq (MHz)	230							
		Dynamic PD (W)	0.058							
		Reliability	0.8840							
	OAV	# Slices	512	535	596	532	531	454	505	514
		Max Freq (MHz)	204	204	208	203	204	214	201	205
		Dynamic PD (W)	0.152	0.137	0.156	0.148	0.149	0.133	0.140	0.149
		Reliability	0.9620	0.9574	0.9615	0.9606	0.9487	0.9497	0.9604	0.9590
	TAV	# Slices	602	439	492	547	551	565	545	490
		Max Freq (MHz)	204	208	204	201	205	207	207	206
		Dynamic PD (W)	0.163	0.123	0.141	0.152	0.144	0.142	0.156	0.132
		Reliability	0.9685	0.9669	0.9733	0.9671	0.9526	0.9532	0.9725	0.9689
	TV	# Slices	681	517	477	453	568	526	546	497
		Max Freq (MHz)	188	202	210	203	201	203	203	205
		Dynamic PD (W)	0.172	0.132	0.150	0.149	0.155	0.140	0.163	0.135
		Reliability	0.9748	0.9752	0.9782	0.9738	0.9562	0.9752	0.9777	0.9751
	CTMR	# Slices	736	730	786	564	592	632	690	578
		Max Freq (MHz)	174	184	164	170	200	169	166	193
		Dynamic PD (W)	0.229	0.196	0.272	0.222	0.180	0.189	0.292	0.189
		Reliability	0.9752	0.9752	0.9798	0.9752	0.9752	0.9752	0.9798	0.9752

Table 5.3: Design space exploration results for the benchmarks b8 and b12 from the ISCAS'99 benchmark suite

which respond differently to the replication and voter-insertion decisions of the algorithms.

5.2 Validating Runtime Tool Flow

The Pareto-filtered implementations generated from the design-time tool flow need to be utilized and reconfigured at run-time using the run-time tool flow discussed in Section 4.2.3. For experimentation, instead of a 32 point implementation set, we use only 3-point set using configurations NR, TMR and CTMR while using voter algorithm CC. The reason for using this small set is to analyze and demonstrate the reconfiguration process effectively instead of presenting numerous reconfigurations required in a larger implementation set. Moreover, the process of reconfiguration holds similar for all sizes of implementation sets. Therefore, to analyze the trade-off between reliability and performance given by different implementation variants of a design, we use two case studies of hardware designs namely *data sorter* and *matrix multiplier*. Though the design-time analysis of these hardware designs would be similar to benchmarks used in the previous section; it is skipped here since we use only 3 implementation points for demonstration purposes instead of using a fully pareto-optimized implementation set.
The DRM implementation on a reconfigurable SoC platform is faced with the challenge of area efficiency. The problem we observed before implementing the DRM concept was that the area of a reconfigurable slot has to be limited to a fixed size, which refers to the maximum area of a redundant implementation we use in the reconfigurable partition, i.e., CTMR. Hence, when the reconfigurable partition is utilized with relatively smaller redundant structures like TMR, we do not obtain an area efficiency since the unused portion of the partition can not be utilized for other logic. Therefore, we use the concept of parallelism to fully utilize the reconfigurable partition for each of the redundant implementations smaller than CTMR, i.e., the NR and TMR structures will be implemented as parallelized versions. Hence, the reconfigurable slot is fully utilized while providing a higher throughput due to parallel application engines.

5.2.1 Data Sorter

The data sorting hardware thread uses bubble sorting algorithm to sort data in an ascending order. The sorting thread operates on 8KB blocks of 32 bit integer data and its performance is measured by the sorting rate, in blocks/minute. The sorting application has been chosen due to its block-based processing usage which is considered typical for signal/multi-media processing, data compression and encryption tasks [85, 86].
In our experiments, we allocate a certain hardware area for implementing the sorting function. The area is chosen such to fit the sorter implementation with the highest level of reliability, i.e., CTMR. The same area can also be used for an internally parallel instance of the hardware sorter in the TMR version, and for a sorter in the NR version that employs eight parallel sorting engines. We have developed these parallel versions of a data sorter for TMR and NR, as well as the CTMR version as ReconOS hardware threads that are designed to operate at 100 MHz. We denote the resulting reliability versions as CTMR,

Application Configuration	Hardware Slot #1	Hardware Slot #2	Hardware Slot #3
Static-Maximum-Reliability	CTMR	CTMR	CTMR
Static-Varying-Reliability	CTMR	TMR*2	NR*8
Reconfigurable	CTMR | TMR*2 | NR*8	CTMR | TMR*2 | NR*8	CTMR | TMR*2 | NR*8

Table 5.4: Hardware slot combinations

TMR*2 and NR*8. Since we keep the hardware area constant for all sorter implementations, a change in the reliability requirement and the subsequent reconfiguration will result in a change of the sorting performance.
We experiment with a ReconOS system as shown in Figure 4.3 employing three hardware slots and compare three different configurations for the data sorter application, respectively, for utilizing them. The configurations are denoted as static-maximum-reliability, static-varying-reliability and reconfigurable. In the static-maximum-reliability configuration, we strive for maximum reliability and employ the CTMR sorter in each of the three slots in a static way, i.e., without partial reconfiguration. The so-called static-varying-reliability configuration is also static but uses all three redundancy versions of the sorter, i.e., NR*8, TMR*2 and CTMR at the same time, each one in separate hardware slot. During runtime, we can switch the hardware threads on and off based on the reliability requirements. Finally, in the reconfigurable configuration, we utilize partial reconfiguration to reconfigure the hardware slots with sorter threads matching the reliability requirements. That is, at a particular instant all the three threads are configured either NR*8 or TMR*2 or CTMR. The organization of these configurations is illustrated in Table 5.4.
The entire ReconOS base system was designed using Xilinx EDK. The run-time reconfiguration and performance measurements were performed via software developed with Xilinx SDK and downloaded to a Microblaze processor implemented on FPGA. The partial bitstreams for the dynamically reconfigurable regions, i.e., the hardware slots, are generated by the Xilinx Partial Reconfiguration toolflow [87]. There are total of nine partial bitstreams representing NR*8, TMR*2 and CTMR versions for each of the three hardware slots. The full bitstream is always generated with the NR versions of the hardware threads.
The decision mechanism we envision for our experiments is radiation-based, utilizing the radiation profile obtained from the Borealis flight (duration 103 minutes) shown in Section 2.5.1[1]. Since we have three reliability versions for the data sorter, i.e., NR, TMR and CTMR, we divide the radiation levels into the following three ranges corresponding to three reliability levels.

- Reliability level 1: 0-300 counts/min
- Reliability level 2: 300-600 counts/min
- Reliability level 3: 600-900 counts/min

[1]Though the radiation strike-rate in Figure 2.9 is shown with respect to altitude, the time-dependent variation, as used in this experimentation, has a similar trend.

The radiation data is stored on the Microblaze and each radiation sample is read and interpreted after a time interval of one minute, according to the data frequency of the Borealis flight. The NR*8, TMR*2 and CTMR implementations, as partial bitstreams, are loaded into the SDRAM associated with the Microblaze and used for reconfiguration via the ICAP interface of the FPGA. The evaluation platform used is Xilinx ML605 Board, which is equipped with a Virtex 6 XC6VLX240T FPGA. The data to be sorted is continuously provided to the hardware threads until the completion of experiment or Borealis flight duration while each block of data is comprised of 2048 32-bit words.
The sorting performance for the three configurations can be compared in Figure 5.1. For reliability level 1, i.e., 0-44 minutes, we can observe that the sorting rate of reconfigurable configuration is more than double and around 8 times higher than static-varying-reliability and static-maximum-reliability configurations respectively. From 44-46 minutes, the sudden increase in radiation rate, as recorded by the sensor, makes the three threads reconfigure to TMR*2 equivalents in the reconfigurable configuration. Similarly, for the static-varying-reliability configuration, the NR*8 thread is switched off while dropping the sorting rate by 3.7 times since the total sorting units decreased by the same magnitude as well. The similar transition occur for 46-50 min range corresponding to reliability level 1. For 50-60 min range, the reliability level shifts to the second category. In this region, the total working units for the reconfigurable configuration are 6 compared to 3 units for each of static-varying-reliability and static-maximum-reliability configurations. However, the sorting rate of static-varying-reliability configuration is slightly lower than static-maximum-reliability configuration due to extra complexity of software code used for continuous radiation monitoring in static-varying-reliability case. Similarly, for the last reliability level, with range 60-103 min, has 3 sorting units working in reconfigurable and static-maximum-reliability configurations while only a single unit is utilized for the static-varying-reliability case. The slight difference for the sorting rate for reconfigurable case compared to static-maximum-reliability one is due to the same code-complexity reason described above. However, the static-varying-reliability configuration is ineffective even compared to static-maximum-reliability configuration in highest reliability requirement.
Overall, the reconfigurable configuration, upon which the DRM technique is based, outperforms both of the other configurations. In lowest reliability requirements, the performance is double and seven and a half times higher compared to static-varying-reliability and static-maximum-reliability configurations respectively. For highest reliability requirement, reconfigurable configuration has comparable performance to static-maximum-reliability and 2.8 times higher than static-varying-reliability configuration. The static-varying-reliability configuration, on the other hand, is 3.6 times faster and 3 times slower than the static-maximum-reliability configuration in highest and lowest reliability requirements respectively. The data throughput of the 3 configurations can also be compared by the total data sorted at the end of the experiment, i.e., 7.43e5, 3.40e5 and 1.76e5 blocks for reconfigurable, static-varying-reliability and static-maximum-reliability configurations respectively. The time spent during each reconfiguration stage is 228ms (76ms for each thread). Since our radiation sampling rate is one minute, the reconfiguration time is negligible and hence, not suitable to be represented on the graph, having time scale in

