UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

LUIS ALBERTO CONTRERAS BENITES

Automated Design Flow for Applying Triple Modular Redundancy in Complex Semi-Custom Digital Integrated Circuits

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Microeletronics

Advisor: Prof. Dr. Fernanda Lima Kastensmidt

Porto Alegre
July 2018

CIP — CATALOGING-IN-PUBLICATION

Contreras Benites, Luis Alberto

Automated Design Flow for Applying Triple Modular Redundancy in Complex Semi-Custom Digital Integrated Circuits / Luis Alberto Contreras Benites. – Porto Alegre: PGMICRO da UFRGS, 2018.

139 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR–RS, 2018. Advisor: Fernanda Lima Kastensmidt.

1. Fault tolerance. 2. Radiation effects. 3. TMR. 4. Equivalence checking. 5. Semi-custom design flow. 6. ASIC. 7. FPGA. I. Lima Kastensmidt, Fernanda. II. Título.

“All the breaks you need in life wait within your imagination. . . .
Imagination is the workshop of your mind, capable of turning mind energy into accomplishment and wealth

— NAPOLEON HILL

ACKNOWLEDGMENTS

This work is one way to thank my parents William and Yuni for their guidance and love. Also, to my siblings Williams and Melany for being always along with me although the distance.

I want to express my gratitude to my girlfriend Thaiane for being supportive, understanding and a wonderful companion.

I would like to sincerely thank my adviser professor Fernanda Lima for her constant support, patience, motivation, and sharing her expertise and knowledge with me.

Thank CEITEC for providing facilities to allow me to attend the master's subjects during work hours. I am particularly grateful with Janaina Costa for her several advice and help.

I am also thankful for the financial aid provided by CAPES and CNPq during the development of my master course in microelectronics and my training in the CI Brasil Program.

Finally, to the persons that I met in Porto Alegre, Mauricio, Sebastian, Nolberto and Jorge for their invaluable friendship.

ABSTRACT

Radiation effects have been one of the most serious issues in military and space applications. But they are also an increasing concern in modern technologies, even for commercial applications at the ground level. Protection or hardening of integrated circuits against radiation effects can be obtained through the use of enhanced fabrication processes and strategies at different stages of the circuit design. The triple modular redundancy (TMR) technique is a widely and well-known technique employed to mask single faults without detecting them. However, the design of TMR circuits is not automated by commercial electronic design automation (EDA) tools and even they can remove partially or totally the redundant logic. On the other hand, there are several tools that can be used to implement the TMR technique in integrated circuits, although most of them are licensed commercial tools, convenient only for specific devices, or with restricted use because of the International Traffic in Arms Regulations (ITAR) regimen. The present work intends to overcome these issues so a methodology is proposed to automate the design of TMR circuits using a commercial design flow. The proposed approach uses a structured netlist to implement automatically TMR circuits at different granularity levels of redundancy for cell-based and field-programmable gate array (FPGA) designs. Optimization of the resulting TMR circuit is also applied based on the gate sizing approach. Moreover, verification of the implemented TMR circuit is based on equivalence checking, and guarantee its correct functionality and its fault-tolerant capability against soft errors. Experiments with an high-level synthesis (HLS)-derived circuit and an obfuscated description of the ARM Cortex-M0 soft-core are performed to show the use and the advantages of the proposed design flow. Several issues related to the removal of the implemented redundant logic were found as well as the impact in the increment of area caused by the majority voters. Furthermore, the reliability of different TMR implementations of the ARM soft-core synthesized in FPGA was evaluated using emulated-simulation fault injection campaigns. As a result, it was reinforced the high-reliability level of the finest granularity implementation even in the presence of up to 10 accumulated faults and the poorest mitigation capacity corresponding to the replication of flip-flops solely.

Keywords: Fault tolerance. Radiation effects. TMR. Equivalence checking. Semi-custom design flow. ASIC. FPGA.

Fluxo de projeto automatizado para aplicar redundância modular tripla em circuitos digitais semicustomizados complexos

RESUMO

Os efeitos de radiação têm sido um dos problemas mais sérios em aplicações militares e espaciais. Mas eles também são uma preocupação crescente em tecnologias modernas, mesmo para aplicações comerciais no nível do solo. A proteção dos circuitos integrados contra os efeitos da radiação podem ser obtidos através do uso de processos de fabricação aprimorados e de estratégias em diferentes estágios do projeto do circuito. A técnica de TMR é bem conhecida e amplamente empregada para mascarar falhas únicas sem detectá-las. No entanto, o projeto de circuitos TMR não é automatizado por ferramentas EDA comerciais e até mesmo eles podem remover parcial ou totalmente a lógica redundante. Por outro lado, existem várias ferramentas que podem ser usadas para implementar a técnica de TMR em circuitos integrados, embora a maioria delas sejam ferramentas comerciais licenciadas, convenientes apenas para dispositivos específicos, ou com uso restrito por causa do regime ITAR. O presente trabalho pretende superar esses incovenientes, para isso uma metodologia é proposta para automatizar o projeto de circuitos TMR utilizando um fluxo de projeto comercial. A abordagem proposta utiliza um netlist estruturado para implementar automaticamente os circuitos TMR em diferentes níveis de granularidade de redundância para projetos baseados em células e FPGA. A otimização do circuito TMR resultante também é aplicada com base na abordagem do dimensionamento de portas lógicas. Além disso, a verificação do circuito TMR implementado é baseada na verificação de equivalência e garante sua funcionalidade correta e sua capacidade de tolerancia a falhas simples. Experimentos com um circuito derivado de HLS e uma descrição ofuscada do soft-core ARM Cortex-M0 foram realizados para mostrar o uso e as vantagens do fluxo de projeto proposto. Diversas questões relacionadas à remoção da lógica redundante implementada foram encontradas, bem como o impacto no incremento de área causado pelos votadores de maioria. Além disso, a confiabilidade de diferentes implementações de TMR do soft core ARM sintetizado em FPGA foi avaliada usando campanhas de injeção de falhas emuladas. Como resultado, foi reforçado o nível de alta confiabilidade da implemntação com mais fina granularidade, mesmo na presença de até 10 falhas acumuladas, e a menor capacidade de mitigação correspondente à replicação de flip-flops apenas.

Palavras-chave: Tolerância a falhas. Efeitos da radiação. TMR. Comprovação de equivalência. Fluxo de projeto semicustomizado. ASIC. FPGA.

LIST OF ABBREVIATIONS AND ACRONYMS

ALPEN	Alpha-particle source/drain penetration
ASIC	Application specific integrated circuit
ASTEP	Altitude Single Event Effects Test European Platform
CERN	Conseil Européen pour la Recherche Nucléaire
CGTMR	Coarse grain TMR
DD	Displace damage
DICE	Dual interlocked storage cell
DSP	Digital signal processing
EDA	Electronic design automation
ELT	Enclosed layout transistor
FGLTMR	Fine grain local TMR
FGDTMR	Fine grain distributed TMR
FGTMR	Fine grain TMR
FI	Fault injection
FPGA	Field-programmable gate array
HDL	Hardware description language
HKMG	High-k metal gate
HLS	High-level synthesis
LHC	Large Hadron Collider
LSM	Laboratory of Modane
IC	Integrated circuit
IP	Intellectual property
ITAR	International Traffic in Arms Regulations
LEO	Low Earth orbit

LET	Linear energy transfer
MBU	Multiple-bit upset
NIEL	Non-ionizing energy loss
PKA	Primary knock-on atom
RHBD	Radiation Hardening by Design
RHBP	Radiation Hardening by Process
RTL	Register-transfer level
TMR	Triple modular redundancy
SAA	South Atlantic Anomaly
SDC	Silent data corruption
SEE	Single-event effect
SEL	Single-event latch-up
SET	Single-event transient
SEU	Single-event upset
SKA	Secondary knock-on atom
SoC	System-on-a-chip
SOI	Silicon-on-insulator
SRAM	Static random-access memory
STI	Sallow trench isolation
TID	Total-ionizing dose

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

1.1 Motivation

Radiation effects have been one of the most serious issues in military and space applications but currently, they are an increasing concern even for commercial applications at the ground level. The radiation effects in semiconductor devices can vary from data corruption to permanent damage such as parametric shifts and complete device failure. For terrestrial applications, the transient faults or single-event effects (SEEs) caused by ionizing radiation are predominant (BAUMANN, 2005; DODD; MASSENGILL, 2003).

The main factors that affect the sensitivity of electronic devices to radiation are the reduction in the transistor's dimensions and the operating voltages in order to satisfy the demand for higher density, functionality and lower power. Whereas technology scaling enables the design of whole systems into a single die, also known as system-on-a-chip (SoC), they also increase design complexity and encounter reliability issues. Thus, ensuring the correct functionality of SoC designs becomes a challenging task.

In order to deal with the effects of radiation, fault-tolerant techniques are employed to prevent malfunctions of electronic systems. Protection or hardening of integrated circuits (ICs) against radiation effects can be done by the use of enhanced fabrication processes and strategies at different stages of the circuit design such as layout, circuit, system and embedded software levels (HUGHES; BENEDETTO, 2003; NICOLAIDIS, 2005). The last approach is not based on a specific technology node, so it can be employed in commercial design flows. However, the implementation of fault-tolerant techniques is not automated by commercial electronic design automation (EDA) tools. The main issue of considering commercial tools to design radiation hardening circuits is that those techniques are based on some type of redundancy. Unfortunately, one typical optimization performed by the EDA tools is to remove redundant logic structures. As a consequence, the designer must restrict the features of the tool to preserve the inserted redundancies. Specialized tools to harden electronics systems are available as well but they affect the design cost because of the extra cost due to license purchase. Moreover, some country regulatory regimes restrict the export of technology related to spatial and military applications. The ITAR regimen imposed by the United States is a clear example of this limitation (BLOUNT, 2008; MONTENEGRO; PETROVIC; SCHOOF, 2010).

Table 1.1: Dedicated tools to implement TMR circuits

Tool	Vendor/Institution	License	Design level	Device
XTMR	Xilinx	Commercial	Synthesis	FPGA
Sinplify Premier	Synopsys	Commercial	Synthesis	FPGA
Precision Hi-Rel	Mentor	Commercial	Synthesis	FPGA
FTMR	Cobham Gaisler	Commercial	RTL	FPGA
TMRG	CERN	Internal use	RTL	ASIC
TLegUp	Macquarie University	Academic	HLS	FPGA
BL-TMR	Brigham Young University	Academic	Netlist	FPGA
-	IHP	Academic	Netlist	ASIC

Source: From the author

1.2 Related work

There are several dedicated tools that are able to implement the TMR technique in ASIC and FPGA designs (XILINX, 2017; SYNOPSYS, 2012; MENTOR, 2018; PRATT et al., 2005; KULIS, 2017; HABINC, 2002; LEE et al., 2017; STAMENKOVIC; PETROVIC; SCHOOF, 2013). Table 1.1 lists some of them and also indicates the step in the design where it is applied the replication of the circuit.

In spite of the features of the aforementioned tools to implement TMR design, none of them automates the verification of the resulting TMR circuit. Fault injection (FI) campaigns are usually employed to observe the behavior of a system in presence of faults. According to the complexity of the system and the design stage where is performed the FI, it can incur in large simulation times and even discover the TMR implemented was removed partially or completely after fabrication of the circuit. An interesting approach is the use of equivalence checking to speed up the verification process in an early design stage. Authors in (MELANY; LABEL; PELLISH, 2016; BELTRAME, 2015; BURLYAEV, 2015; SESHIA; LI; MITRA, 2007) details the challenges of using formal verification for design verification against soft errors.

1.3 Objective

This work intends to overcome the issues encountered in the use of commercial EDA tools to design fault-tolerant circuits. The well known TMR technique is employed for that purpose. It was developed a methodology to automate the implementation, optimization, and verification of the TMR technique for complex circuits.

The main features of the devised approach in this work are:

- Reuse of any behavioral description such as HLS, RTL, and structured netlist.
- No dependence in the code-style of the behavioral description.
- Selection of the desired granularity of the TMR technique.
- Selectivity to protect an entire system or only its critical blocks.
- Use of customized voter elements.
- Insertion of voters to protect finite state machines.
- Redundancy of designs with undefined blocks (black boxes).
- Implementation of TMR circuit for ASIC and FPGA designs.
- Verification of the functionality and fault tolerant against soft errors.
- Further optimization of the resulting TMR circuit.

1.4 Outline

This document is compound of 6 chapters. In Chapter 2 are described the radiation environments and their operating conditions that affect electronic systems. Moreover, the mechanisms of the radiation effects are explained in order to understand its interaction with electronics. Chapter 3 brings a compilation of several techniques to mitigate the effects of radiation in digital ICs. Chapter 4 shows the additional steps of the proposed fault-tolerant design flow to automate the design of TMR circuits. Chapter 5 shows the use of the proposed methodology through experimentation in two study-case circuits. Eventually, the conclusion of the present work and future works are exposed in Chapter 6.

2 RADIATION EFFECTS IN INTEGRATED CIRCUITS

2.1 Introduction

Development of man-made technology is highly driven by the use of microelectronics. Furthermore, the technological development of these miniaturized devices is powered by the advance of the semiconductor industry. The transistor, which is compound of complex semiconductor structures, becomes smaller, faster, dissipates less power and is cheaper to fabricate in each new technology generation. Despite these advantages, the constant shrink of the transistor's dimensions along with the shortcomings of the lithographic techniques employed in the fabrication process cause that characteristics of manufactured devices to be unpredictable. Moreover, devices become more susceptible to failures induced by radiation.

The levels of radiation that semiconductor devices encounter during their life cycle depend on the radiation environment and their operating conditions. In general, it is possible to identify three main environments where man-made technology is deployed and exposed to the effects of radiation: space, terrestrial and artificial-made environments. In the following sections, these environments and also the main effects of radiation are detailed.

2.2 Radiation environments

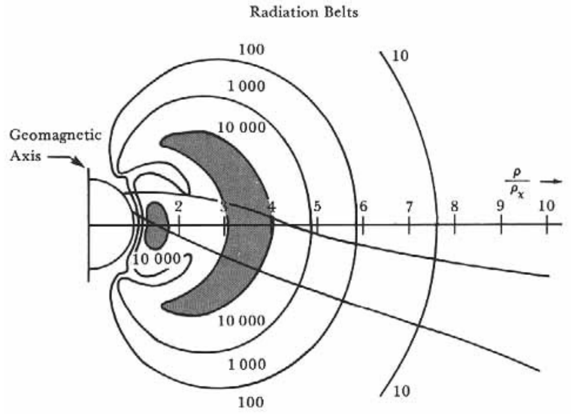
2.2.1 Space environment

The space environment is governed by three main sources of radiation: galactic cosmic rays, particles emitted by the Sun and particles trapped into earth's magnetic field.