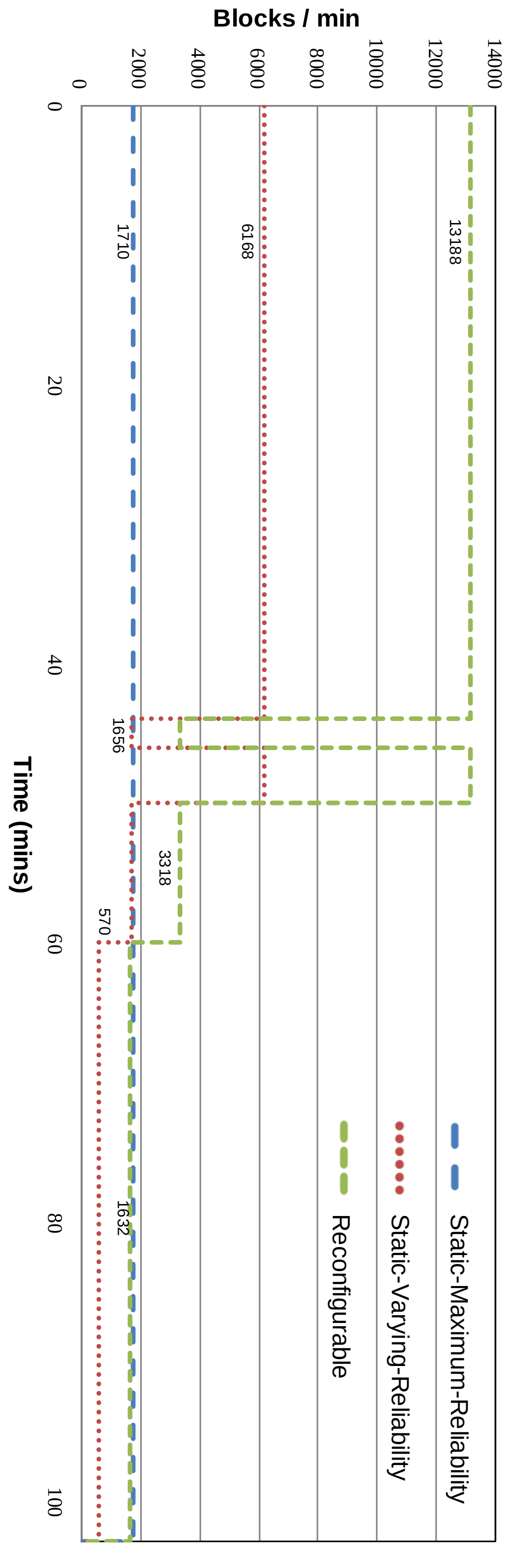


Figure 5.1: Performance results for static-maximum-reliability, static-varying-reliability and reconfigurable configurations of data sorter

minutes. Moreover, the reconfiguration time can be further decreased by using different FPGA architectures [88] or utilizing processor-independent partial reconfiguration [89].

5.2.2 Matrix Multiplier

Matrix multiplication is a heart of many image processing algorithms [90, 91]. The matrix multiplication application we use as our case study multiplies integer matrices of 128 columns and 128 rows. It works by reading in Matrix B completely and Matrix A row-wise, thereby performing a row-wise multiplication and then writing back the result row-wise until the whole matrix multpilication is completed. The hardware thread is part of an application that uses the Strassen algorithm [92] to split the multiplication of a 512 columns by 512 rows matrix into 49 multiplications of smaller (128 x 128) matrices. This way, the matrix multiplication is parallelizable and the workload can be distributed among many hardware threads.
The experimentation platform for this application, including hardware generation and experimental duration is the same as for the data sorter. However, due to different size of this hardware application, the parallelization is different, i.e., the CTMR version of matrix multiplier accommodates two instances of TMR and five instances of NR, denoted as CTMR, TMR*2 and NR*5 respectively, in contrast to eight NR versions for the data sorter application. Moreover, the reconfiguration time for each slot is double compared to data sorter due to double the size of hardware utilized, i.e., two clock regions compared to one for data sorter.
The performance results of the matrix multiplier are shown in Figure 5.2. For reliability level 1, i.e., 0-44 minutes, we can observe that the multiplication rate of reconfigurable configuration is 1.7 and 3.7 times higher than static-varying-reliability and static-maximum-reliability configurations respectively whereas the number of working units are higher by a factor of almost double and five times respectively. The performance does not exactly scale with the number of working units and depends on the architecture and how efficiently a hardware design is parallelized. Overall, the reconfigurable configuration again outperforms both of the other configurations. In lowest reliability requirements, as mentioned before, the performance is 1.7 and 3.7 times higher compared to static-varying-reliability and static-maximum-reliability configurations. For highest reliability requirement, reconfigurable configuration has comparable performance to static-maximum-reliability and 2.86 times higher than static-varying-reliability configuration. The static-varying-reliability configuration, on the other hand, is 2.2 times faster and 2.86 times slower than the static-maximum-reliability configuration in lowest and highest reliability requirements respectively. The data throughput of the 3 configurations can also be compared by the total matrices multiplied at the end of the experiment, i.e., 3.85e5, 2.11e5 and 1.64e5 matrices for reconfigurable, static-varying-reliability and static-maximum-reliability configurations respectively.

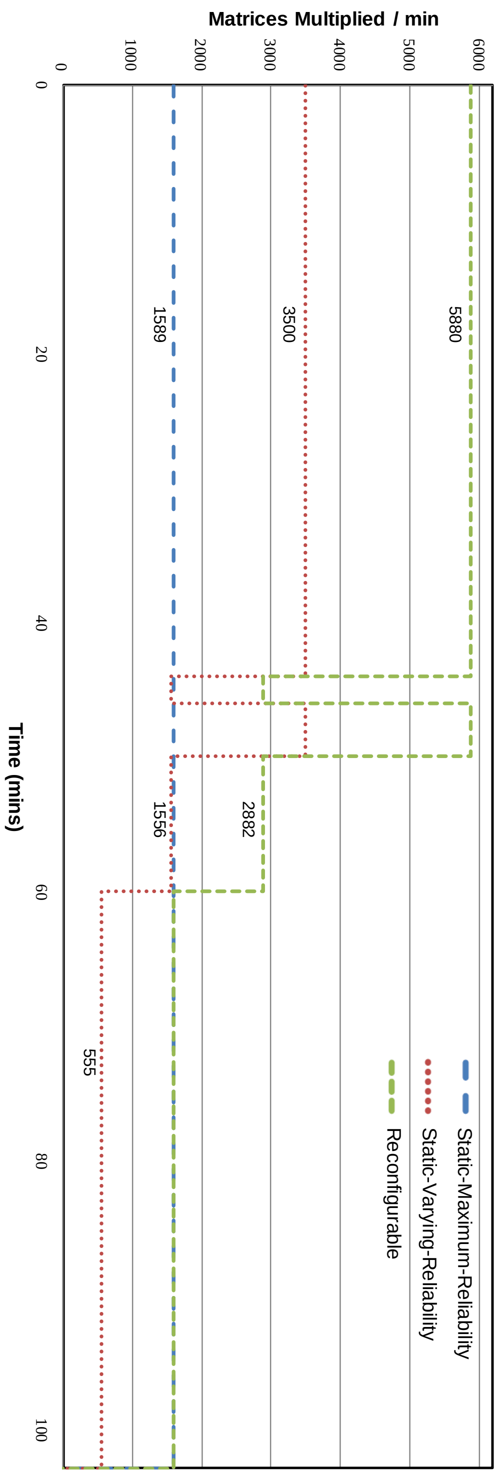


Figure 5.2: Performance results for static-maximum-reliability, static-varying-reliability and reconfigurable configurations of matrix multiplier

5.3 Variation of Pareto-optimal Implementations of DRM Tool Flow

In this section, we investigate how the DRM Pareto-optimal implementations differ with the exact reliability values as well as with different error-probability values.

5.3.1 Using Relative versus Exact Reliability Values

In chapter 3, we introduced our BDEC reliability evaluation tool that could provide exact reliability magnitudes of redundant implementations based on the input error probability values. However, before the availability of this tool, the Pareto-optimization was performed with relative reliability values in our previous work [5]. In this section, we compare the Pareto-optimal points based on the exact and relative reliability analysis. Therefore, we have used the benchmark s713 to check how the Pareto-optimal points differ in the two scenarios. The results of the design space exploration of s713 benchmark are shown in Table 5.5. The highlighted points in the table refer to non-dominated, i.e., Pareto-optimal, redundant implementations of the circuit. The benefit of using the automated BDEC tool is evident that our exact reliability values filtered 12 Pareto-optimal implementations in contrast to 10 relative points (excluding NR version which is always a Pareto-optimal point). Hence, the error due to relative reliability assumption is removed which assumes that the reliability is independent of the voter insertion algorithm. Based on this experimentation, we can conclude that the relative reliability assumption should not be used since it can result in arbitrarily different number of Pareto-optimization results.

5.3.2 Using Different Error Probability Values

To investigate the variation of Pareto-optimal implementations with respect to error-probability, we used eight different values for error-probability of gate/component, input and voter consecutively for the s713 benchmark. The lower range of the error probability has been taken up to 1e-9 keeping in mind the practical failure rates which have small magnitudes particularly for space applications [2, 3, 46, 63]. However, for the low error probability magnitudes, the output reliability needs to be computed up to large number of decimal places to find the correct Pareto optimal implementations. The results are shown in Table 5.6. The results of area, latency and power are removed in this table to avoid repetition of similar data. The trend of resulting Pareto optimal points can be split into four ranges of error probability with respect to the threshold point. Remind that the threshold point refers to the value of error-probability after which the higher redundancy results in decrease in output reliability (see Section 3.3). The first range corresponds to the error-probability less than minus 1% error around the threshold point. The two middle ranges of error-probability approximately lie for plus and minus 1% error around the threshold point respectively. Finally, the fourth region corresponds to more than plus 1% error above the threshold point. The threshold point, in this example, lies between 10% and 11% error probability and appears earlier compared to parameter variability results

Error Probability	Redundancy Configurations	Performance Parameters	CC	AFC	BFC	BD	HFC	HFFC	HFFIC	HFFOC
relative	NR	# Slices	42							
		Max Freq (MHz)	336							
		Dynamic PD (W)	1.476							
		Reliability	1							
	OAV	# Slices	93	78	86	93	81	85	85	85
		Max Freq (MHz)	233	262	249	233	263	273	255	273
		Dynamic PD (W)	1.997	1.988	1.986	1.997	1.993	1.993	1.988	1.993
		Reliability	2	2	2	2	2	2	2	2
	TAV	# Slices	101	97	97	101	98	106	93	106
		Max Freq (MHz)	205	265	218	205	220	231	246	231
		Dynamic PD (W)	1.993	1.993	2.006	1.993	1.990	1.999	1.986	1.999
		Reliability	3	3	3	3	3	3	3	3
	TV	# Slices	102	117	94	102	86	117	92	117
		Max Freq (MHz)	227	207	222	227	240	207	233	207
		Dynamic PD (W)	2.002	1.996	2.001	2.002	1.988	1.996	1.998	1.996
		Reliability	4	4	4	4	4	4	4	4
	CTMR	# Slices	211	214	223	211	173	214	223	214
		Max Freq (MHz)	219	232	228	219	219	232	228	232
		Dynamic PD (W)	2.083	2.073	2.079	2.083	2.050	2.073	2.079	2.073
		Reliability	5	5	5	5	5	5	5	5
0.01	NR	# Slices	42							
		Max Freq (MHz)	336							
		Dynamic PD (W)	1.476							
		Reliability	0.95691							
	OAV	# Slices	93	78	86	93	81	85	85	85
		Max Freq (MHz)	233	262	249	233	263	273	255	273
		Dynamic PD (W)	1.997	1.988	1.986	1.997	1.993	1.993	1.988	1.993
		Reliability	0.97661	0.97629	0.97617	0.97661	0.97637	0.97625	0.97615	0.97625
	TAV	# Slices	101	97	97	101	98	106	93	106
		Max Freq (MHz)	205	265	218	205	220	231	246	231
		Dynamic PD (W)	1.993	1.993	2.006	1.993	1.990	1.999	1.986	1.999
		Reliability	0.97712	0.97664	0.97643	0.97712	0.97676	0.97657	0.97639	0.97657
	TV	# Slices	102	117	94	102	86	117	92	117
		Max Freq (MHz)	227	207	222	227	240	207	233	207
		Dynamic PD (W)	2.002	1.996	2.001	2.002	1.988	1.996	1.998	1.996
		Reliability	0.97744	0.97692	0.97666	0.97744	0.97710	0.97684	0.97660	0.97684
	CTMR	# Slices	211	214	223	211	173	214	223	214
		Max Freq (MHz)	219	232	228	219	219	232	228	232
		Dynamic PD (W)	2.083	2.073	2.079	2.083	2.050	2.073	2.079	2.073
		Reliability	0.97895	0.97890	0.97889	0.97895	0.97892	0.97890	0.97889	0.97890

Table 5.5: Pareto Optimization of s713 Benchmark with respect to relative and exact error probabilities

in Section 3.3 because error probability of gates, components and voters are increasing simultaneously. The error-probability regions along with their respective reliability relations are shown in Table 5.7.

As can be seen from Table 5.6, the Pareto-optimal points stay the same for error probability scaling less than 1% from the threshold point. Reliability relation number 1 holds in this case with respect to any particular voter-insertion algorithm. R refers to respective reliability of each of the redundant configurations. For the region of up to minus 1% error

Error Probability	Redundancy Configurations	CC	AFC	BFC	BD	HFC	HFFC	HFFIC	HFFOC
1e-9	NR	0.9999999954030032478							
	OAV	0.9999999979130434752	0.9999999979130434705	0.9999999979130434688	0.9999999979130434752	0.9999999979130434716	0.9999999979130434699	0.9999999979130434685	0.9999999979130434699
	TAV	0.9999999979130434822	0.9999999979130434758	0.9999999979130434728	0.9999999979130434822	0.9999999979130434774	0.9999999979130434748	0.9999999979130434722	0.9999999979130434748
	TV	0.9999999979130434862	0.9999999979130434801	0.9999999979130434764	0.9999999979130434862	0.9999999979130434819	0.9999999979130434800	0.9999999979130434755	0.9999999979130434800
	CTMR	0.9999999979130435037	0.9999999979130435037	0.9999999979130435037	0.9999999979130435037	0.9999999979130435037	0.9999999979130435037	0.9999999979130435037	0.9999999979130435037
1e-6	NR	0.9999954033034							
	OAV	0.9999979130146	0.9999979130099	0.9999979130082	0.9999979130146	0.9999979130110	0.9999979130093	0.9999979130079	0.9999979130093
	TAV	0.9999979130216	0.9999979130152	0.9999979130122	0.9999979130216	0.9999979130168	0.9999979130142	0.9999979130116	0.9999979130142
	TV	0.9999979130256	0.9999979130195	0.9999979130158	0.9999979130256	0.9999979130213	0.9999979130184	0.9999979130149	0.9999979130184
	CTMR	0.9999979130431	0.9999979130431	0.9999979130431	0.9999979130431	0.9999979130431	0.9999979130431	0.9999979130431	0.9999979130431
1e-4	NR	0.999540608758932							
	OAV	0.9997910161	0.9997909692	0.9997909523	0.9997910161	0.9997909803	0.9997909629	0.9997909493	0.9997909629
	TAV	0.9997910862	0.9997910217	0.9997909921	0.9997910862	0.9997910376	0.9997910119	0.9997909862	0.9997910119
	TV	0.9997911261	0.9997910650	0.9997910274	0.9997911261	0.9997910832	0.9997910535	0.9997910190	0.9997910535
	CTMR	0.9997913003	0.9997913002	0.9997913002	0.9997913003	0.9997913003	0.9997913002	0.9997913002	0.9997913002
0.01	NR	0.95691							
	OAV	0.97661	0.97629	0.97617	0.97661	0.97637	0.97625	0.97615	0.97625
	TAV	0.97712	0.97664	0.97643	0.97712	0.97676	0.97657	0.97639	0.97657
	TV	0.97744	0.97692	0.97666	0.97744	0.97710	0.97684	0.97660	0.97684
	CTMR	0.97895	0.97890	0.97889	0.97895	0.97892	0.97890	0.97889	0.97890
0.05	NR	0.8273							
	OAV	0.8604	0.8589	0.8581	0.8604	0.8592	0.8587	0.8580	0.8587
	TAV	0.8632	0.8606	0.8591	0.8632	0.8613	0.8600	0.8589	0.8600
	TV	0.8652	0.8612	0.8600	0.8652	0.8631	0.8608	0.8598	0.8608
	CTMR	0.8810	0.8788	0.8782	0.8810	0.8795	0.8788	0.8782	0.8788
0.1	NR	0.7226							
	OAV	0.7259	0.7256	0.7250	0.7259	0.7256	0.7254	0.7250	0.7254
	TAV	0.7265	0.7262	0.7251	0.7265	0.7264	0.7257	0.7251	0.7257
	TV	0.7262	0.7252	0.7252	0.7262	0.7270	0.7251	0.7252	0.7254
	CTMR	0.7318	0.7310	0.7298	0.7318	0.7312	0.7310	0.7298	0.7310
0.11	NR	0.7064							
	OAV	0.7031	0.7032	0.7026	0.7031	0.7031	0.7030	0.7026	0.7030
	TAV	0.7033	0.7035	0.7026	0.7033	0.7036	0.7030	0.7026	0.7030
	TV	0.7025	0.7022	0.7026	0.7025	0.7039	0.7023	0.7025	0.7023
	CTMR	0.7027	0.7028	0.7017	0.7027	0.7028	0.7028	0.7017	0.7028
0.2	NR	0.6026							
	OAV	0.5566	0.5576	0.5577	0.5566	0.5574	0.5576	0.5578	0.5576
	TAV	0.5549	0.5570	0.5573	0.5549	0.5567	0.5570	0.5574	0.5570
	TV	0.5528	0.5557	0.5569	0.5528	0.5559	0.5560	0.5571	0.5560
	CTMR	0.5078	0.5114	0.5114	0.5078	0.5105	0.5114	0.5114	0.5114

Table 5.6: Pareto optimal implementations of s713 benchmark with varying error probabilities