Galactic cosmic rays are particles originated outside our solar system. Protons are the predominant particles in this environment (about 85%). These particles can reach very high energies of up to hundreds of GeV. As a consequence, the associated high power penetration makes unpractical to shield using reasonable amounts of material. Fluxes of these particles are in order of a few particles per cm^2 per second (CLAEYS; SIMOEN, 2002)

Solar activity generates solar particles that includes all types of elements, from protons to uranium and their flux depends on the solar cycle. The flux of solar particles

Figure 2.1: Two doughnut-shaped regions of trapped radiation with maximum intesity (inner and outter radiation belts)



Source: (Van Allen; FRANK, 1959)

can reach 10^5 particles/cm^2/s with energies higher than 10 MeV (MOOR; DE, 2011). The solar activity consists of a cycle of 7 years of high activity and 4 years of low activity. Solar cycle influences galactic cosmic ray flux. During the high solar activity, the flux of cosmic rays is reduced due to the shielding effects of solar particles.

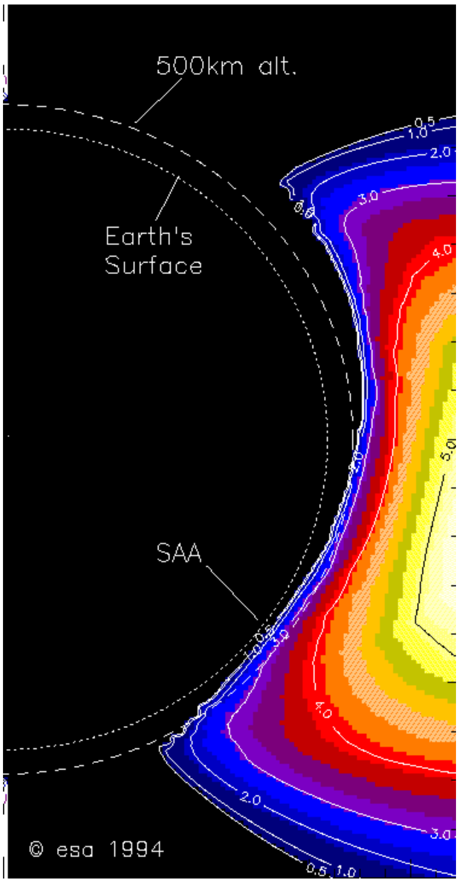
Charged particles can be trapped by the Earth's magnetic field. These trapped particles form two permanent doughnut-shaped regions knows as van Allen Belts (Van Allen; FRANK, 1959). The Inner belt is located from about 500 Km from the earth's surface and it extends to 13000 Km. Protons and low-energy electrons (from 1 up to 5 MeV) compose the inner belt. The Outer belt, located further out the Inner belt, contains high energy electrons with energies in the range from 10 to 100 MeV. Figure 2.1 depicts these charge trapped regions.

The earth's magnetic field is tiled 11°with respect to the geographic axis. In consequence, van Allen belts come closer to the earth surface over South America and South Atlantic (Figure 2.2). This region is called as South Atlantic Anomaly (SAA), where spacecrafts traveling inside the low Earth orbit (LEO) encounters the Inner belt (Figure 2.3).

2.2.2 Terrestrial environment

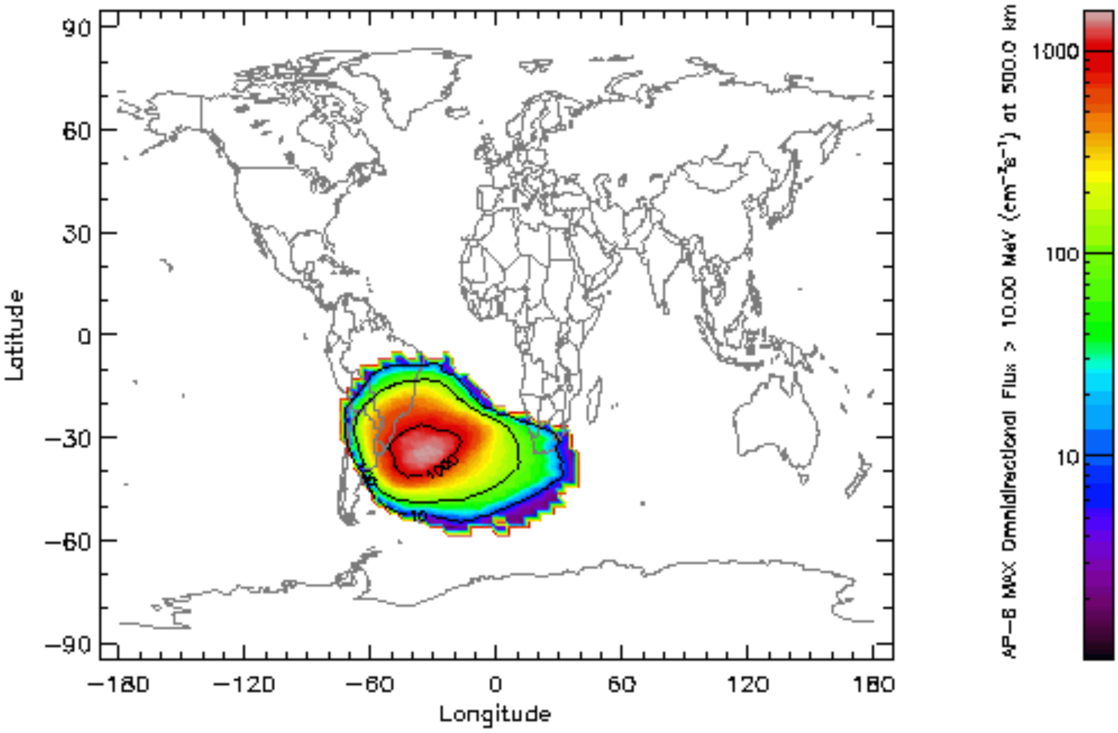
Particles in the wind solar are not able to penetrate Earth's atmosphere. Only cosmic ray particles with energies of more than 2 GeV go through the atmosphere layers and interact with Nitrogen and Oxygen atoms to produce a cascade of secondary particles (Figure 2.4). These high-energy cosmic rays, mainly protons and helium nuclei, are

Figure 2.2: Inner belt is closer to the ground over South America and the South Atlantic



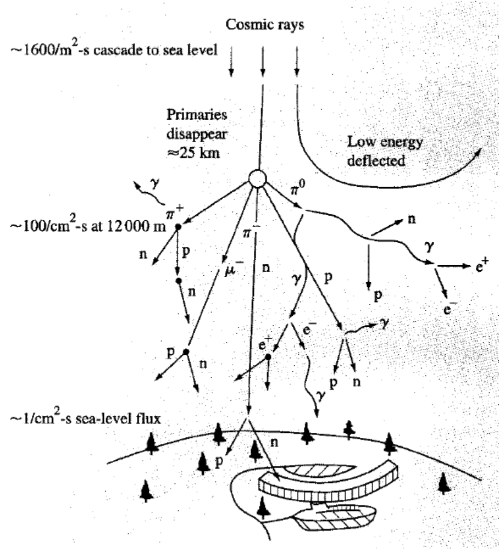
Source: (NASA, 2018)

Figure 2.3: Map of radiation dosage over South Atlantic Anomaly (SAA)



Source: (NASA, 2018)

Figure 2.4: Representation of cosmic rays cascade through the earth's atmosphere



Source: (ZIEGLER et al., 1996)

called primary cosmic rays. The interaction of primary cosmic rays with the atmosphere produces secondary cosmic rays (protons, muons, pions, neutrons) and an electromagnetic component. These particles have enough energy to create further particle cascades. However, the amount of secondary particles decreases when the shielding effect of the atmosphere overcomes the production of particles.

High-energy neutrons (> 10 MeV) or atmospheric neutrons, which are originated in the atmosphere's outer layers, are among the most abundant ionizing particle at the sea level. Despite neutrons are not charged particles, they can trigger nuclear reactions with chip materials, giving rise to charged secondary products. Atmospheric neutron flux depends on altitude, solar activity, latitude, and atmospheric pressure. The neutron flux of New York City, which is 14 n/cm2/hour with energy above 10MeV, is taking as a reference (GORDON et al., 2004). Tables are available in order to calculate neutron flux in other locations (JEDEC, 2006; WILKINSON, 2006). The neutron flux increases with altitude but around 15 km of altitude, neutron flux shows a peak. That is the reason why avionics is an application where neutrons are a serious concern for electronics. The energy spectrum in aircraft altitudes and on the ground is essentially identical (NORMAND; BAKER, 1993). On the other side, thermal neutrons exhibit short lifetime of around 0.1 s and energy levels about 25 meV. Their effects can be considerable because of the high probability of interacting with the boron isotope $^{10}B.$ often found in inter-metal layers of integrated circuits or as a dopant. The reaction produces an alpha particle and $^{7}Li,$ with a combined energy of 2.3 MeV. The flux of terrestrial thermal neutron presents an average value of 4 cm^{-2}h^{-1} at the sea level (DIRK et al., 2003).

Table 2.1: Alpha emissivities of various materials

Material	**Emissivity ($cm^{-2}h^{-1}$)**
Fully Processed Wafers	< 0.0004
30um thick Cu Metal	< 0.0003
20um thick AlCu Metal	< 0.0003
Mold compound	< 0.024 – < 0.0005
Flip Chip Underfill	< 0.004 – < 0.0007
Eutectic Pb-based Solders	< 7.2 – < 0.0009

Source: (JEDEC, 2006)

The second source of radiation-induced effects in electronics at terrestrial level is alpha particles. They are produced from the decay of radioactive elements intentionally used in integrated circuit fabrication or unexpected impurities. Table 2.1 shows alpha particle surface emissions from some key production materials used during the fabrication process.

2.2.3 Artificial Man-made radiation

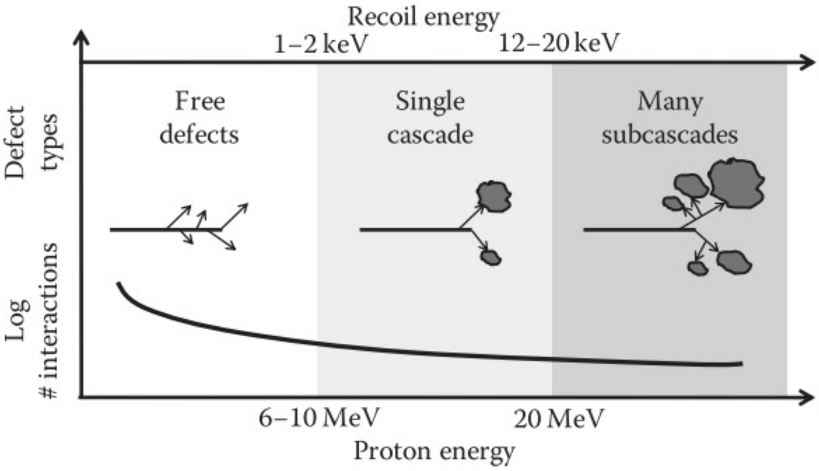
Some man-made radiation environments such as high-energy physics experiments and nuclear plants are very harsh in terms of ionizing radiation. For instance, Conseil Européen pour la Recherche Nucléaire (CERN) plans to use doses above 100 Mrad(Si) in the High-Luminosity Large Hadron Collider (HL-LHC), which is the planned upgrade of the current Large Hadron Collider (LHC) (BAGATIN; GERARDIN, 2015).

Ionizing radiation is also a concern in nuclear fission plants and future fusion plants under development. In the ITER project, which is a magnetic fusion reactor, large fluxes of neutrons with a maximum energy of 14.1 MeV are expected to hit electronic systems (BAGATIN et al., 2012).

2.3 Classification of radiation effects

Exposition of electronic components to radiation can lead to displacement of atoms from their lattice sites and generation of electron-hole pairs (ionization). The former effect is known as displacement damage (DD) meanwhile, the last can be divided into total ionizing dose (TID) and single-event effects (SEE) according to the place where electron-hole pairs are deposited.

Figure 2.5: Defect types and sub-cascade formation related to primary knock-on atom energy in Silicon



Source: (BAGATIN; GERARDIN, 2015)

2.3.1 Displacement Damage (DD)

High-energy particles (neutrons, protons, heavy ions, electrons and indirectly photons) can transmit enough energy to a lattice atom of the target material to dislodge it from its original location. The capacity of an energetic particle to trigger a DD is established by its non-ionizing energy loss (NIEL) coefficient. NIEL measures the amount of energy lost per unit of length of the striking particle due to non-ionizing processes. It is directly related to the linear energy transfer (LET) or stopping power for ionizing events. Atomic displacement leads to the creation of a vacancy in the lattice and an interstitial defect in a non-lattice position (Frenkel pair). The first displaced atom, also known as primary knock-on atom (PKA), can induce further displacements of secondary knock-on atoms (SKA), which are able to generate additional defects. Resulting groups of defects close to each other are denominated clusters (SROUR; MARSHALL; MARSHALL, 2003). Figure 2.5 illustrates the defect types related to DD.

Vacancies travel across the lattice until they become stable because of the high probability of recombination of the Frenkel pair soon after its generation or due to the creation of other types of defects. More stable defects can be the result of the combination of two close vacancies (divacancies) or a vacancy and an impurity close to ones another (defect-impurity complexes).

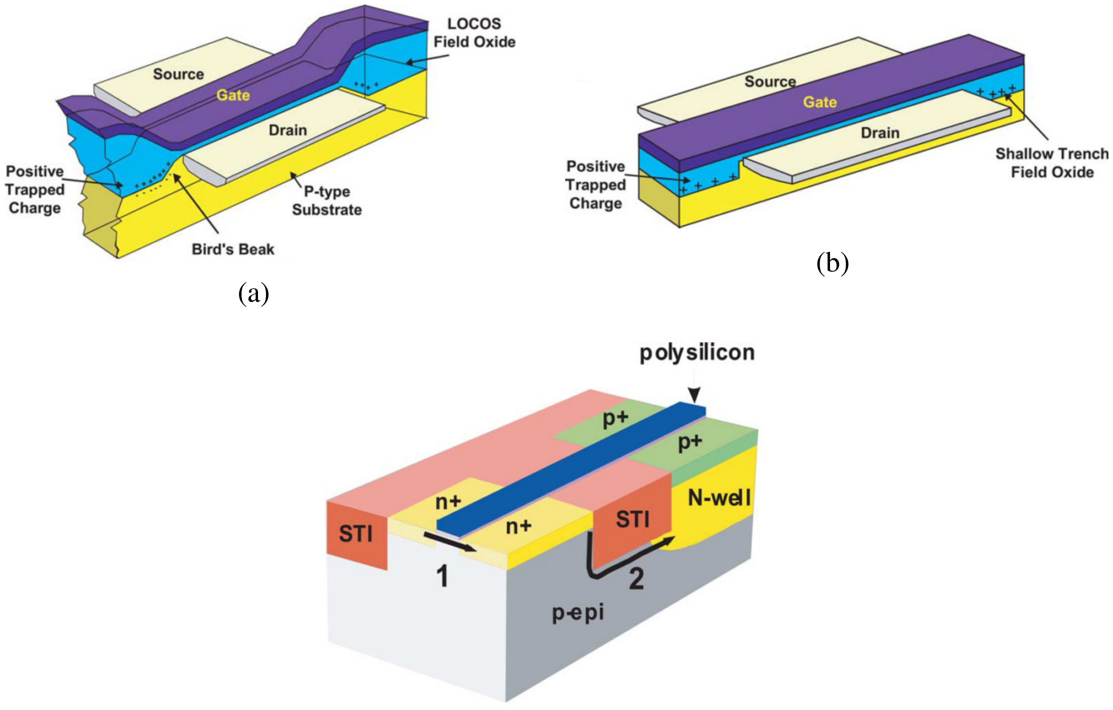
DD causes alteration of electrical properties of the semiconductor crystal and can lead to degradation or failure of the device.