Error Probability Region	Reliability Relations	Reliability Relation Number
<Threshold point -1%	$R_{NR} < R_{OAV} < R_{TAV} < R_{TV} < R_{CTMR}$	1
Threshold point -1% - Threshold Point	$R_{NR} < R_{OAV\|\|TAV\|\|TV} < R_{CTMR}$	2
Threshold Point - Threshold point +1%	$R_{NR} > R_{OAV\|\|TAV} > R_{CTMR}$	3
>Threshold point +1%	$R_{NR} > R_{OAV} > R_{TAV} > R_{TV} > R_{CTMR}$	4

Table 5.7: Error probability regions with their corresponding equations for s713 benchmark

probability from threshold point, the reliability and hence Pareto optimal points for OAV, TAV and TV vary arbitrarily, not following the expected relation ($R_{OAV} < R_{TAV} < R_{TV}$), however still holding reliability relation number 2. For error probability up to 1% above threshold point, the Pareto optimal point is only one, i.e., NR version being superior in reliability than any of the redundant versions. The expected relation ($R_{OAV} > R_{TAV} > R_{TV}$) does not hold in this case, however, it is represented by reliability relation number 3. Finally, for error probability 1% above threshold point, non-redundant version, NR, remains to be the only Pareto optimal point where the expected reliability relation number 4 holds.
It can be concluded for this benchmark that beyond the small transition region around the threshold point, the Pareto optimal implementation points do not differ with variation of error probability. However, to provide a firm statement on whether this trend holds for any test circuit, we have to perform this analysis for various benchmarks which corresponds to our future work, due to extensive time required for analysis of a single benchmark.

5.4 Conclusion

This chapter verifies the tool flow of the DRM technique presented in the previous chapter. The design-time and run-time tool flows of DRM have been verified with standard ISCAS benchmarks which leads us to certain conclusions. The design-time tool flow in particular, shows that the performance factors of a redundant design, i.e., area, latency and power depend strongly on the placement and routing decisions taken by the FPGA synthesis tool. While focusing on speed and/or area efficiency, the routed FPGA design leads to different performance parameters as the expected results. In particular, we have seen that the optimization potential is high for large HDL designs thereby routing the big circuits more efficiently on FPGAs. For example, a slight increase in area consumption might lead to dramatic decrease in power consumption and/or latency. Similarly, an increase in the reliability level could even improve the performance factors. Moreover, our experimental observation shows that the trend in variation of performance factors strongly depends on the circuit benchmark suite. The run time tool flow, though presented with a smaller implementation set than used for the design time, proves that the reconfigurable redundancy configuration, upon which our DRM tool flow is based, outperforms the static reliability schemes based on static-maximum-reliability and static-varying-reliability redundancy configurations. For our test sorting and matrix multiplication applications, we found DRM to be 7.5 and 3.7 times performance-efficient respectively, compared to static reliability management schemes.

CHAPTER 6

Summary and Conclusions

This chapter summarizes the contributions of this thesis and draws conclusions. It further discusses future research directions.

6.1 Summary

Radiation induced errors in FPGA hardware act as a bottleneck for reliable computing in aerospace missions. Conventionally, hardware redundancy is used as a standard method to mitigate these errors. However, the cost of this reliability approach is a fixed overhead in the performance factors of area consumption, latency and power dissipation. The fixed overhead corresponds to the worst case radiation scenario. Based on the radiation pattern studies of space orbits in literature, we observed a big variation in the radiation strength during the mission time span. The motivation behind this research is to utilize various redundant levels of a hardware design according to the reliability requirements, i.e., lower redundancy levels for low radiation environments and vice versa. As a result, the performance of a hardware design can be improved in low reliability requirements. This concept of adapting reliability levels to radiation strength is named Dynamic Reliability Management (DRM) in our research.
DRM is a system level concept that has to be recognized on a system-on-chip platform at both the hardware and software layers. It consists of a design-time and a run-time part. The design-time part involves a tool flow that takes as input a target application design written in a hardware description language like Verilog or VHDL. The tool flow generates several redundant implementations of this hardware design using BANL TMR tool. The original BANL tool was capable of generating only TMR configuration which we extended in this research to three more redundant configurations, and hence extended the design space to 32 redundant implementations of a hardware design. Due to the varying optimization potential of these hardware designs on an FPGA platform, we have to conduct the analysis of the performance factors, for each of the redundant implementations, using

standard Xilinx tools. However, the computation of reliability for each of these implementations offered a new challenge. The reliability computation approaches exist in literature with different application domains. However, we chose Boolean difference error calculus (BDEC) approach from literature that fulfills our reliability computation requirements. The original BDEC model is not straightforwardly usable for computing reliability of the redundant structures. Therefore, we have first extended this model in two ways, i.e., interpreting redundant structures as well as sequential circuits. The revised BDEC model is afterwards automated in MATLAB which made reliability calculations easier for a circuit of any size. The overall design-time tool flow, now being able to compute area, latency, power and achieved reliability, automatically analyzes the 32-point implementation set. The final step is the Pareto-filtration of the redundant implementations on the basis of three performance factors and exact reliability of each redundant design.
The second part of DRM, i.e., run-time tool flow is responsible for utilizing the Pareto-filtered implementations according to the reliability requirements. The decisions on when and which implementation to configure at a particular instant, is taken by the decision module. The decision module is the central unit, implemented in software or hardware, that is responsible for taking decisions when to reconfigure the application thread to higher or lower reliability levels based on the external, time, cooperative or radiation/error based data. The run-time tool flow is implemented on a system-on-chip platform using the embedded operating system ReconOS on a Xilinx ML605 evaluation board. ReconOS, which is an operating system for reconfigurable logic cores switches among various redundant implementations of the hardware design based on the decisions taken by the decision module. The tool flow consists of standard IP cores of Xilinx like Microblaze processor and DDR3 SDRAM, debugger modules and peripherals, etc. Our target application is implemented as a hardware thread on a reconfigurable partition. The challenge arises when we had to fix the partition size to the maximum size of redundant implementation in the Pareto optimal set. This problem was resolved by implementing the parallelized versions of the hardware threads with reliability levels lower than the maximum reliable configuration. In this way, the reconfigurable partition is fully utilized at all times while the parallelized hardware threads offers the performance benefit as well.
Each of the DRM parts have been validated separately on various benchmarks. The design-time tool flow has been validated for a set of six ISCAS benchmarks from three categories of combinational and sequential circuit architectures. The results, while presenting the area, latency, power and reliability magnitudes, also focus on the filtration of non-dominated Pareto-optimal circuit implementations. The run-time part has been validated by practical data sorting and matrix multiplication case studies. The run-time benchmarks are evaluated for performance benefits on a system-on-chip platform utilizing our DRM approach in contrast to static reliability techniques observing fixed overheads. The results show that our DRM technique, which is based on dynamic partial reconfiguration outperforms all the static reliability management techniques and helps us to maintain a suitable reliability-performance trade off at all times.

6.2 Conclusions and Lessons Learned

Dynamic Reliability Management is the technique for adaptive optimization of application reliability with the performance during its operation time span. While implementing this concept for FPGA based hardware designs, we have developed tool flows and performed thorough experimentation which leads us to the following conclusions.

- Our motivation behind proposing the concept of DRM is to avoid the usage of static redundant structures in reliability-required scenarios. Therefore, while performing research on the space missions, we have observed that the logic of utilizing fixed redundant structures in FPGAs results in performance loss when the system does not operate under worst radiation scenario. The reason is the varying nature of radiation patterns for different satellite orbits as observed from the radiation studies in literature. Therefore, utilizing the benefit of *reconfigurability* of FPGAs, we have concluded that dynamically varying the redundant structures on FPGAs corresponding to the radiation levels could be highly useful in maintaining a suitable trade-off between reliability and system performance at all times.

- The probabilistic computational reliability models, presented in the literature, have been found to be not taking into account the effect of redundancy. It has been observed that the redundant structures, when evaluated by these schemes provides us even lower reliability of redundant circuits while they are supposed to be superior in reliability compared to a non-redundant circuit. However, in this research, we utilized redundancy theory and merged it with one of the error probabilistic models to comprehend the redundancy effects in circuits. Therefore, the cumulative reliability of redundant circuits does not always decrease from primary inputs to outputs. Instead, there exist voter elements that are capable of improving circuit reliability at intermediate stages of the circuit.

- The probabilistic computational models including BDEC have been found to not report the results on sequential circuit analysis. In this research, we have merged our utilized model, i.e., BDEC with the loop breaking and time frame expansion technique to simulate the sequential circuits. Hence, the reliability analysis which was limited to only combinational circuits has been extended to sequential elements in this research.

- The development of reliability evaluation tool using BDEC leads us to a concept of a threshold point. Generally, it has been considered that the higher levels of redundancy always improve circuit reliability. However, our analysis tells us that this assumption is true up to a certain threshold values of component, input and voter error probabilities as well as the signal probability. Going beyond the threshold point renders the redundancy concept useless where higher orders of redundancy result in even lesser circuit reliability. Such magnitudes of input, voter, component and signal probabilities are termed as threshold points with respect to each of these control parameters. Furthermore, we have concluded that the voter error probability is the most critical control parameter since the threshold point arrives earliest with increase in voter error probability.

- The experimentation conducted using DRM design-time tool flow has made us conclude that the redundant circuit configurations can highly vary among the performance factors like area, latency, power and reliability level, depending on the used granularity level and voter-insertion scheme. The high variation is influenced by the mapping and placement and routing decisions taken by the FPGA design softwares. While one would expect all the performance factors to be scaled linearly with higher orders of redundancy, the experiments have proved this assumption wrong. The conclusion we have drawn from the experiments is that the optimization potential of the redundant circuits increase with the higher orders of redundancy. Similarly, the placed and routed design can vary the critical path length thus varying latency. Finally, the organization of the FPGA components can greatly vary the expected power dissipation. Hence, a complete experimentation of all the redundant configurations should be exercised to obtain the realistic data on performance parameters which could make higher orders of redundancy cost-effective as was observed in our experiments.

- While implementing the DRM concept for run-time tool flow, we observed that DRM cannot be implemented straightforwardly on the FPGA platform. The reason is the constraint that the reconfigurable partition has to be fixed to a certain area as required by practical run-time systems such as ReconOS. This constrained area should correspond to the area required by the biggest redundant structure utilized. Constraining the reconfigurable partition does not serve the area efficiency with the DRM since the unused area cannot be utilized for other logic when lower levels of redundancy are used. Therefore, we introduced the concept of parallelism in the hardware threads. For lower levels of redundancy, the hardware threads are configured with the parallelized versions of application engines so that the area can be utilized fully and the performance can be improved as well. In contrast, dynamically allocating reconfigurable areas is possible by low-level FPGA bitstream manipulation though this direction is out of scope for our research work.

6.3 Future Directions

The research work in this thesis can be extended by making the BDEC reliability tool more efficient as well as by extending the DRM experimentation platform.

1. **Analysis using BDEC tool:** In future, we plan to extend the BDEC tool in five ways.
    - The signal probability values in BDEC model have been arbitrarily taken as 50% in this research as well as in the literature. However, in the future, we will develop an automated signal probability module that accurately calculates the

wire probability being at logic 1 during the observation period of the circuit. This module will be built on the analysis of value change dump file, i.e. .vcd of the Xilinx testbench analysis.

- Afterwards, we will perform the sequential circuit analysis for multiple iterations to observe whether the reliability at the outputs of sequential elements drops to a certain threshold or oscillate between bounds due to voter elements inserted after the sequential elements.
- The third improvement to this model will be to observe the effect of re-convergent fanout on the output reliability.
- Another extension will be providing a fault model that will identify the effect of individual wires on the output reliability and how to harden the sensitive ones using redundancy.
- The timing-efficiency of this tool will be improved by parallelizing the code and using multiple computing engines to run the BDEC algorithm.
- Lastly, we want to open a new research direction for modelling the gate error probability which has been used arbitrarily in the past by all the related research works. This fundamental research requires huge effort to model the major sources of errors at the device layer nowadays, e.g., NBTI (Negative-Bais Temperature Instability, HCI (Hot Carrier Injection) and threshold voltage variation.

2. **DRM Experimentation Platform:** During the course of validation of DRM, we developed an experimentation platform that could be extended in multiple ways. It is practically hard to enlarge the scope of our experimentation to cover all the possible aspects of system improvements in our experimentation. However, our future work will address some or all of these improvements as discussed in the following points.
    - We have, so far, focussed on studying the reliability-performance trade off for the application module only. However, the overall operating system and its interface should be accounted for similar reliability requirements. The simplest approach of ensuring the reliability of the base system is to implement it to the highest reliability/redundancy level, as also proposed in [2, 46], thereby adding a constant overhead in the performance of the system including application modules. Our future work will involve making the base system redundant including the processor Microblaze and then observe the performance benefit of DRM.
    - The performance of the system has been studied only while utilizing redundancy in this work whereas the reliable systems employ scrubbing as an additional reliability-enhancement technique. In the presence of scrubbing, the reliability of the system will be computed according to a more complex mathematical model [93]. Moreover, since scrubbing is performed via ICAP interface of the FPGA and so is the partial reconfiguration for our DRM approach, it may

cause performance degradation while the reconfiguration process is blocked by the scrubbing cycle. This effect will be investigated in our future work.

- The implementation of decision mechanism in DRM can be studied for different approaches based on error-rate or radiation level measurements. In our work, we have utilized a basic approach of dividing radiation data into different domains and taking the reconfiguration decision based on crossing the domain thresholds. However, the decision mechanisms can be made more accurate by real-time error-rate measurements while taking the reconfiguration decisions with more accuracy, using, for example a BRAM sensor [46]. Moreover, the performance of a decision mechanism can be improved by implementing it in hardware in contrast to the software approach. It is also worth mentioning that utilizing a non-redundant (NR) implementation for lowest reliability requirement is just for the sake of understanding the reconfiguration concept; it does not guarantee that the system will be reliable being non-redundant.
- The storage mechanism of partial bitstreams has to ensure the integrity of bitstream since we cannot afford the corruption of these bitstreams representing the whole functionality of hardware design. In our work, we stored them in DRAM while proposing ECC to protect them against corruption. The storage mechanism can be made more robust by additional reliability techniques which corresponds to our future work.
- In our work, the reliability of different redundant implementations were calculated offline using a known analytic reliability model. This approach can be made more compact by characterizing reliability at runtime using mathematical approaches. The decision algorithm can be made more compact which takes into account the parameters based on the hardware design, environmental conditions and system constraints.
- The real space applications involve extensive simulations considering space weather and solar conditions. In this work, we demonstrated the approach in a broader picture without stressing that our approximate reliability computation method is comparable to the real-time simulations for space environments. In our future work, we will utilize real aerospace environment and use CREME96 tool to observe the difference in performance of a DRM based application.

Acronyms

DRM	Dynamic Reliability Management
TMR	Triple Modular Redundancy
NR	No Redundancy
OAV	One Alternate Voter
TAV	Two Alternate Voters
TV	Triplicated Voter
CTMR	Cascaded Triple Modular Redundancy
ABFT	Algorithm based Fault Tolerance
DWC	Duplication With Comparison
ECC	Error Checking and Correcting Codes
LUT	Lookup Table
FPGA	Field Programmable Gate Array
ASIC	Application-specific Integrated Circuit
MUX	Multiplexer
RAM	Random Access Memory
CC	Connectivity Cutset
AFC	After Flipflop Cutset
BFC	Before Flipflop Cutset
BD	Basic Decomposition
HFC	Highest Fanout Cutset
HFFC	Highest Flipflop Fanout Cutset
HFFIC	Highest Flipflop Fanin Input Cutset
HFFOC	Highest Flipflop Fanin Output Cutset
HDL	Hardware Description Language
BDEC	Boolean Difference Error Calculator
PGM	Probabilistic Gate Model
PTM	Probabilistic Transfer Matrices Model
ADD	Algebraic Decision Diagram
PDD	Probabilistic Decision Diagram
BYU	Brigham Young University
LANL	Los Alamos National Laboratory
SoC	System-on-Chip
SAA	South Atlantic Anomaly
TID	Total Ionizing Dose

SEU	Single Event Upset
SET	Single Event Transient
SEFI	Single Event Functional Interrupt
MBU	Multiple Bit Upset
SEL	Single Event Latchup
SEB	Single Event Burnout
SEGR	Single Event Gate Rupture
CMOS	Complementary Metal Oxide Semiconductor
SOI	Silicon on Insulator
CLB	Configurable Logic Block
SRAM	Static Random Access Memory
DRAM	Dynamic Random Access Memory
BRAM	Block Random Access Memory
DSP	Digital Signal Processor
CRC	Cyclic Redundancy Check
SECDED	Single Error Correction Double Error Detection
FES	Feedback Edge Set
SCC	Strongly Connected Component
LEO	Low Earth Orbit
HEO	Highly Elliptical Orbit
TLE	Two-line Element
MTBF	Mean Time Between Failures
MTTF	Mean Time to Failure
FIT	Failure-in Time
R4R	Reconfiguration for Reliability
RFT	Reconfigurable Fault Tolerance
PRR	Partially Reconfigurable Region
HP	High Performance
PLB	Processor Local Bus
GUI	Graphical User Interface
FMU	Fault Management Unit
RCU	Reconfiguration Control Unit
BFD	BRAM Fault Detector
ReconOS	Reconfigurable Operating System
OSIF	Operating System Interface
MEMIF	Memory Interface
FIFO	First In First Out
NBTI	Negative Bias Temperature Instability
HCI	Hot Carrier Injection

APPENDIX A

Formulation of Error Probability Equations

In this appendix, we will show how to compute the error probability equations of 2-input OR and XOR gates as well as 3-input lookup table (LUT) with a reference initialization string. This formulation is based on the five step process described in Section 3.2.2.