2.3.2 Total Ionization Dose (TID) damage

TID represents the accumulated amount of energy deposited by ionization process in the target material. In electronic devices, radiation-induced currents in insulators are generally not a problem because of lower mobility and lower amount of electron-hole pairs are created. However, charges can be trapped for long period of time. The trapped charges generate internal space-charge electric fields that can lead to a shift in the threshold voltage, reduction in transconductance and leakage current in MOS devices. In modern low-voltage MOSFETs, radiation-induced charge trapping in thin gate oxides is negligible.

TID issues are associated with thick lateral isolation and oxide spacers. Unlike gates oxides that are thermally grown, insulation layers like field oxides and shallow trench isolation (STI), used in older and state-of-the-art devices respectively, are generally deposited (Figure 2.6. So, these oxides exhibit lower quality than gate oxides that becomes them more susceptible to TID effects (SCHWANK et al., 2008).

Figure 2.6: Cross section representation and positive trapped charge of a) LOCOS and b) STI isolated transistors. c) Leakage paths (1 and 2 arrows) in STI transistors



(a)

(b)

(c)

Source: (SCHWANK et al., 2008)

2.3.3 Single-Event Effect (SEE)

An SEE is caused when a single high-energy particle goes through sensitive regions of a microelectronic device. According to the consequences of an SEE over the device, it may be classified as soft effects and hard effects. The former leads to no permanent damage (nondestructive), meanwhile the last causes irreversible physical damage (destructive).

Unlike TID and DD effects that are cumulative and build up over time, an SEE is a stochastic event.

2.3.3.1 Basic mechanisms of SEE

An ionizing particle deposits charge along its path in the target material. In consequence, electron-hole pairs are generated until the striking particle loses all of its energy. The energy loss per unit path length is described by the linear energy transfer (LET). The LET depends on the mass and energy of the incident particle and the target material. In Silicon, a charge deposition of 1 pC/um correspond to a LET of 97 MeV-cm2/mg.

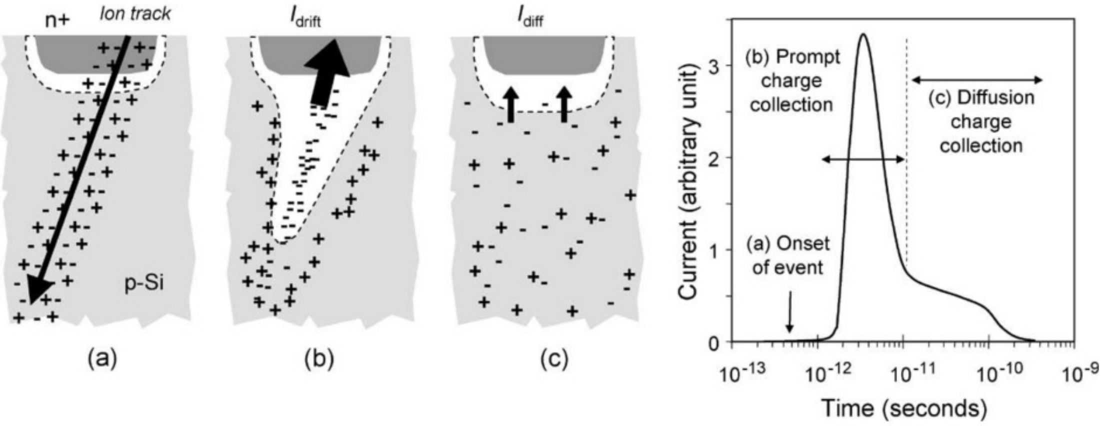
The most sensitive regions under particle strikes are usually reversed-biased p-n junctions. There is a high-electric field into the depletion region of a reversed-biased junction. Thus, electron-hole pairs can efficiently be collected through drift process and it generates a transient current at the junction contact. Charge generated along the particle track can distortion temporarily the electrostatic potential of the junction into a funnel shape (field funnel). This funneling effect can enhance charge collection by extending the junction's electric field away from the junction and deep into the substrate.

Carriers generated beyond the depletion region may diffuse back toward the junction where they can be efficiently collected. However, diffusion occurs on a longer timescale. Figure 2.7 illustrates the resulting current pulse induced by a single radiation event.

In deep sub-micron MOS transistor, charge collection mechanism may be triggered when an alpha-particle at grazing strike passes through both drain and source of the transistor. In consequence, a disturbance in the channel potential can induce a short-lived current that mimics the on-state of the transistor. This charge collection effect is denominated as the alpha-particle source/drain penetration effect (ALPEN) (TAKEDA; HISAMOTO; TOYABE, 1988).

Furthermore, an ionizing particle can generate electron-hole pairs inside the well

Figure 2.7: Charge deposition and drift and diffusion collection processes. Pulse current duration



Source: (BAUMANN, 2005)

or body region of the transistor. The potential of the well is altered as well. For instance, in an NMOSFET, the generated carriers can be collected at the drain/well junction. Hole diffusion raises the potential of the p-well and the source/well junction becomes forward-biased. The source injects electrons into the channel and can be collected at the drain, strengthen the effect of the original particle-induce current. A parasitic bipolar effect is evidenced in this mechanism where the well, the source, and the drain represents the base, emitter and the collector respectively.

2.3.3.2 Types of SEEs

According to the effect of the SEE over the performance of a circuit, they can be classified as permanent or hard and soft for recovery faults. Table 2.2 gives a short description of the types of SEE.

2.4 Radiation effects in modern world

Technology feature size of semiconductor devices has scaled from sub-micron to deep sub-micron, and in the last years to sub-100 nm. TID hardness of modern CMOS microelectronics components is high and the dominated radiation effects are related to single-event effects. Consequences of the rapidly shrinking geometry are the increasing of density integration, high operation frequencies, reduction of nodal capacities, and the use of lower power voltages. These factors contribute to SEEs are one of the key reliability issues in advanced integrated circuits.

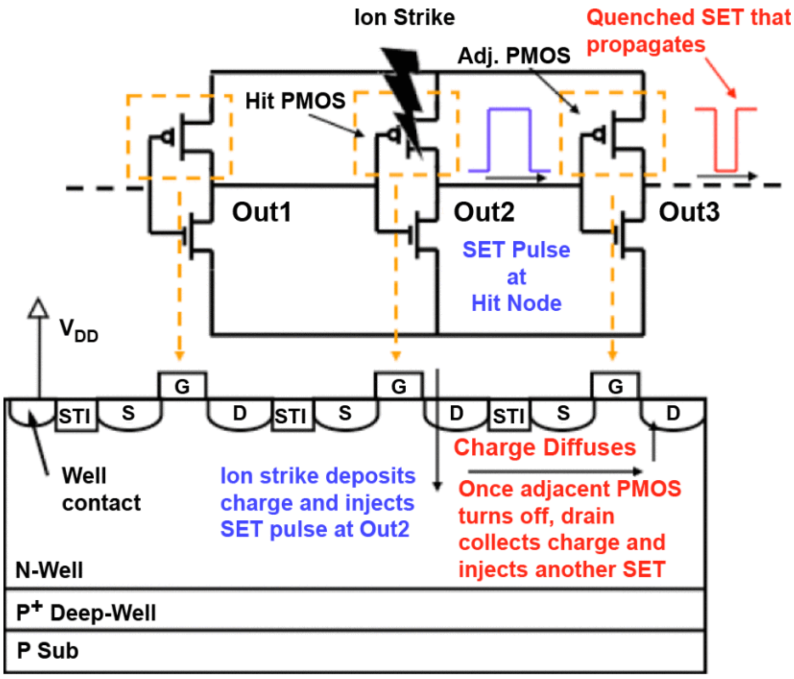
Table 2.2: SEE classification

SEE	Description
Single-event upset (SEU)	Corruption of electric state of a storage cell such as memory cell, latches, and flip-flops. SEU is often called as a soft error.
Multiple-bit upset (MBU)	A single particle strike changes the state of more than one adjacent storage cells.
Single-event transient (SET)	Current/voltage transient pulse that can propagate and be latched by a storage element leads to soft error.
Single-event functional interrupt (SEFI)	Perturbation of control registers or clock signal induces temporal loss of device functionality
Single-event latch-up (SEL)	A strong and sudden increase of power supply current due to radiation-induced activation of parasitic bipolar structures.
Single-event snapback (SES)	A sustained high-current condition in SOI devices. The bipolar amplification effect is similar to SEL
Single-event burnout (SEB)	Activation of parasitic bipolar structures causes burnout of power devices (e.g. power MOSFETs and IGBTs).
Single-event gate rupture (SEGR)	Irreversible rupture of transistor's gate oxide observed especially in power MOSFETs.

Source: From the author

Pulse quenching consists in the SET width pulse reduction due to the delayed charge collection as a direct result of charge sharing. Figure 2.8 illustrates the pulse quenching mechanism (AHLBIN et al., 2009).

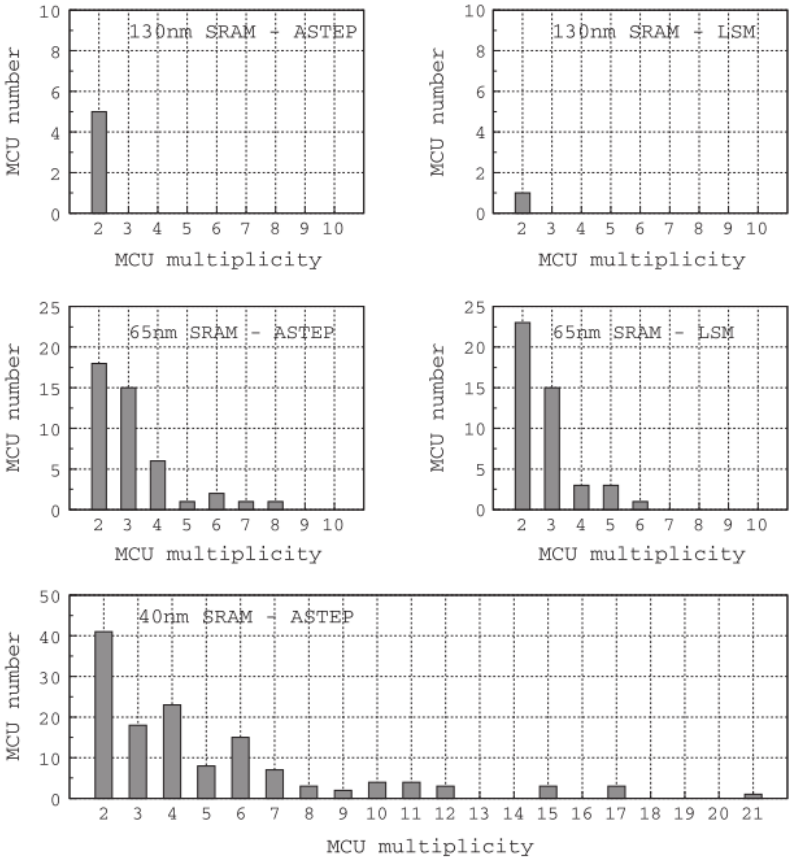
Figure 2.8: Pulse quenching mechanism



Ion Strike
Quenched SET that propagates
Adj. PMOS
Hit PMOS
Out1
Out2
Out3
SET Pulse at Hit Node
V_{DD}
G
G
G
STI
S
D
STI
S
D
STI
S
D
Charge Diffuses
Well contact
Ion strike deposits charge and injects SET pulse at Out2
Once adjacent PMOS turns off, drain collects charge and injects another SET
N-Well
P+ Deep-Well
P Sub

Source: (AHLBIN et al., 2009)

Figure 2.9: Real-time soft-error rate measurements from three technological generations of single-port SRAMs: CMOS 130, 65 and 40 nm

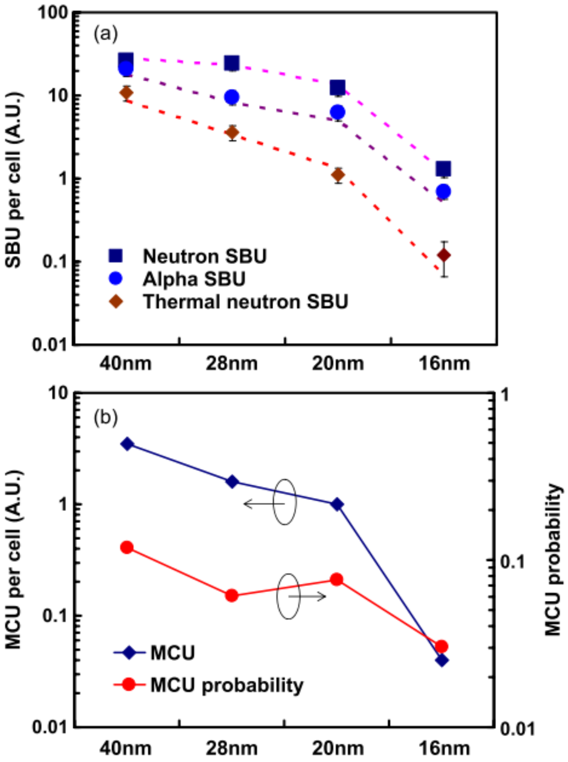


Source: (AUTRAN et al., 2014)

The high density of transistors in the same die enables that multiple transistors can be vulnerable to a single ion strike. Multiple-cell upsets (MCU) events in planar devices can be caused by bipolar amplification (OSADA et al., 2004) and charge sharing (AMUSAN et al., 2008). Authors in (AUTRAN et al., 2014) present real-time soft-error rate measurements of advanced static random-access memory (SRAM) technologies conducted during the last decade on the Altitude Single Event Effects Test European Platform (ASTEP) and at the Underground Laboratory of Modane (LSM). They found experimentally up to 21, 8 and only 2 MCUs for the CMOS 40, 65 and 130 nm single-port SRAM technologies respectively. The MCU multiplicity (i.e. number of bit flips per MCU event) for the different technologies is depicted in Figure 2.9.