A.1 2-Input OR Gate

Step 1:

$$f = x_1 + x_2 \tag{A.1}$$

Step 2:

$$\frac{\partial f}{\partial x_1} = \overline{x_2} \tag{A.2}$$

$$\frac{\partial f}{\partial x_2} = \overline{x_1} \tag{A.3}$$

$$\frac{\Delta f}{\Delta(x_1 x_2)} = \overline{x_1}\ \overline{x_2} + x_1 x_2 \tag{A.4}$$

Step 3:

$$Pr\{\frac{\partial f}{\partial x_1}\} = 1 - p_2 \tag{A.5}$$

$$Pr\{\frac{\partial f}{\partial x_2}\} = 1 - p_1 \tag{A.6}$$

$$Pr\{\frac{\Delta f}{\Delta(x_1 x_2)}\} = (1 - p_1)(1 - p_2) + p_1 p_2 = 1 - (p_1 + p_2) + 2p_1 p_2 \tag{A.7}$$

Step 4:

$$\varepsilon_{in} = \varepsilon_1(1 - \varepsilon_2)Pr\{\frac{\partial f}{\partial x_1}\} + \varepsilon_2(1 - \varepsilon_1)Pr\{\frac{\partial f}{\partial x_2}\} + \varepsilon_1\varepsilon_2 Pr\{\frac{\Delta f}{\Delta(x_1 x_2)}\} \tag{A.8}$$

$$\varepsilon_{in} = \varepsilon_1(1-\varepsilon_2)(1-p_2) + \varepsilon_2(1-\varepsilon_1)(1-p_1) + \varepsilon_1\varepsilon_2(1-(p_1+p_2)+2p_1p_2) \tag{A.9}$$

Step 5:

$$\varepsilon_{AND2} = \varepsilon_g + (1-2\varepsilon_g)\varepsilon_{in} \tag{A.10}$$

$$\varepsilon_{OR2} = \varepsilon_g + (1-2\varepsilon_g)(\varepsilon_1(1-p_2) + \varepsilon_2(1-p_1) + \varepsilon_1\varepsilon_2(1-2(p_1+p_2)+2p_1p_2)) \tag{A.11}$$

A.2 2-Input XOR Gate

Step 1:

$$f = x_1\overline{x_2} + \overline{x_1}x_2 \tag{A.12}$$

Step 2:

$$\frac{\partial f}{\partial x_1} = 1 \tag{A.13}$$

$$\frac{\partial f}{\partial x_2} = 1 \tag{A.14}$$

$$\frac{\Delta f}{\Delta(x_1x_2)} = 0 \tag{A.15}$$

Step 3:

$$Pr\{\frac{\partial f}{\partial x_1}\} = 1 \tag{A.16}$$

$$Pr\{\frac{\partial f}{\partial x_2}\} = 1 \tag{A.17}$$

$$Pr\{\frac{\Delta f}{\Delta(x_1x_2)}\} = 0 \tag{A.18}$$

Step 4:

$$\varepsilon_{in} = \varepsilon_1(1-\varepsilon_2)Pr\{\frac{\partial f}{\partial x_1}\} + \varepsilon_2(1-\varepsilon_1)Pr\{\frac{\partial f}{\partial x_2}\} + \varepsilon_1\varepsilon_2 Pr\{\frac{\Delta f}{\Delta(x_1x_2)}\} \tag{A.19}$$

$$\varepsilon_{in} = \varepsilon_1(1-\varepsilon_2) + \varepsilon_2(1-\varepsilon_1) \tag{A.20}$$

Step 5:

$$\varepsilon_{XOR2} = \varepsilon_g + (1-2\varepsilon_g)\varepsilon_{in} \tag{A.21}$$

$$\varepsilon_{XOR2} = \varepsilon_g + (1-2\varepsilon_g)(\varepsilon_1 + \varepsilon_2 - 2\varepsilon_1\varepsilon_2) \tag{A.22}$$

A.3 3-Input LUT

Reference initiation string: 00100000

Step 1:

The initialization string of LUTs are decoded based on the truth table analysis, where the least significant bit corresponds to $x_1x_2x_3$ and most significant $\overline{x_1}\ \overline{x_2}\ \overline{x_3}$.

$$f = x_1\overline{x_2}x_3 \tag{A.23}$$

Step 2:

$$\frac{\partial f}{\partial x_1} = \overline{x_2}x_3 \tag{A.24}$$

$$\frac{\partial f}{\partial x_2} = x_1x_3 \tag{A.25}$$

$$\frac{\partial f}{\partial x_3} = x_1\overline{x_2} \tag{A.26}$$

$$\frac{\Delta f}{\Delta(x_1x_2)} = x_3(x_1\overline{x_2} + \overline{x_1}x_2) \tag{A.27}$$

$$\frac{\Delta f}{\Delta(x_1x_3)} = \overline{x_2}(x_1x_3 + \overline{x_1x_3}) \tag{A.28}$$

$$\frac{\Delta f}{\Delta(x_2x_3)} = x_1(\overline{x_2}x_3 + x_2\overline{x_3}) \tag{A.29}$$

$$\frac{\Delta f}{\Delta(x_1x_2x_3)} = x_1\overline{x_2} + x_1x_3 + \overline{x_1}x_2x_3 \tag{A.30}$$

Step 3:

$$Pr\{\frac{\partial f}{\partial x_1}\} = p_3(1-p_2) \tag{A.31}$$

$$Pr\{\frac{\partial f}{\partial x_2}\} = p_1p_3 \tag{A.32}$$

$$Pr\{\frac{\partial f}{\partial x_3}\} = p_1(1-p_2) \tag{A.33}$$

$$Pr\{\frac{\Delta f}{\Delta(x_1x_2)}\} = p_3(p_1 + p_2 - 2p_1p_2) \tag{A.34}$$

$$Pr\{\frac{\Delta f}{\Delta(x_1x_3)}\} = 1 - p_1 - p_3 - 2p_1p_3(1-p_2) + p_2(p_1+p_3) \tag{A.35}$$

$$Pr\{\frac{\Delta f}{\Delta(x_2x_3)}\} = p_1(p_2 + p_3 - 2p_2p_3) \tag{A.36}$$

$$Pr\{\frac{\Delta f}{\Delta(x_1x_2x_3)}\} = p_2p_3 + p_1(1-p_2+p_3+p_2p_3) \tag{A.37}$$

Step 4:

$$\begin{aligned}\varepsilon_{in} =& \varepsilon_1(1-\varepsilon_2)(1-\varepsilon_3)Pr\{\frac{\partial f}{\partial x_1}\} + \varepsilon_2(1-\varepsilon_1)(1-\varepsilon_3)Pr\{\frac{\partial f}{\partial x_2}\}+\\ &\varepsilon_3(1-\varepsilon_1)(1-\varepsilon_2)Pr\{\frac{\partial f}{\partial x_3}\} + \varepsilon_1\varepsilon_2(1-\varepsilon_3)Pr\{\frac{\Delta f}{\Delta(x_1x_2)}\}+\\ &\varepsilon_1\varepsilon_3(1-\varepsilon_2)Pr\{\frac{\Delta f}{\Delta(x_1x_3)}\} + \varepsilon_2\varepsilon_3(1-\varepsilon_1)Pr\{\frac{\Delta f}{\Delta(x_2x_3)}\}+\\ &\varepsilon_1\varepsilon_2\varepsilon_3Pr\{\frac{\Delta f}{\Delta(x_1x_2x_3)}\}\end{aligned} \tag{A.38}$$

$$\begin{aligned}\varepsilon_{in} =& \varepsilon_1(1-\varepsilon_2)(1-\varepsilon_3)(p_3(1-p_2))+\\ &\varepsilon_2(1-\varepsilon_1)(1-\varepsilon_3)(p_1p_3)+\\ &\varepsilon_3(1-\varepsilon_1)(1-\varepsilon_2)(p_1(1-p_2))+\\ &\varepsilon_1\varepsilon_2(1-\varepsilon_3)(p_3(p_1+p_2-2p_1p_2))+\\ &\varepsilon_1\varepsilon_3(1-\varepsilon_2)(1-p_1-p_3-2p_1p_3(1-p_2)+p_2(p_1+p_3))+\\ &\varepsilon_2\varepsilon_3(1-\varepsilon_1)(p_1(p_2+p_3-2p_2p_3))+\\ &\varepsilon_1\varepsilon_2\varepsilon_3(p_2p_3+p_1(1-p_2+p_3+p_2p_3))\end{aligned} \tag{A.39}$$

Step 5:

$$\varepsilon_{LUT3} = \varepsilon_g + (1-2\varepsilon_g)\varepsilon_{in} \tag{A.40}$$

$$\begin{aligned}\varepsilon_{LUT3} =& \varepsilon_g + (1-2\varepsilon_g)(\{\varepsilon_1(1-\varepsilon_2)(1-\varepsilon_3)(p_3(1-p_2))+\\ &\varepsilon_2(1-\varepsilon_1)(1-\varepsilon_3)(p_1p_3)+\\ &\varepsilon_3(1-\varepsilon_1)(1-\varepsilon_2)(p_1(1-p_2))+\\ &\varepsilon_1\varepsilon_2(1-\varepsilon_3)(p_3(p_1+p_2-2p_1p_2))+\\ &\varepsilon_1\varepsilon_3(1-\varepsilon_2)(1-p_1-p_3-2p_1p_3(1-p_2)+p_2(p_1+p_3))+\\ &\varepsilon_2\varepsilon_3(1-\varepsilon_1)(p_1(p_2+p_3-2p_2p_3))+\\ &\varepsilon_1\varepsilon_2\varepsilon_3(p_2p_3+p_1(1-p_2+p_3+p_2p_3))\})\end{aligned} \tag{A.41}$$

List of Figures

List of Tables

Bibliography

[1] NASA, "Today's Tiny Transistors," The Next Wave Journal, Vol. 20, No. 3, 2014.

[2] A. Jacobs, G. Cieslewski, A. D. George, A. Gordon-Ross, and H. Lam, "Reconfigurable Fault Tolerance: A Comprehensive Framework for Reliable and Adaptive FPGA-Based Space Computing," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 5, no. 4, pp. 21:1–21:30, 2012.

[3] J. S. Hane, B. J. LaMeres, T. Kaiser, R. Weber, and T. Buerkle, "Increasing Radiation Tolerance of Field-Programmable-Gate-Array-Based Computers Through Redundancy and Environmental Awareness," *Journal of Aerospace Information Systems*, vol. 11, no. 2, pp. 68–81, 2014.

[4] J. Anwer, S. Meisner, and M. Platzner, "Dynamic Reliability Management: Reconfiguring Reliability-Levels of Hardware Designs at Runtime," in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, 2013, pp. 1–6.

[5] J. Anwer, M. Platzner, and S. Meisner, "FPGA Redundancy Configurations: An Automated Design Space Exploration," in *2014 IEEE International Parallel Distributed Processing Symposium Workshops (IPDPSW)*, 2014, pp. 275–280.

[6] J. Anwer and M. Platzner, "Analytic reliability evaluation for fault-tolerant circuit structures on FPGAs," in *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2014, pp. 177–184.

[7] ——, "Boolean Difference Based Reliability Evaluation of Fault-Tolerant Circuit Structures on FPGAs," in *2016 Euromicro Conference on Digital System Design (DSD)*, Aug 2016, pp. 1–8.

[8] ——, "Evaluating Fault-Tolerance of Redundant FPGA Structures Using Boolean Difference Calculus," *Microprocessors and Microsystems*, vol. 52, pp. 160 – 172, 2017.

[9] F. L. Kastensmidt, L. Carro, and R. Reis, *Fault-Tolerance Techniques for SRAM-Based FPGAs (Frontiers in Electronic Testing)*. Springer-Verlag New York, Inc., 2006.

[10] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits (2nd Edition)*. Prentice Hall, 2003.

[11] M. Goldstein, "Magnetospheric Physics - Turbulence On a Small Scale," *NATURE*, vol. 436, no. 7052, pp. 782–783, 2005.

[12] J. D. Engel, K. S. Morgan, M. J. Wirthlin, and P. S. Graham, "Predicting On-Orbit Static Single Event Upset Rates in Xilinx Virtex FPGAs," Brigham Young University, Tech. Rep.

[13] B. J. LaMeres, "FPGA-Based Radiation Tolerant Computing." Research Presentation - Montana State University, 2014.

[14] H. Quinn, D. Roussel-Dupre, M. Caffrey, P. Graham, M. Wirthlin, K. Morgan, A. Salazar, T. Nelson, W. Howes, E. Johnson, J. Johnson, B. Pratt, N. Rollins, and J. Krone, "The cibola flight experiment," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 8, no. 1, pp. 3:1–3:22, 2015.

[15] Space-grade Xilinx Virtex 5QV. [Online]. Available: http://www.xilinx.com/products/silicon-devices/fpga/virtex-5qv.html

[16] B. Bridgford, C. Carmichael, and C. W. Tseng, "Single-Event Upset Mitigation Selection Guide," Xilinx Technical Report, XAPP987 v1.0, 2008.

[17] M. M. McCormack, "Trade Study and Application of Symbiotic Software and Hardware Fault-tolerance on a Microcontroller-based Avionics System," Ph.D. dissertation, Department of Aeronautical and Astronautical Engineering, Massachusetts Institute of Technology, 2011.

[18] R. Do, "Automated Triple Modular Redundancy," Web-seminar, Mentor Graphics, 2011. [Online]. Available: http://www.mentor.com/products/fpga/multimedia/automated-triple-modular-redundancy-how-and-when-to-use-it

[19] G. Swift and G. Allen, "Virtex-5QV Static SEU Characterization Summary," Technical report, Xilinx Radiation Test Consortium, 2012.

[20] H. Quinn, K. Morgan, P. Graham, J. Krone, and M. Caffrey, "Eight Years of MBU Data: What Does It All Mean?" Presentation- Single Event Effects Symposium (SEE), 2007.

[21] H. Quinn, K. Morgan, P. Graham, J. Krone, M. Caffrey, and K. Lundgreen, "Domain crossing errors: Limitations on single device triple-modular redundancy circuits in xilinx fpgas," *IEEE Transactions on Nuclear Science*, vol. 54, no. 6, pp. 2037–2043, Dec 2007.

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

[23] M. L. Shooman, "N-modular redundancy," in *Reliability of Computer Systems and Networks: Fault Tolerance, Analysis and Design.* Wiley, 2002, pp. 145–201.

[24] C. Carmichael, "Triple Module Redundancy Design Techniques for Virtex FPGAs," Xilinx Application Note, XAPP197 (v1.0.1), 2006.

[25] S. Lee, J. Jung, and I. Lee, "Voting Structures for Cascaded Triple Modular Redundant Modules," *IEICE Electronics Express*, vol. 4, no. 21, pp. 657–664, 2007.

[26] D. Bhaduri and S. K. Shukla, "NANOPRISM: A Tool for Evaluating Granularity vs. Reliability Trade-offs in Nano Architectures," in *Proceedings of the 14th ACM Great Lakes symposium on VLSI*, 2004, pp. 109–112.

[27] P. Bergsman, "Xilinx FPGA Blasted into Orbit," *Xilinx Xcell Journal,*, vol. 46, pp. 86–88, 2003.

[28] M. Caffrey, K. Morgan, D. Roussel-Dupre, S. Robinson, A. Nelson, A. Salazar, M. Wirthlin, W. Howes, and D. Richins, "On-Orbit Flight Results from the Reconfigurable Cibola Flight Experiment Satellite (CFESat)," in *17th IEEE Symposium on Field Programmable Custom Computing Machines (FCCM)*, 2009, pp. 3–10.

[29] H. Quinn, P. Graham, K. Morgan, Z. Baker, M. Caffrey, D. Smith, and R. Bell, "On-Orbit Results for the Xilinx Virtex-4 FPGA," in *IEEE Radiation Effects Data Workshop (REDW)*, July 2012, pp. 1–8.

[30] D. Ratter, "FPGAs on Mars," *Xilinx Xcell Journal,*, vol. 50, pp. 8–11, 2004.

[31] A. Lesea, S. Drimer, J. J. Fabula, C. Carmichael, and P. Alfke, "The Rosetta Experiment: Atmospheric Soft Error Rate Testing in Differing Technology FPGAs," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 317–328, 2005.

[32] Actel RTAX-S/SL FPGAs. [Online]. Available: http://www.microsemi.com/products/fpga-soc/radtolerant-fpgas/rtax-s-sl

[33] Actel RT ProASIC3 FPGA. [Online]. Available: http://www.microsemi.com/products/fpga-soc/radtolerant-fpgas/rt-proasic3

[34] H. Quinn, "An Introduction to Mission Risk and Risk Mitigation for Xilinx SRAM FPGAs." Presentation- Los Alamos National Laboratory, 2009.

[35] F. H. Schmidt, "Fault Tolerant Design Implementation on Radiation Hardened By Design SRAM-Based FPGAs," Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2013.

[36] I. Herrera-Alzu and M. López-Vallejo, "Self-Reference Scrubber for TMR Systems Based on Xilinx Virtex FPGAs," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6951, pp. 133–142.

[37] G. Miller, C. Carmichael, and G. Swift, “Single-Event Upset Mitigation for Xilinx FPGA Block Memories,” Xilinx Technical Report, XAPP962 (v1.1), 2008.

[38] N. H. Rollins, “Hardware and Software Fault-Tolerance of Softcore Processors Implemented in SRAM-Based FPGAs,” Ph.D. dissertation, Department of Electrical and Computer Engineering, Brigham Young University, 2012.

[39] C. Carmichael and C. W. Tseng, “Correcting Single-Event Upsets in Virtex-4 FPGA Configuration Memory,” Xilinx Technical Report, XAPP1088 v1.0, 2009.

[40] J. Johnson, W. Howes, M. Wirthlin, D. McMurtrey, M. Caffrey, P. Graham, and K. Morgan, “Using Duplication with Compare for On-line Error Detection in FPGA-based Designs,” in *IEEE Aerospace Conference*, 2008, pp. 1–11.

[41] D. L. Foster, *Area Constrained Partial Fault Tolerance.* ProQuest, UMI Dissertation Publishing, 2011.

[42] G. Bosilca, R. Delmas, J. Dongarra, and J. Langou, “Algorithm-based Fault Tolerance Applied to High Performance Computing,” *Journal of Parallel and Distributed Computing*, vol. 69, no. 4, pp. 410–416, 2009.

[43] TMR Tool, http://www.xilinx.com/ise/optional_prod/tmrtool.htm, 2012.

[44] Precision Hi-Rel Synthesis Software, http://www.mentor.com/products/fpga/synthesis, 2012.

[45] BYU EDIF Tools Homepage, http://reliability.ee.byu.edu/edif, 2012.

[46] R. Glein, B. Schmidt, F. Rittner, J. Teich, and D. Ziener, “A Self-Adaptive SEU Mitigation System for FPGAs with an Internal Block RAM Radiation Particle Sensor,” in *IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2014, pp. 251–258.

[47] J. M. Johnson and M. J. Wirthlin, “Voter Insertion Algorithms for FPGA Designs Using Triple Modular Redundancy,” in *Proceedings of the 18th annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, 2010, pp. 249–258.

[48] J. M. Johnson, “Synchronization Voter Insertion Algorithms for FPGA Designs Using Triple Modular Redundancy,” Master’s thesis, Department of Electrical and Computer Engineering, Brigham Young University, 2010.

[49] T. Buerkle, B. J. LaMeres, T. Kaiser, E. Gowens, L. Smoot, T. Heetderks, K. Schipf, L. Clem, S. Schielke, and R. Luhr, “Ionizing Radiation Detector for Environmental Awareness in FPGA-Based Flight Computers,” *IEEE Sensors Journal*, vol. 12, no. 6, pp. 2229–2236, 2012.