Authors in (FANG; OATES, 2016) characterize single-bit upset (SBU) and MCU events for 40, 28, and 20 nm planar SRAMs and 16nm FinFET SRAM. Figure 2.10 shows how both SBUs and MCUs per cell decrease with technology scaling. Moreover, FinFET SRAMs reach a significant reduction in both SBUs and MCUs with regard to planar devices.

Figure 2.10: Measurements of SBU and MCU events as a function of technology node



Source: (FANG; OATES, 2016)

The use of new materials has decreased considerably the soft error rate. It is the case of strained silicon devices that achieve a higher ON current due to the boost of carrier mobility. Thereby, the use of strain transistors increases the driven current as well as the necessary critical charge to induce a current transient. Authors in (MAHATME et al., 2012) show a reduction of around 50% in the total SER and most importantly, the lower SER is achieved without any area penalty.

However, the presence of high-Z materials (like tungsten) can increase the SEU and MCU cross sections of high critical charge devices exposed to terrestrial neutrons (CLEMENS et al., 2011). Besides, even though the BPSG is not utilized in modern semiconductor processes, ^{10}B can subsist within the back-end-of-line (BEOL) structure as a coating over tungsten plugs, for instance (AUTRAN et al., 2014).

Evaluation of HKMG (high-k metal gate) SRAM devices in (ZHANG, 2011) shows a negligible soft error rate induced by thermal neutrons comparing to the effect of terrestrial high-energy neutrons. Also, they conclude that thermal neutrons do not provoke MBU/SEFI/SEL in 32 nm HKMG SRAMs.

2.5 Conclusion

This section brought a survey of the main concepts related to the effects of radiation in electronic devices. In summary, electronic systems are affected by radiation according to the environment where is deployed, and their operating conditions. The radiation environments can be classified in space, terrestrial and artificial environments. Each one of these environments is characterized by its own spectrum of particles and energy distribution. Radiation exposure of electronic components can lead to the displacement of atoms from their lattice sites (displace damage) and generation of electron-hole pairs (ionization). Accordingly, to the regions where electron-hole pairs are deposited into the structure of the transistor, ionization can affects insulators (TID) or reverse-biased p-n junctions (SEE). In modern devices at the sub-100nm node, microelectronic devices are more resilient to TID and SEEs become the key concern in the design of advanced integrated circuits.

3 FAULT-TOLERANCE TECHNIQUES FOR DIGITAL CIRCUITS

3.1 Introduction

The effects of radiation on electronics devices represent an increasing concern for reliability even in terrestrial applications. Furthermore, the use of commercial technologies is not enough to ensure the correct functionality. Hence, strategies that deal with these issues becomes mandatory, especially in deep submicron technologies.

Protection or hardening of circuits against radiation effects can be taken into account in different categories. The employment of enhanced fabrication process is denominated as Radiation Hardening by Process (RHBP). On the other hand, the use of strategies at different stages of the circuit design is known as Radiation Hardening by Design (RHBD). This approach is not based on a technology process so, it can be considered in commercial applications.

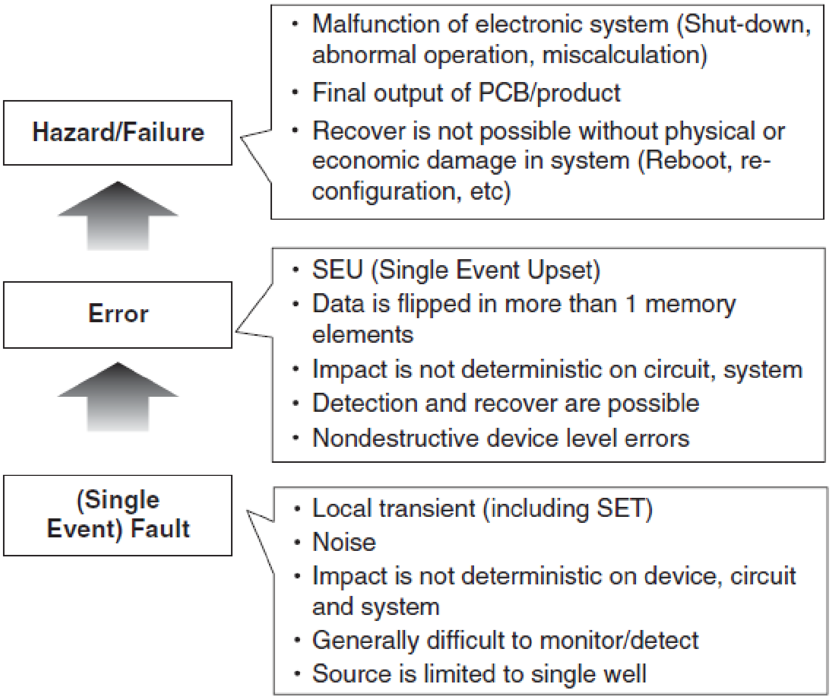
The following sections will give a revision about the most popular RHBD techniques. Knowing these mitigation techniques is useful for planning strategies to accomplish the desired fault tolerance for the circuit. Moreover, the employment of these techniques must be accompanied by a trade-off between the improvement in soft error tolerance and the unavoidable penalties of area, power or performance that implicates their use.

3.2 Fault classification

Any system is far away from being totally free of faults during their operational life. A fault is a physical defect or a flaw in hardware component or software. An error is the manifestation of a fault and represents the deviation from correctness or accuracy in the behavior of a component. Furthermore, the presence of an error in the system can propagate to its final output and cause malfunction. This deviation from the specified functionality of a system is known as a failure. Figure 3.1 depicts the hierarchy of fault, error and failure events.

Faults can be caused by several issues at the specification, implementation and fabrication stages of the design process. Also, external factors such as inappropriate operator's use or environmental disturbances can generate faults. On the other hand, according to the duration of the faults, they can be classified into permanent, intermittent or tran-

Figure 3.1: Fault, failure and error events


Hazard/Failure
• Malfunction of electronic system (Shut-down, abnormal operation, miscalculation)
• Final output of PCB/product
• Recover is not possible without physical or economic damage in system (Reboot, re-configuration, etc)
Error
• SEU (Single Event Upset)
• Data is flipped in more than 1 memory elements
• Impact is not deterministic on circuit, system
• Detection and recover are possible
• Nondestructive device level errors
(Single Event) Fault
• Local transient (including SET)
• Noise
• Impact is not deterministic on device, circuit and system
• Generally difficult to monitor/detect
• Source is limited to single well

Source: (IBE, 2015)

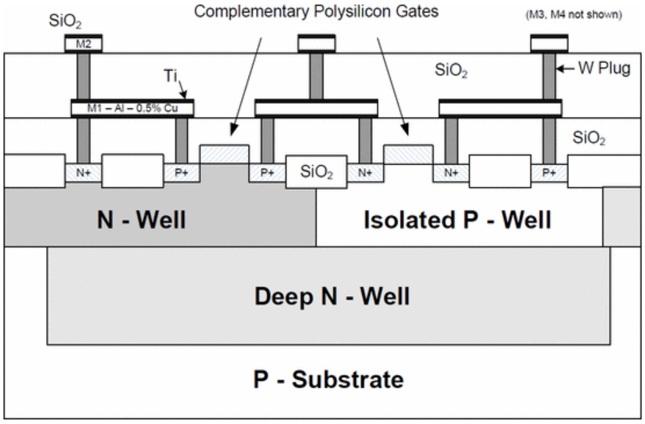
sient. A permanent fault remains continue and stable until a corrective action is taken. An intermittent fault never disappears entirely, it oscillates between being quiescent and active. A transient fault goes away after some time and the functionality of the system is fully recovered. The origin of transient faults is mostly due to environmental conditions as electrical power drops, overheating, mechanical shock, electrostatic discharge, electromagnetic interference or ionizing radiation. This master thesis is focused on the radiation interference.

3.3 Enhanced Process

Commercial CMOS fabrication process is not addressed to deal with the radiation effects. Thus, dedicated processes are used to avoid these undesired effects. The use of enhanced processes to mitigate these effects is called Radiation Hardening by Process (RHBP). The main drawback of using dedicated fabrication processes is its expensive production cost.

For instance, triple-well technology is typically used in deep submicron CMOS technology. This process consists of independent P-well and N-well regions. Thus, an NMOS transistor is electrically isolated in a p-type substrate. Figure 3.2 shows the iso-

Figure 3.2: Triple Well process



Source: (WILSON et al., 2011)

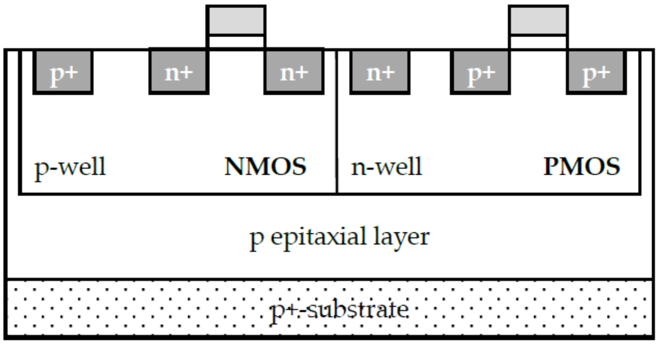
lated P-well embedded into N-well in a P-substrate. The triple-well process reduces substrate noise currents improving transistor performance, limiting the charge deposition by heavy ions and avoiding the latch-up effect (WILSON et al., 2011).

Another well-known approach to reducing the risk of latch-up is the addition of a thin epitaxial layer to increase the bulk resistance as depicted in Figure 3.3. Initially used to mitigate the latch-up effect, however, in scaled technologies epitaxial substrates showed to be affected by latch-up (SHELDON, 2005).

Silicon-on-insulator (SOI) CMOS technology historically was developed for rad-hard and power applications. This once niche market has now reached maturity and is challenging traditional bulk CMOS in all sectors of the market (KONONCHUK; NGUYEN, 2014).

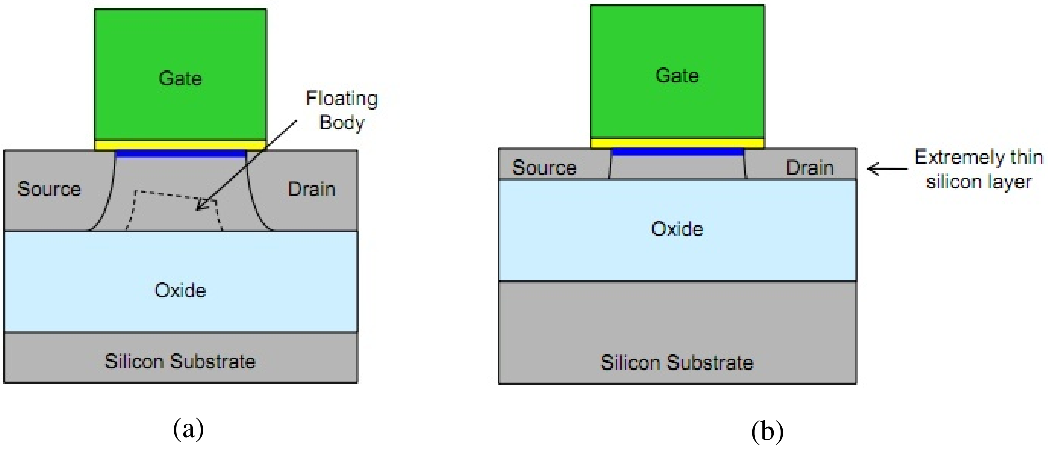
An SOI structure is composed of an insulating layer that separates a top single-crystal silicon layer from the bulk substrate. In microelectronics, it is widely employed as a buried oxide (BOX) layer SiO2 as the insulator. Based on the thickness of the top

Figure 3.3: Epitaxial bulk CMOS



Source: (SECRETARY, 2016)

Figure 3.4: Cross sections of a) Partially Depleted and b) Fully Depleted SOI MOSFETs



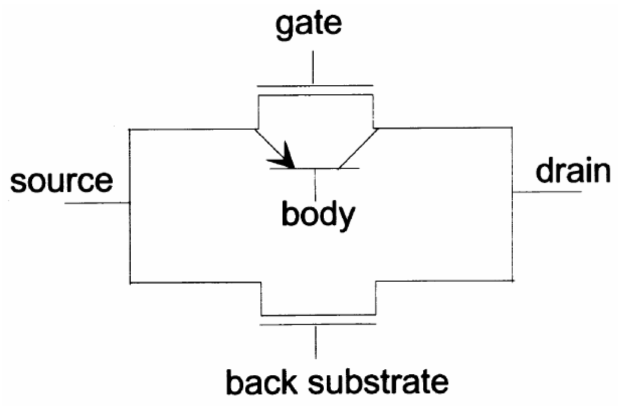
Source: (BOHR; MISTRY, 2011)

silicon layers, two types of SOI transistor structures can be identified: Fully and Partially Depleted Transistors.

Fully depleted (FD) SOI transistor has a very thin film silicon to form the channel (typically less than 50 nm). Hence, the whole body area is underneath the depletion region and there is no need to dope the channel. On the other side, the partially depleted (PD) SOI transistor uses a thicker silicon film (normally greater than 100 nm). This structure exhibits a no depleted area at the bottom of the body, forming a neutral region called floating body. Figure 3.4a and 3.4b illustrates the PD SOI and the FD SOI transistors respectively.

In presence of radiation, parasitic structures illustrated in Figure 3.5 are activated in the SOI structure (SCHWANK et al., 2003). The floating body can behave as the base of a parasitic bipolar transistor. In order to avoid the activation of this undesired element,

Figure 3.5: SOI parasitic a)PBJT and b) MOS



Source: (SCHWANK et al., 2003)

the body region can be tied to the source potential or ground. So, the radiation generated charge can be removed through the body tie. On the other hand, the bulk substrate and the thick BOX can act as the gate and the gate insulator of the parasitic back gate transistor.

SOI devices are completely immune to latch-up, although they can be affected by single-event snapback originated by the parasitic bipolar transistor.

3.4 Radiation Hardening by Design

As an alternative to the use of enhanced CMOS fabrication processes, Radiation Hardening by Design (RHBD) consists of designing integrated circuits using available commercial fabrication processes. In order to accomplish with the desired fault tolerance level, RHBD techniques may be applied at different levels of design. RHBD takes advantage of process maturity, high levels of die yield, availability of many silicon foundries, low cost and quick prototyping, low volume of production and the portability to modern technologies.