[50] CREME96 Tool Website, https://creme.isde.vanderbilt.edu.

[51] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Probabilistic Transfer Matrices in Symbolic Reliability Analysis of Logic Circuits," *ACM Transactions on Design Automation of Electronic Systems*, vol. 13, no. 1, pp. 8:1–8:35, 2008.

[52] J. Han, H. Chen, E. Boykin, and J. Fortes, "Reliability Evaluation of Logic Circuits Using Probabilistic Gate Models ," *Microelectronics Reliability*, vol. 51, no. 2, pp. 468–476, 2011.

[53] T. Rejimon, K. Lingasubramanian, and S. Bhanja, "Probabilistic Error Modeling for Nano-domain Logic Circuits," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 17, no. 1, pp. 55–65, 2009.

[54] N. Mohyuddin, E. Pakbaznia, and M. Pedram, "Probabilistic Error Propagation in a Logic Circuit Using the Boolean Difference Calculus," in *Advanced Techniques in Logic Synthesis, Optimizations and Applications*, K. Gulati, Ed. Springer New York, 2011, pp. 359–381.

[55] J. Liang, J. Han, and F. Lombardi, "New Metrics for the Reliability of Approximate and Probabilistic Adders," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760–1771, 2013.

[56] U. Khalid, J. Anwer, N. Singh, N. Hamid, and V. Asirvadam, "Reliability-Evaluation of Digital Circuits Using Probabilistic Computation Schemes," in *IEEE National Postgraduate Conference (NPC)*, 2011, pp. 1–4.

[57] A. Abdollahi, "Probabilistic Decision Diagrams for Exact Probabilistic Analysis," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2007, pp. 266–272.

[58] D. Bhaduri, S. Shukla, P. Graham, and M. Gokhale, "Reliability Analysis of Large Circuits Using Scalable Techniques and Tools," *IEEE Transactions on Circuits and Systems I*, vol. 54, no. 11, pp. 2447–2460, 2007.

[59] S. Sivaswamy, K. Bazargan, and M. Riedel, "Estimation and Optimization of Reliability of Noisy Digital Circuits," in *Quality of Electronic Design (ISQED)*, 2009, pp. 213–219.

[60] M. R. Choudhury and K. Mohanram, "Accurate and Scalable Reliability Analysis of Logic Circuits," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2007, pp. 1454–1459.

[61] N. Mohyuddin, E. Pakbaznia, and M. Pedram, "Probabilistic Error Propagation in Logic Circuits Using the Boolean Difference Calculus," in *IEEE International Conference on Computer Design (ICCD)*, 2008, pp. 7–13.

[62] E. Johnson, M. J. Wirthlin, and M. Caffrey, "Single-Event Upset Simulation on an FPGA," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, T. P. Plaks and P. M. Athanas, Eds. CSREA Press, Jun. 2002, pp. 68–73.

[63] Xilinx. (2015) Device Reliability Report UG116(v10.2.1). [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug116.pdf

[64] J. Hussein and G. Swift. (2015) Mitigating Single-Event Upsets, Xilinx White Paper(WP395) (v1.1). [Online]. Available: http://www.xilinx.com/support/documentation/white_papers/wp395-Mitigating-SEUs.pdf

[65] M. Berg, "Field Programmable Gate Array (FPGA) Single Event Effect (SEE) Radiation Testing," 2012, NASA Electronic Parts and Packaging Report. [Online]. Available: https://nepp.nasa.gov/files/23779/FPGA_Radiation_Test_Guidelines_2012.pdf

[66] K. Siozios, D. Soudris, and M. Hübner, "A Framework for Supporting Adaptive Fault-Tolerant Solutions," *ACM Transactions on Embedded Computing Systems*, vol. 13, no. 5s, pp. 169:1–169:22, 2014.

[67] A. M. Keller and M. J. Wirthlin, "Benefits of Complementary SEU Mitigation for the LEON3 Soft Processor on SRAM-Based FPGAs," *IEEE Transactions on Nuclear Science*, vol. 64, no. 1, pp. 519–528, Jan 2017.

[68] K. Siozios, I. Savidis, and D. Soudris, "A Framework for Exploring Alternative Fault-Tolerant Schemes Targeting 3-D Reconfigurable Architectures," in *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, July 2016, pp. 336–341.

[69] E. Grade, A. Hayek, and J. Borcsok, "Implementation of a Fault-Tolerant System Using Safety-Related Xilinx Tools Conforming to the Standard IEC 61508," in *2016 International Conference on System Reliability and Science (ICSRS)*, Nov 2016, pp. 78–83.

[70] M. Vavouras and C. S. Bouganis, "Area-Driven Partial Reconfiguration for SEU Mitigation on SRAM-Based FPGAs," in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Nov 2016, pp. 1–6.

[71] M. Brusati, A. Camplani, M. Cannon, H. Chen, M. Citterio, M. Lazzaroni, H. Takai, and M. Wirthlin, "Mitigated FPGA Design of Multi-Gigabit Transceivers for Application in High Radiation Environments of High Energy Physics Experiments," *Journal of Measurement (Elsevier)*, 2017. [Online]. Available: acceptedforpublication

[72] F. Rittner, R. Glein, and A. Heuberger, "Detection and Isolation of Permanent Faults in FPGAs with Remote Access," in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Nov 2016, pp. 1–4.

[73] C. Bolchini, A. Miele, and C. Sandionigi, "A Novel Design Methodology for Implementing Reliability-Aware Systems on SRAM-Based FPGAs," *IEEE Transactions on Computers*, vol. 60, no. 12, pp. 1744–1758, 2011.

[74] F. Wang and V. D. Agrawal, "Soft Error Rate Determination for Nanoscale Sequential Logic," in *International Symposium on Quality Electronic Design (ISQED)*, 2010, pp. 225–230.

[75] C. Yu and C. Zhuo, "Soft Errors Verification for Sequential Circuits," 2015, project Report: Department of EECS, University of Michigan, Ann Arbor, Accessed April 2015.

[76] K. Mohammadi, H. Jahanirad, and P. Attarsharghi, "Fast Reliability Analysis Method for Sequential Logic Circuits," in *International Conference on Systems Engineering (ICSEng)*, 2011, pp. 352–356.

[77] H. Jahanirad and K. Mohammadi, "Sequential Logic Circuits Reliability Analysis," *Journal of Circuits, Systems and Computers*, vol. 21, no. 05, p. 1250040, 2012.

[78] BYU-LANL Triple Modular Redundancy, Usage Guide. Version 0.5.2. Brigham Young University, Configurable Computing Lab, 2009. [Online]. Available: http://reliability.ee.byu.edu/edif/

[79] Y. Cao, "Pareto Front," http://www.mathworks.de/matlabcentral/fileexchange/17251-pareto-front, 2007.

[80] E. Lübbers and M. Platzner, "ReconOS: An RTOS Supporting Hard-and Software Threads," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2007, pp. 441–446.

[81] ——, "Cooperative Multithreading in Dynamically Reconfigurable Systems," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2009, pp. 1–4.

[82] ——, "ReconOS: Multithreaded Programming for Reconfigurable Computers," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 9, no. 1, 2009.

[83] A. Agne, M. Happe, A. Keller, E. Lubbers, B. Plattner, M. Platzner, and C. Plessl, "ReconOS: An Operating System Approach for Reconfigurable Computing," *IEEE Micro*, vol. 34, no. 1, pp. 60–71, 2014.

[84] Benchmarks homepage, http://www.pld.ttu.ee/ maksim/ benchmarks, 2007.

[85] S. Dong, X. Wang, and X. Wang, "A Novel High-Speed Parallel Scheme for Data Sorting Algorithm Based on FPGA," in *2nd International Congress on Image and Signal Processing (CISP)*, 2009, pp. 1–4.

[86] D. Mihhailov, V. Sklyarov, I. Skliarova, and A. Sudnitson, “Optimization of FPGA-based Circuits for Recursive Data Sorting,” in *2010 12th Biennial Baltic Electronics Conference (BEC)*, 2010, pp. 129–132.

[87] Partial Reconfiguration User Guide, 2012. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/ug702.pdf

[88] K. Papadimitriou, A. Dollas, and S. Hauck, “Performance of Partial Reconfiguration in FPGA Systems: A Survey and a Cost Model,” *ACM Transactions on Reconfigurable Technology and Systems*, vol. 4, no. 4, pp. 36:1–36:24, Dec. 2011.

[89] K. Vipin and S. Fahmy, “ZyCAP: Efficient Partial Reconfiguration Management on the Xilinx Zynq,” *IEEE Embedded Systems Letters*, vol. 6, no. 3, pp. 41–44, 2014.

[90] S. Belkacemi, K. Benkrid, D. Crookes, and A. Benkrid, “Design and Implementation of a High Performance Matrix Multiplier Core for Xilinx Virtex FPGAs,” in *2003 IEEE International Workshop on Computer Architectures for Machine Perception*, 2003, pp. 4 pp.–159.

[91] S. Aslan, C. Desmouliers, E. Oruklu, and J. Saniie, “An Efficient Hardware Design Tool for Scalable Matrix Multiplication,” in *2010 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1262–1265.

[92] V. Strassen, “Gaussian elimination is not optimal,” *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, 1969. [Online]. Available: http://dx.doi.org/10.1007/BF02165411

[93] D. McMurtrey, K. S. Morgan, B. Pratt, and M. J. Wirthlin, “Estimating TMR Reliability on FPGAs Using Markov Models,” Brigham Young University, Tech. Rep.