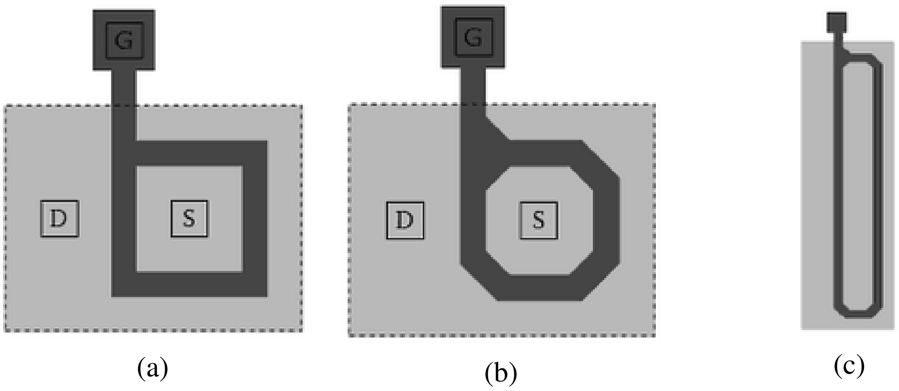
3.4.1 Layout

RHBD layout techniques have been demonstrated to be very effective in eliminating single event latch-up and preventing radiation-induced leakage currents related with Total Ionizing Dose (TID) (CAO PAUL LEROUX, 2015). Furthermore SET and SEU are also mitigated (SECRETARY, 2016). Additionally, standard layout techniques also can improve radiation tolerance. For instance, folded MOSFET transistors are drawn as interconnected fingers with reduced drain and source areas. Thus, the folding technique reduces the sensitive p-n cross-section areas (SCHRIMPF; FLEETWOOD, 2004).

3.4.1.1 Edgeless transistor

A very effective RHBD practice is to avoid the edge leakage. The layout draws a closed gate shape separating the drain and source region of the CMOS transistor. The transistor's edge is designed to connect common implant areas such as source-source or drain-drain or to eliminate completely all the edges, where only the drain or source touch the edge (Figure 3.6). The former receives the name of annular transistor and the last is denoted as the Enclosed Layout Transistor (ELT) (Figure 3.7).

Figure 3.6: Annular transistor with a) 90o angle, b) 45o angle and large W/L case



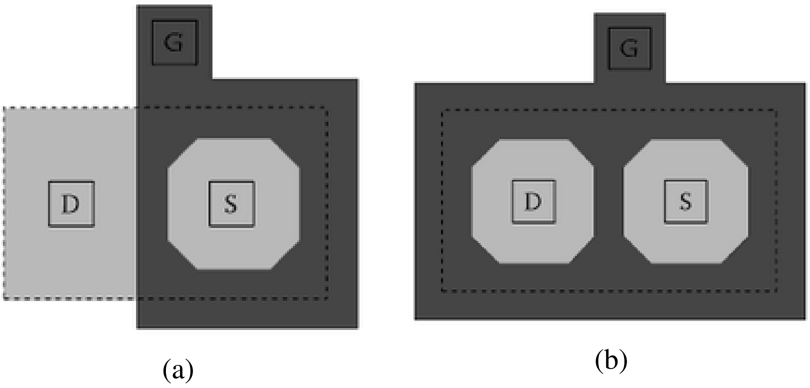
(a) (b) (c)

Source: (CRESSLER; MANTOOTH, 2012)

The main drawbacks of the ELT transistor are its relatively large area and its inaccurate modeling (CAO PAUL LEROUX, 2015). The minimum typical width of the annular transistor is around four to five times the standard transistor's minimum width. On the other side, it is possible to draw a small width ELT (CRESSLER; MANTOOTH, 2012).

Reduction of the drain's area diminishes the cross-section of the device and consequently the sensitivity to SET and SEU. Thus, the drain is generally located on the internal side of the ring-shaped transistor (CAMPLANI et al., 2014). This approach is applied typically on NMOS transistors. PMOS transistors can be designed conventionally because is not prone to current leakage, reducing the area overhead.

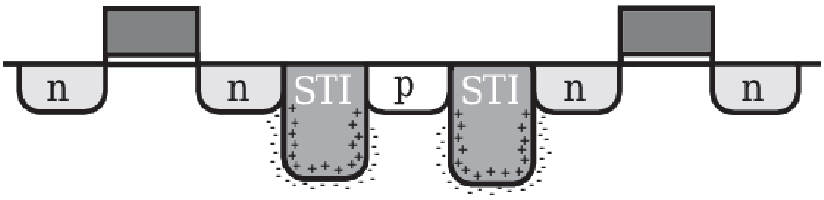
Figure 3.7: Enclosed layout transistors with a) just source enclosed and b) both drain and source enclosed



(a) (b)

Source: (CRESSLER; MANTOOTH, 2012)

Figure 3.8: Cross section of two adjacent NMOS transistor with guard band between them



Source: (CAMPLANI et al., 2014)

3.4.1.2 Guard rings

Guard rings consist of p-well or n-well diffusions that are inserted around NMOS or PMOS transistors respectively. In Figure 3.8, the use of guard rings allows reducing the leakage current through thick field oxides between N-type diffusions of adjacent NMOS transistors. Moreover, guard rings are used to prevent the latch-up effect by reducing the gain of the parasitic thyristor structure.

The main drawback is also the increase of the area of RHBD circuits by the use of additional p-well diffusions. This structure is available in conventional CMOS process. Thus, this approach is commonly employed along with the edgeless layout technique.

3.4.2 Circuit Level

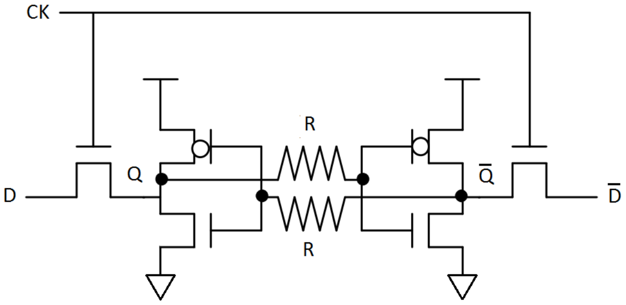
Design of RHBD circuits takes advantage of concepts with regard to circuit topologies, electrical properties, and redundancy. Circuits can contain interconnecting transistors, in order to create logic cells, or use a set of logic cells to design a more complex circuit. Thus, two views of design are defined: cell view and block view respectively.

3.4.2.1 Hardened Memory cells

The most representative case of RHBD circuit at the cell view is the design of hardened memory cells. In the literature can be found a great diversity of storage elements designed to be insensitive to radiation-induced single event upsets. However, most of them are derived from well-known designs. The importance of understanding the operation of this cells is due to they can be found in commercial Rad-Hard libraries of standard cells for ASICs.

The use of decoupling or feedback resistors in SRAM cells is an effective hardening approach to reduce the SEU tolerance (WEAVER et al., 1987). Decoupling resistors

Figure 3.9: Resistor memory cell schematic



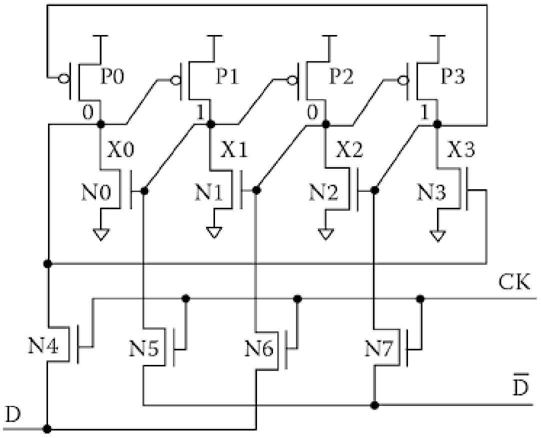
Source: From the author

delays the propagation of the current transient through the regenerative feedback of the SRAM cell. Thus, increasing the resistors' values leads to the reduction of SEU susceptibility of the SRAM cell. Figure 3.9 illustrates the schematic of an SRAM cell with the decoupling resistors.

The Dual Interlocked Storage Cell (DICE) was proposed as a hardened storage element insensitive to single-event effects. It does not require special constraints on transistor sizes (CALIN; NICOLAIDIS; VELAZCO, 1996). The DICE structure is basically a 4-node structure. Figure 3.10 shows the composition of the latch DICE. If a node Xi is affected by a single event effect, the transistors controlled by this node are affected. However, the other transistors float storing its correct values temporally until the SEE disappears. Because of the necessary time to propagate the see through the rest of nodes is very high (AMUSAN et al., 2007), the nodes return to their internal values. Figure 3.10 shows master-slave flip-flops implemented using two DICE latches (WANG; GONG, 2004).

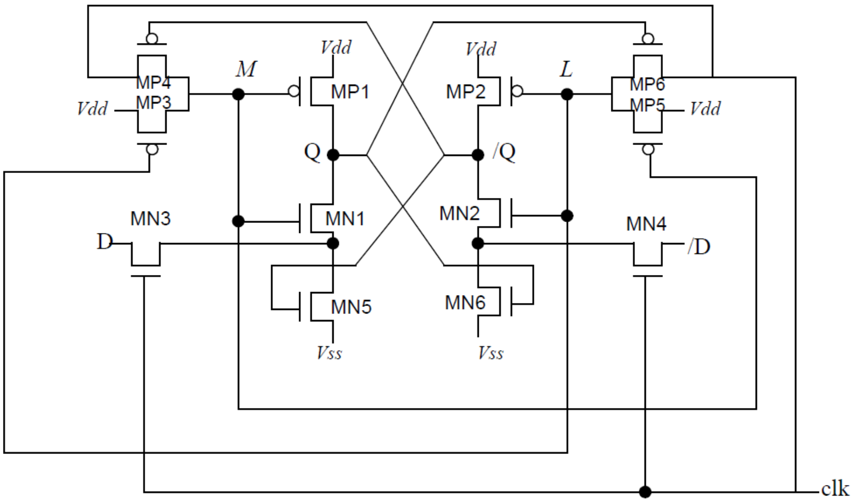
Several commercial rad-hard libraries are based on the DICE structure for memory cells. Cogenda, Atmel, ATK, and Aeroflex are some of them (SECRETARY, 2016).

Figure 3.10: DICE memory cell schematic



Source: (CALIN; NICOLAIDIS; VELAZCO, 1996)

Figure 3.11: HIT memory cell schematic



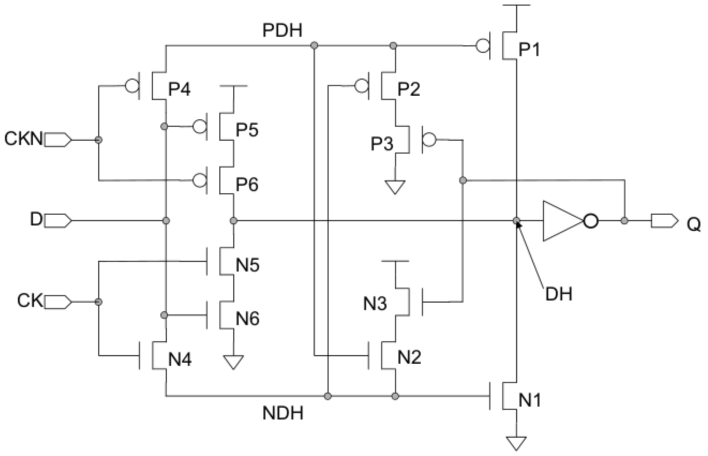
Source: (BESSOT; VELAZCO, 1993)

The Heavy Ion Tolerant (HIT) cell was designed for fast recovery time after a single event upset. Moreover, its low power consumption reduces the impact on performance (BESSOT; VELAZCO, 1993). The HIT cell is built using two storage structures and it is shown in Figure 3.11. The nodes Q and Qhat conserves their respective values while the clock signal is inactive. To modify the value stored in the cell the new values have to be put in the data lines. When the clock signal goes to high, the transistor MP4 or MP6 is responsible for activating the transistors MN1 and MN2. If a particle strikes the nodes Q, Qhat or M for an initial value of Q in high level, the circuit of the cell restores its original values but is affected when more than one out of nodes is attacked.

In (ARIMA et al., 2004) proposes an almost free soft error latch that keeps its state on three storage nodes PDH, NDH, and DH. The error in one node never propagates to the other nodes. The recovery process after a soft error is not depended on the duration of the charge is deposited doing the circuit almost soft-error free for any nuclear radiation energy spectrum. PDH and NDH nodes never fail simultaneously because of the polarity of soft errors; however, one of them along with the DH node might fail as a result of one neutron impact. Figure 3.12 shows the schematic of the Immune Latch Circuit.

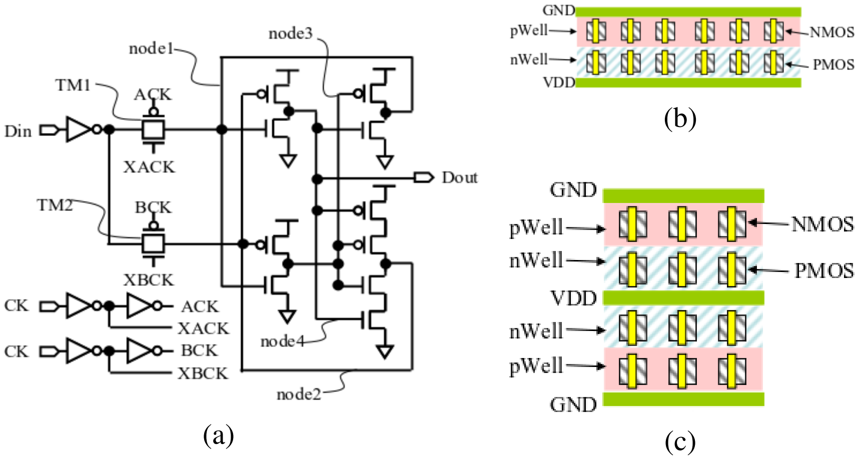
The Soft Error Immune Latch (SEILA) developed in (UEMURA et al., 2010) is a robust latch that enhances single event rate efficiency against multi-node single event effects (MNSEEs) and single event transients on the local clock. The SEILA cell is an inter-lock type storage structure hence data corruption on one node can not corrupt storage data on the storage structure (Figure 3.13a). The data in the latch is corrupted if two critical areas, the NMOS areas on node n1 and n2, are affected by an ion strike. Though

Figure 3.12: Immune Latch diagram



Source: (ARIMA et al., 2004)

Figure 3.13: a) SEILA memory cell schematic, b) and c) shows the same area utilization for single and double height cell schemes respectively

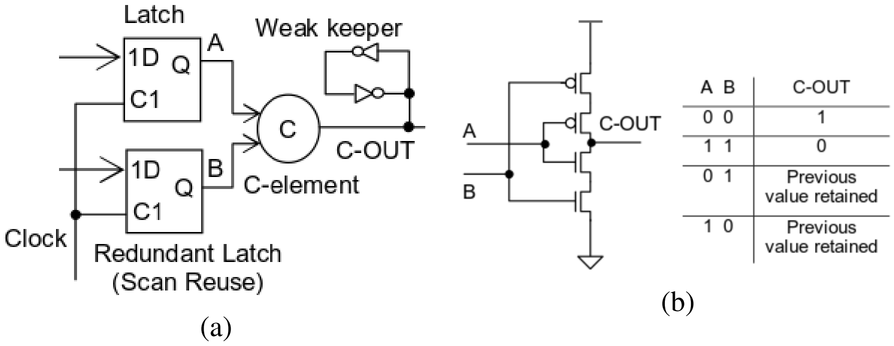


Source: (UEMURA et al., 2010)

if a third area is compromised, the PMOS area on node 3, the data is preserved. The last one is denominated canceling area. In order to accomplish with this placement approach, the layout latch is based on double height cell (DHC) scheme illustrated in Figure 3.13b. Thus the two critical areas are keeping away putting them at the top and at the bottom of the cell layout. Otherwise, MNSEEs are attenuated by placing the canceling area between the two critical areas. It can be noticed that the size of the cell of DHC is not changed from single height cell. The mitigation of transients on the local clock of the latch is achieved by using redundant clock buffers.

In (MITRA et al., 2006) is proposed the Built-In Soft Error Resilience (BISER) technique that uses the advantage of the reduced overhead generated by replication. The BISER technique incorporates a voter for correcting soft errors. The voter is based on

Figure 3.14: a) BISER latch diagram and b) C-element diagram and truth table



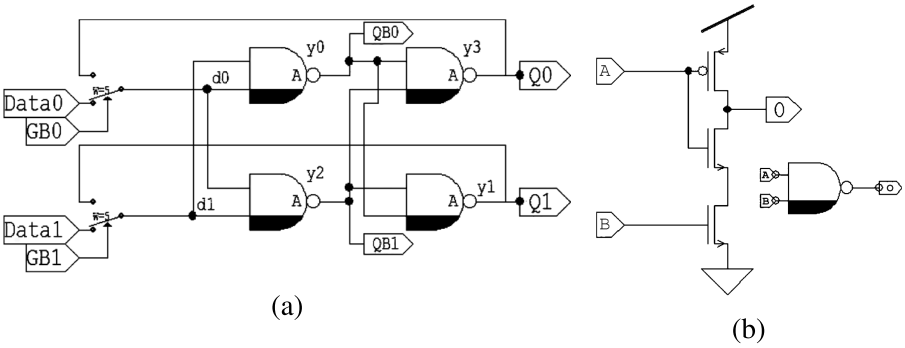
A B	C-OUT
0 0	1
1 1	0
0 1	Previous value retained
1 0	Previous value retained

(a)

(b)

Source: (MITRA et al., 2006)

Figure 3.15: a) SERT diagram and b) half transition NAND gate



(a)

(b)

Source: (SHULER et al., 2009)

the use of the C-element, also called guard gate, along with a weak keeper in its output. Figure 3.14a and Figure 3.14b show respectively the configuration used for correcting errors in latches and the C-element along with its truth table.

The Single Event Resistant Topology (SERT) is a dual-rail latch design. It does not present state conflicts in its internal nodes under single event transient, so recovery is no dependent on transistor sizing (SHULER et al., 2009). The dual-rail SERT latch is shown in Figure 3.15a. The SERT latch scheme consists of an input multiplexer and half transition NAND gates. The half transition NAND gate is almost identical with the C-element (Figure 3.15b) though without one of the series-PMOS.

3.4.2.2 Rad-hard library of digital cells

Radiation hardened (rad-hard) libraries are developed over commercial process based on the use of the previously detailed layout techniques and including hardening memory cells. Also, some rad-hard libraries are developed over RHBP technology to improve the reliability of the circuits.

Table 3.1: Radhard standard cell libraries

Radiation hardened Library	**CMOS Technology node**
ATC18RHA ATMEL	180 nm
ATK	350 nm
C65SPACE STMicroelectronics	65 nm
CERN	250 nm
UT90nHBD/UT130nHBD Cobham	90 and 130 nm
Cogenda	65 and 130 nm
DARE IMEC	180 nm
BAEsystems	150 nm
Radlib Redcat	130, 150 and 180 nm
Ramonchips	130 and 180 nm

Source: From the author

In Table 1.1 are listed several available commercial rad-hard standard cell libraries.

Author in (SHULER, 2015) makes an interesting proposal relying on the uses a small cell count RHBD library. The aim of using a reduced set of cells is to make designs portable between processes of the same technology node. Moreover, to reduce the effort for the migration to modern nodes. The basic 10-cell library is composed by the following gates: an inverter, a buffer, a strong buffer, 3 variations of NAND, only one NOR, two types of 2-input MUX and a tristate buffer. Additionally, it is included a delay cell for SET filtering in dual input circuits. One of the implementations for MUX is for combinational logic use and the other one is employed to build dual interlocked memory cells as DICE or SERT. The tristate buffer may be used to implement the guard gate (C-element).

3.4.2.3 Filtering

A transient pulse can be filtered or masked by the intrinsic properties of a combinational circuit. This natural capability to avoid the propagation of glitches is strongly depended on the physical and the temporal location of the fault. The fault masking mechanisms can be classified into logical, electrical and window (timing) masking.

Logical masking avoids the propagation of a transient fault due to the input values of a multi-input gate, which receives the faulty signal. Electrical masking occurs when a transient pulse is attenuated through its propagation path. So, the pulse amplitude is not enough to induce an error. Furthermore, sizing up a gate reduces the probability that a striking particle cause a glitch of strong magnitude. Authors in (RAO; BLAAUW; SYLVESTER, 2006) show an average reduction in the soft error rate of around 8 times after and area penalty of only 5% after applying gate sizing over a set of benchmark circuits.

Windows masking refers to the case when a transient fault reaches a memory element out of its clocking window. In spite of the natural contention to faults of the aforementioned masking mechanisms, they are not enough to guarantee the correct functionality of a circuit in radiation environments.

3.4.2.4 Redundancy

Redundancy represents the use of additional resources of a system in order to get correct operation even in presence of faults. Moreover, redundant structures can be omitted without affecting the normal operation of a system (LALA, 2001). In the design of circuits that resources may include additional hardware, information, time or a combination of these.

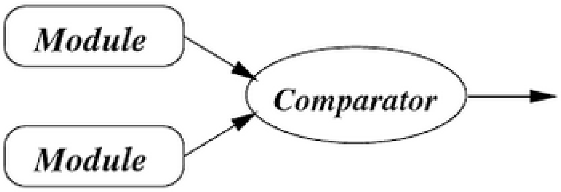
Hardware redundancy

Hardware redundancy, also called space redundancy, is probably the most commonly redundancy approach. It is based on the use of an M-out-of-N system consisting of N modules and at least M of them available for proper operation. This type of systems accomplishes fault tolerance without the need of detecting any fault.

The simplest form of redundancy is the use of only two identical modules and a voting element. This approach is known as Duplex system or Dual Modular Redundancy (DMR). The DMR is only able to detect the presence of a faulty module but further processing is necessary to handle it. Figure 3.16 shows the implementation of DMR.

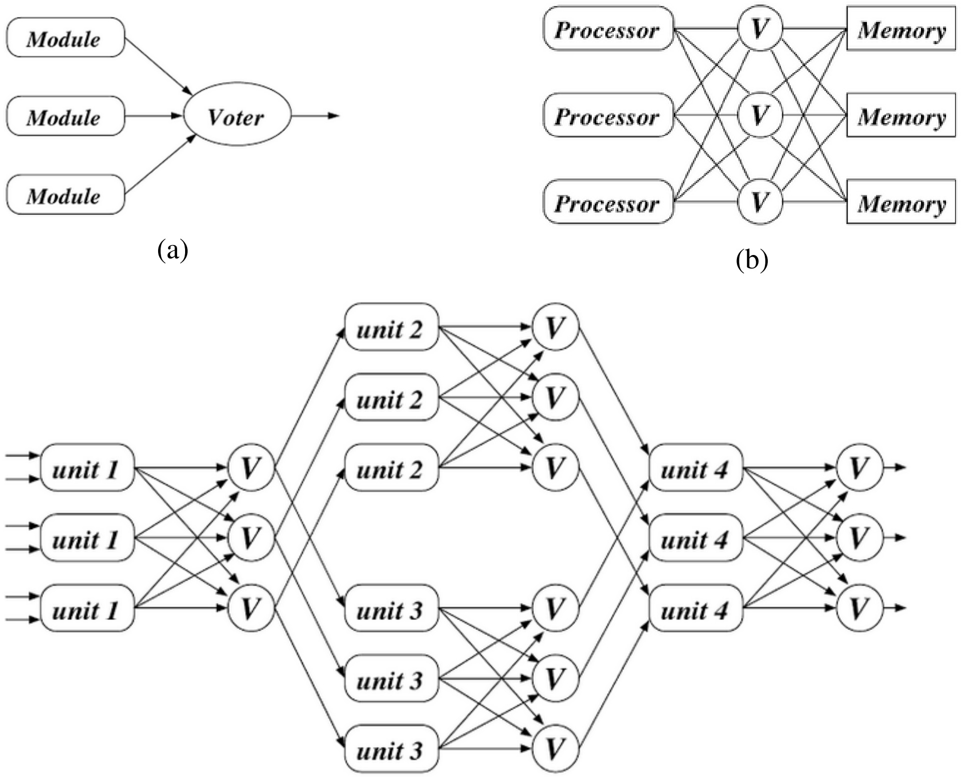
Perhaps the most important M-out-of-N system is the 2-of-3 or Triple Modular Redundancy (TMR) (KOREN; KRISHNA, 2010). The TMR implementation, originally suggested by von Neumann, uses three identical modules and a voting element as illustrated in Figure 3.17a. A module may represent a whole circuit or even a logic gate. The voting element receives the output from the three modules and selects the majority output.

Figure 3.16: Dual Modular Redundancy (DMR)



Source: (KOREN; KRISHNA, 2010)

Figure 3.17: a) Triple Modular Redundant (TMR) structure, b) Triplicated TMR of a processor/memory system and c) Sub-system TMR



(a)

(b)

(c)

Source: (KOREN; KRISHNA, 2010)

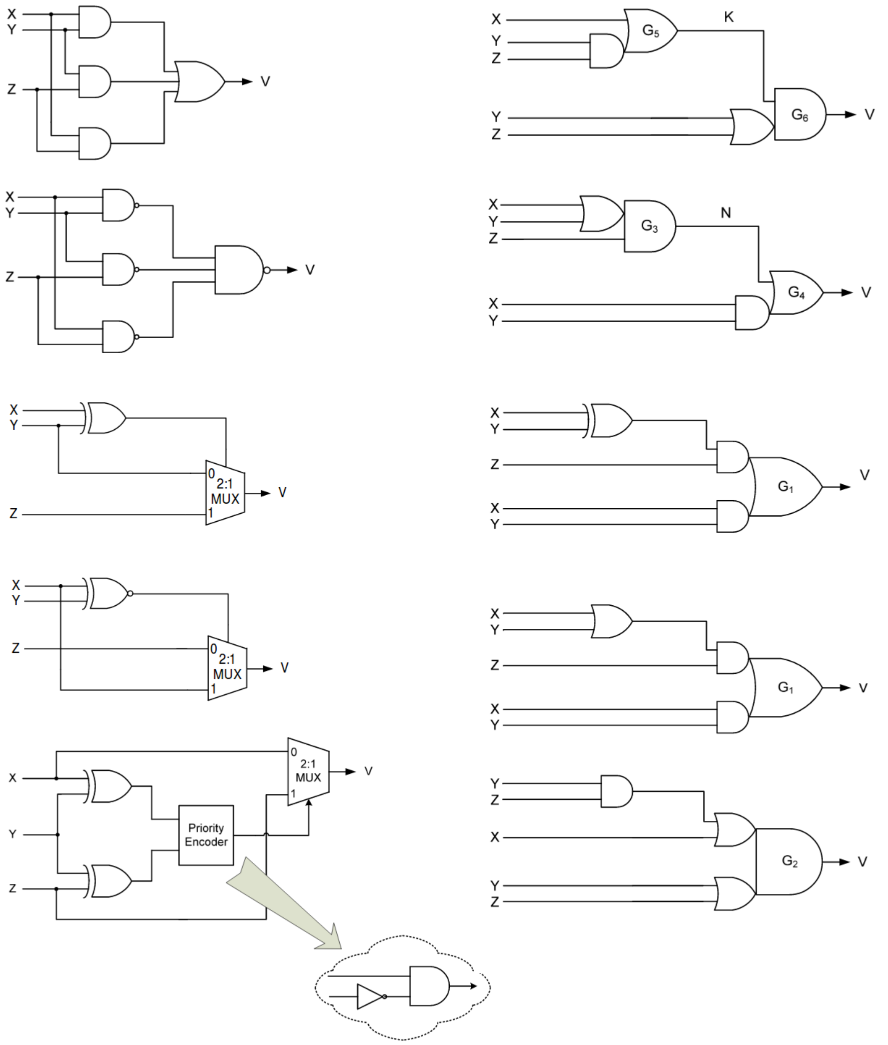
The voter can be considered as a critical point of failure. Thus, the reliability of a voter can be improved by employing three identical copies. This approach is called the triplicated TMR and is depicted in Figure 3.17b.

Furthermore, replication and voting can be applied at the subsystem level instead of the entire system. Figure 3.17c illustrates the application of the TMR technique at the internal individual units of a system.

The fault mitigation success of the TMR technique is conditioned by the voters. They are responsible for giving the final value that interfaces with other modules. In Figure 3.18 is listed a set of alternative implementations for the voter element in TMR designs (BALASUBRAMANIAN; MASTORAKIS, 2015).

The general case of TMR is called N-modular redundancy (NMR) and represents an M-out-of-N system. Practically, N is considered to be an odd number although can also be an even number (example N=4). An NMR system can operate correctly in the presence of up to n module failures, where n = (N-1)/2. For instance, the Quintuple Modular Redundancy (5MR) uses five identical circuits and is able to tolerate up to two erroneous responses in different modules.

Figure 3.18: Different implementations of majority voters for TMR systems

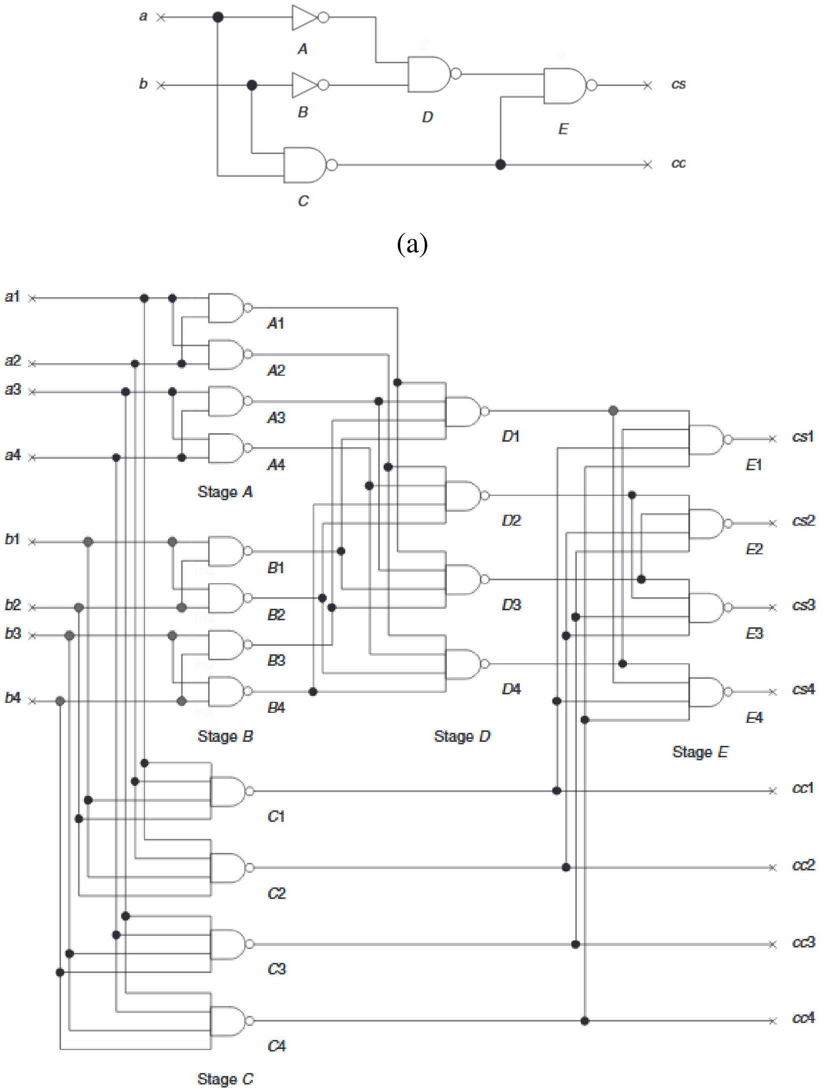


Source: (BALASUBRAMANIAN; MASTORAKIS, 2015)

An alternative to the N-of-M system approach is the interwoven redundant logic. Interwoven logic uses alternating stages of NOR, NAND, AND and OR gates to correct errors using its logic characteristics. These gates own a dominant input defining their output values. For instance, if a 2-input NAND gate receives a '0' in any of its inputs, the NAND's output is always '0'. This means that in presence of the dominant input value, the output is independent of the failure in the other input.

Quadded Logic is an implementation of the interwoven redundant logic and uses the quadruplication of the circuit interconnected in a systematic way to correct errors. Figure 3.19 depicts how is implemented the quadded logic for a given combinational logic. The quadded implementation is obtained by the duplication of the inputs of each gate and dividing its output into two branches to connect them with the next logic stage (HAN et al., 2005). It does not require specific voters for fine grain output elements.

Figure 3.19: a) Schematic of a non redundant complementary half adder implemented with NAND logic and b) Quadded implementation of the complementary half adder



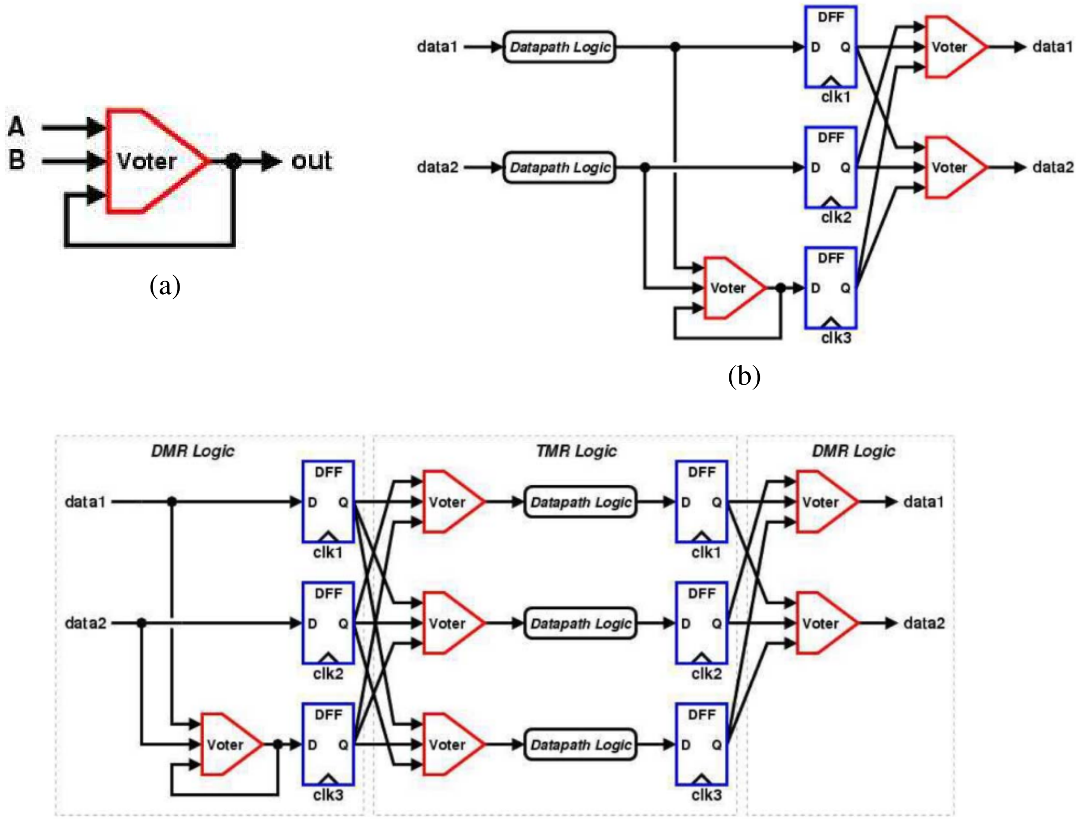
(a)

(b)

Source: (HAN et al., 2005)

The main drawback of using hardware redundancy is the area overhead. (TEIFEL, 2008) presents an alternative TMR implementation taking advantage of module duplication, triplication and the utilization of self-voting elements. The self-voting element is a majority voter identical to the one used for the TMR technique but is used to deliver the majority vote of two combinational logic units' outputs and its own output as is shown in Figure 3.20a. This approach is based on the triplicated TMR with the flexibility of combinational logic units may be replaced by a self-voting element as showed in Figure 3.20b. Also, it is possible does not replace some combinational logic units to have a mixed approach. Figure 3.20c illustrated this combined design.

Figure 3.20: a) Self-voting dual modular redundancy circuit, b) Self-voting majority circuit and c) DMR-to-TMR and TMR-to-DRM mixed circuits



(a)

(b)

(c)

Source: (MITRA et al., 2006)

Information redundancy

The most common type of information redundancy is coding, which involves the addition of check bits to the original data. Employing coding is possible to verify the correctness of the data and even correct the erroneous bits in some cases. For the case of finite state machines (FSM), single fault tolerance can be achieved by the state codification with a minimum Hamming distance of 3.

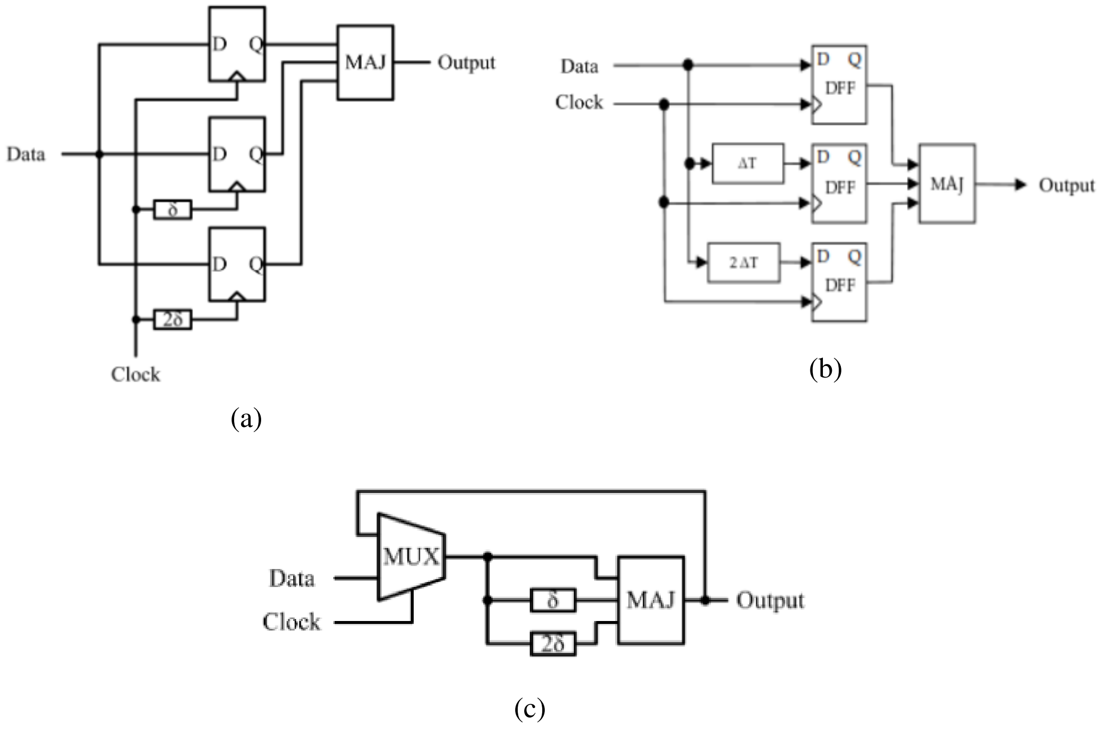
In memory systems, single bit failures may be effectively mitigated by means of Error Checking and Correction (ECC) techniques (LALA, 2001). Furthermore, in order to avoid accumulating errors in memory banks, memory scrubbing technique is employed. It consists in to reload completely or partially the content of memory bits from another safety memory bank. Memory scrubbing is often employed in SRAM-FPGA designs.

Temporal redundancy

Temporal redundancy relays on the use of a module at different times and the comparison of the delayed responses. Unlike hardware redundancy, temporal redundancy achieves fault tolerance without much additional hardware. However, using an extra time for processing impacts negatively in performance. This approach is useful in the presence of transient faults.

The use of temporal redundancy along with the TMR technique is very attractive against the effect of SETs and SEUs. Basically, transient faults in combinational paths can be mitigated by delaying their outputs or the clock signals to feed TMR flip-flops. The inserted delay (dT) must be greater than the transient pulse duration to mitigate it. It is useful to notice the ease of inserting delays in the data path. However, special treatment is necessary when delays are placed in the clock line. Automatic clock tree generation may affect the relation between the inserted delays. Figure 3.21 shows several ways of use delays in latches.

Figure 3.21: Temporal sampling in TMR latch with a) internal clock delays and b) internal data delays. c) Minimal temporal sampling latch replicating itself in time



Source: (MAVIS; EATON, 2002)

3.5 Conclusion

In this chapter, several techniques at different levels of design were reviewed. Radiation hardening can be achieved by the use of specialized fabrication processes (RHBP) and the use of mitigation techniques during the design process (RHBD). Moreover, it is important to identify the effectiveness of each mitigation technique against the different types of radiation effects and the penalties that they involve. Table 3.2 summaries the relation between the reviewed techniques and the radiation effects that they are able to overcome.

In order to design of fault-tolerant digital ICs, Table 3.2 also includes a classification regards to the need of using customized the logic cells. This is considered to highlight the use of mitigation approaches for customized or semi-custom design flows respectively.

Table 3.2: Summary of techniques, the effects that they mitigate and the need of logic gates customization

Mitigation technique	**Radiation effect**	**Customized logic gates**
Triple-well Technology	SEU, SEL	-
Thin epitaxial layer	SEL	-
Silicon on Insulator Technology	SEU, SET, SEL	-
Edgeless transistor	TID	Yes
Guard rings	TID, SET, SEL	Yes
Hardened cells	SEU	Yes
Rad-hard library	TID, SET, SEU, SEL	No
Hardware redundancy	SEU, SET	No
Information redundancy	SEU	No
Temporal redundancy	SEU, SET	No

Source: From the author

4 AUTOMATED TMR DESIGN

4.1 Introduction

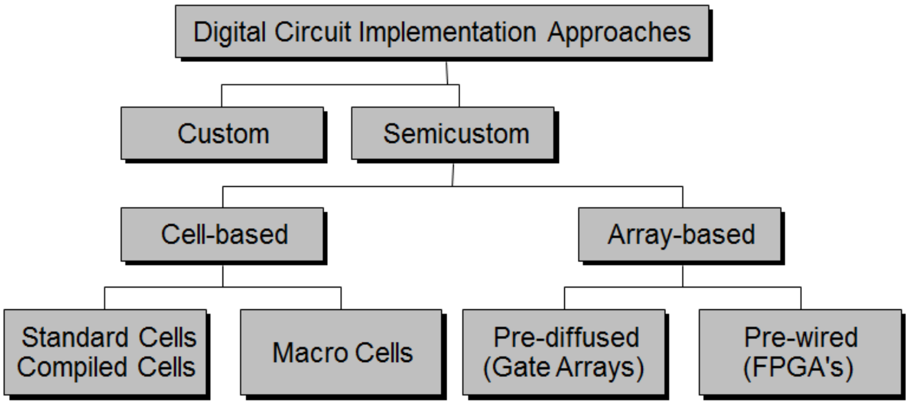
The TMR technique is widely employed to design fault-tolerant designs. Its popularity relays on its capacity to mask single faults without detecting them and with slightly impact over performance. Although area overhead, reduction of transistor dimensions and high-density integration enables its utilization in commercial applications. However, the design of TMR circuits is not automated by commercial Electronic Design Automation (EDA) tools. Moreover, they perform some types of optimization that can affect the implementation of redundancy-based mitigation techniques. In consequence, they can be implemented by hand or by using dedicated tools. The former approach is prone to human error and may become cumbersome for large designs. On the other hand, dedicated commercial tools may be employed to automatically implement TMR circuits. For instance Xilinx TMRTool, Synopsys Sinplify Premier and Mentor Precision HiRel are able to apply TMR in both coarse and fine grain granularity levels during the synthesis process. The use of these tools can add extra cost to the project due to license purchase. Moreover, some country regulatory regimes restrict the export of technology related to spatial and military applications. The International Traffic in Arms Regulations (ITAR) imposed by the United States is a clear example of this. An alternative is the use of the BYU BL-TMR tool to implement TMR designs. In spite of this tool is open source, it can be employed for FPGA designs, specifically for Xilinx's devices. Table 1.1 listed several existing tools to implement the TMR technique and highlighted the shortage of tools to implement the TMR technique in cell-based designs.

In order to overcome these issues, this work presents an approach to automatically design TMR circuits using a commercial EDA tool and following the cell-based and FPGA design approaches.

4.2 Implementation of digital Integrated Circuits

There are several approaches to implement a microelectronics design according to the requirements demanded by the user in terms of speed, power consumption, cost, and production volume. Figure 4.1 shows an overview of the typical approaches to design digital integrated circuits.

Figure 4.1: Typical approaches to implement digital integrated circuits


Digital Circuit Implementation Approaches
Custom
Semicustom
Cell-based
Array-based
Standard Cells Compiled Cells
Macro Cells
Pre-diffused (Gate Arrays)
Pre-wired (FPGA's)

Source: (RABAEY; CHANDRAKASAN; NIKOLIC, 2003)

Custom circuit design includes the design of standard cells or larger circuit blocks at the mask level (custom mask layout). Custom design can be prohibitively expensive and its use is conditioned to reuse, large volume and where cost is the main design criterion. On the other hand, a semi-custom design employs a small subset of fabrication layers in order to reduce costs and automate the design process. Cell-based design reduces implementation effort by the reuse of a limited library of pre-designed cells. A library can be compound of standard-cells and macro cells. Standard-cells consists of logic gates (such as inverter, AND, OR, XOR, and its negative counterparts buffer, NAND, NOR, XNOR respectively), flip-flops and latches as well as complex logic functions, multiplexer, adder, and so on. Macro cells or mega cells are also ready-to-use layout cells but with larger size and complexity than standard-cells. For instance, multipliers, data paths, memories (RAM and ROM), soft-core microprocessors and digital signal processing (DSP) modules. Macro cells can be also acquired from third-party vendors and they are called intellectual property (IP) modules.

The cell-based design reduces the design time however it does not address the manufacturing process time. A complete fabrication run can take from weeks to several months. Hence, the array-based approaches come up to avoid the complete run through the manufacturing process. However, designs exhibit lower performance, lower integration density or higher dissipation than the cell-based approach.

Gate-array and sea-of-gates wafers contain arrays of primitive transistors or cells that are offered by vendors. Only the interconnection layers must be added in order to get the final design. Another option to enable fast implementation is the pre-wired array of cells and also called field-programmable gate array (FPGA). It is a pre-fabricated die that can be configured or programmed for the user outside the semiconductor foundry. The advantage of this approach is the complete separation of the manufacturing process from the design process. In general, there are three different techniques to store the configuration or program based on the employed memory technology: Fused-based, nonvolatile, and volatile or RAM-based FPGAs. The first type is one-time programmable but the area overhead corresponding to the configuration memory is very small. For the second type, a nonvolatile memory is employed to retain the configuration of the design when the power supply is turned off. Electrically erasable Read-Only memory (EEPROM) and Flash memory are examples of nonvolatile memories. Programming and erasing these memory cells require high voltages (> 10 V).

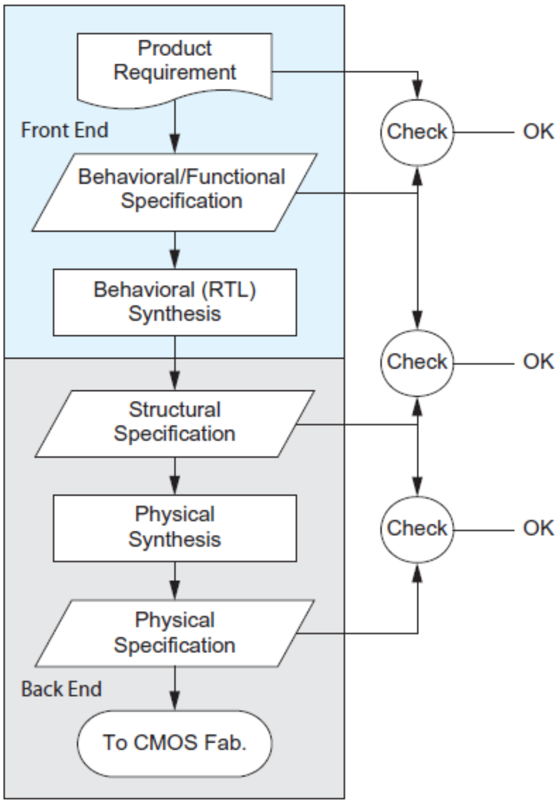
The last type employs commonly volatile static random-access memory (SRAM) cells to store the configuration. In these memories, information is stored on cross-coupled inverters. However, they lose the stored configuration when the FPGA is powered off. FPGAs do not require special manufacturing processes so they can be fabricated in commercial CMOS foundries. Implementation of FPGA designs requires the configuration of both logic functions (programmable logic) and the interconnections among them (programmable interconnection).

4.3 Semi-custom design flow

The semi-custom design flow can be split up into two main stages: front-end and back-end. During the front-end stage, the circuit is described using a hardware description language (HDL) such as Verilog and VHDL, or purely behavioral description languages as C programs and MATLAB models for data and signal processing. Using a high level of abstraction allows the designer to put mainly efforts on logic design and minimize the concern about physical issues. Figure 4.2 shows a generalized semi-custom design flow.

The description of the circuit is processed by a synthesis tool that translates it into a set of interconnected gates. These gates belong to a library of standard cells or logic primitives in the cell-based and FPGA approaches respectively. During logic synthesis, the circuit is optimized with the aim of minimizing area while meeting some time constraints. According to the level of abstraction of the entry design, synthesis can be named: logic synthesis, register-transfer level (RTL) synthesis, and architectural synthesis. Logic synthesis generates combinational gates networks and simple finite state machines (FSMs). A logic synthesis tool accepts the use of logic equations built from basic operators such as NOT, AND, OR, XOR, truth tables, state graphs, and so on. RTL

Figure 4.2: Generalized semi-custom design flow



Source: (WESTE; HARRIS, 2015)

synthesis views an entire circuit as a network compound of storage elements (registers and memories) and combinational logic between them. The circuit is represented by its behavioral specification and it is not limited to logic operations because it can include arithmetic functions, string operation, arrays, etc. Architecture synthesis, which is also called high-level synthesis (HLS), does not explicit the necessary hardware resources. The advantage of this method is the exploration of architectures in order to generate a close-to-optimal design under several constraints as performance, power, and area. Eventually, a gate-level netlist, which contains information about the circuit composition, is generated from the automatic synthesis process.

The back-end stage involves physical aspects of design such as floorplanning, placement, and routing of the netlist's elements. Moreover, the resulting circuit from each automatic transformation step in the design flow must be checked for correctness. The verification approaches that are employed along the design flow include behavioral verification, formal verification, timing analysis, power analysis, and electrical integrity analysis (noise, IR drop and electromigration issues). Once the circuit meets all design constraints and functions, a binary file with the information for mask generation is sent to the semiconductor foundry. This last step is known as tape out. Unlike cell-based design, the last step of the FPGA approach is the bit-stream generation that configures the device.

4.4 Proposed fault-tolerant design flow

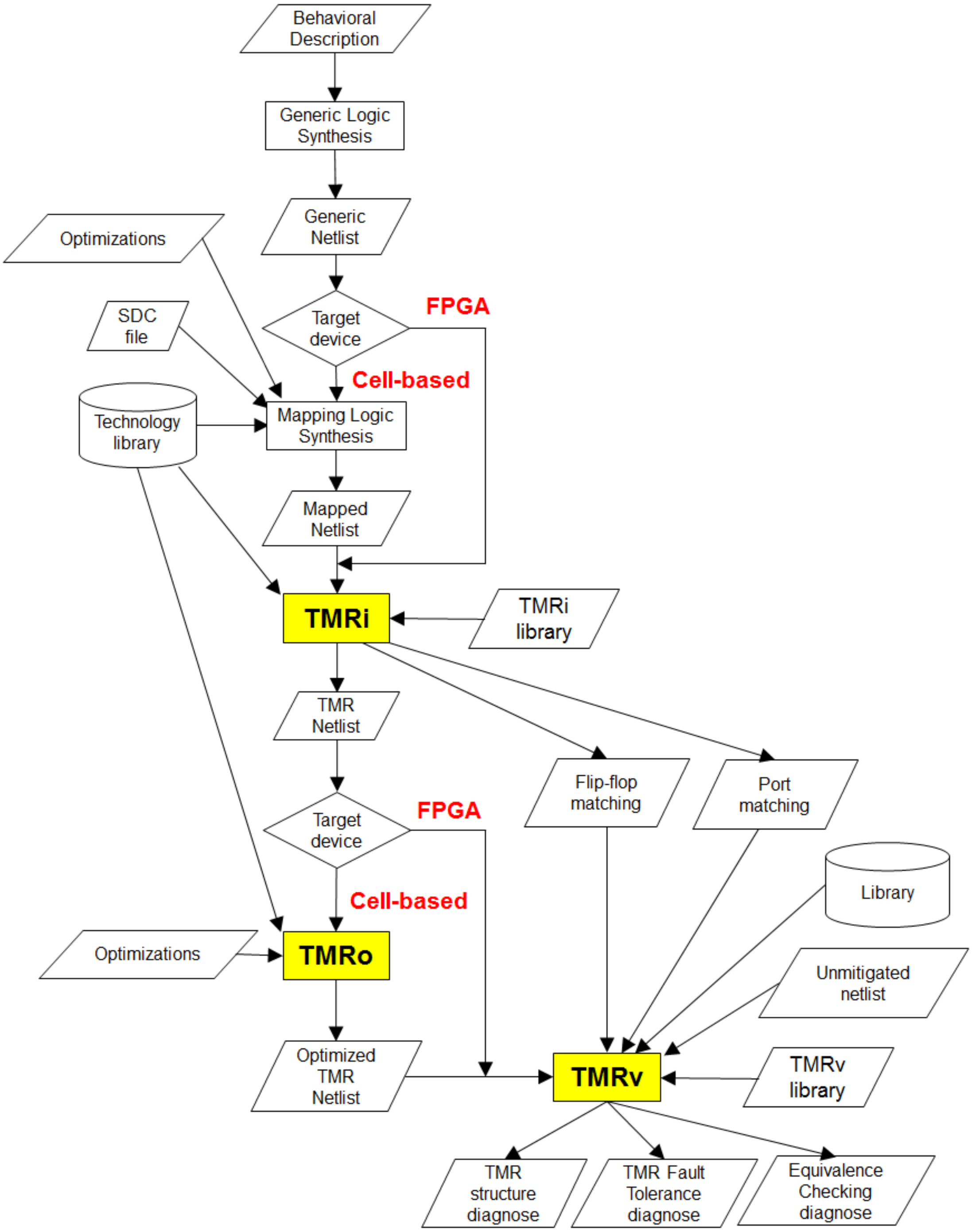
Logic treatment of the circuit is mainly performed during the front-end stage. Thus, the implementation of TMR circuits is suitable at this stage. The TMR technique can be applied during the behavioral design or using the gate-level netlist in a standard design flow. For instance, authors in (ENTRENA; LÓPEZ; OLÍAS, 2001; LEVEUGLE, 2002; HABINC, 2002; KULIS, 2017; LEE et al., 2017; SANTOS et al., 2017) implement redundancy during the RTL and HLS design. Implementing TMR for flip-flops is relatively simple but it is cumbersome to implement it efficiently for combinational logic. Also, in order to be feasible to apply TMR in the behavioral description, the circuit's description needs to be clearly split between combinational and sequential elements. That represents a limitation when it is intended to apply the TMR in behavioral description. Furthermore, that approach needs the circuit to be properly constrained to avoid commercial tool removes the inserted redundant logic. In consequence, over-constraining the design result in a non-optimal implementation. On the other hand, a post-synthesis netlist is handled to make modifications directly in its structure (STAMENKOVIć; PETROVIć; SCHOOF, 2013). The main advantage of this approach is the possibility of using an optimized netlist and preventing easily the elimination of the replicas.

The aforementioned works only show the implementation of TMR circuits but none of them verify if they were successfully implemented. Thus, this work comes up with an integrated solution that automates the implementation and verification processes to design TMR circuits using the semi-custom design flow (BENITES; KASTENSMIDT, 2018). Also, an additional step is considered to attempt to resolve timing issues but preserving the redundant logic in the resulting TMR circuit. Integration of the proposed solution to the design flow is depicted in Figure 4.3. The different steps to design TMR circuits for cell-based and FPGA designs are detailed in the following sections.

The proposed methodology was devised using Cadence EDA tools with the aim of being helpful for researchers from institutions that are part of the Cadence University Software Program and IC design centers in Brazil (CADENCE, 2018),(CI-BRASIL, 2018). The approach can also be extended to other commercial design flows.

In Appendix A it is shown a simple tutorial with the application of the proposed approach to automate the TMR design of a simple circuit from ISCAS'89 benchmark set.

Figure 4.3: Proposed fault tolerant design flow to automate the implementation, optimization and verification steps of TMR circuits



Source: From the author

4.4.1 TMR implementation process (TMRi)

This work considers three variants of the TMR approach. For convenience, they are identified using the name convention employed in (NIKNAHAD, 2012). Granularity levels of redundancy can be classified into coarse grain and fine grain. Coarse grain TMR (CGTMR) is the same as the conventional modular TMR, where a module is considered to be the entire design and the modules' outputs are voted. The CGTMR is suitable for intellectual property (IP) modules that are bought from third-parties and cannot be edited by the final user, for instance. The fine grain TMR (FGTMR) replicates sub-modules instead of the whole design. One case is the replication of sequential gates while the combinational paths remain the same. Also, a single voter is placed at the outputs of triplicated sequential gates. This approach is known as fine grain local TMR (FGLTMR). However, the trend for combinational logic in advanced technology nodes is to be more susceptible to radiation effects. Thus, a more effective approach is the fine grain distributed TMR (FGDTMR), where both combinational paths and sequential gates are replicated. Voters are also triplicated with the aim of reducing the amount of single point of failures. In this case, voters can be placed at the outputs of the sequential gates, and the final outputs, but also at the end of some combinational paths according to the need for the designer. Figure 4.4 illustrates a simple circuit and its resulting TMR implementations.

In this work, TMR circuits are implemented using Cadence Genus synthesis tool. The TMRi library was developed based on the design information hierarchy and built-in commands of the aforementioned tool (CADENCE, 2016). Tcl scripts were used for the development of the TMRi library because Tcl is commonly used in automatic synthesis tools, well documented and very intuitive (OUSTERHOUT et al., 2010).

The devised TMRi library provides three commands to implement each TMR version with the following syntax:

- TMRi_CGTMR
  <top_level_design_name>
  <top_level_cgtmr_design_name>
  <voter_design_name>
- TMRi_FGLTMR
  <top_level_design_name>
  <top_level_fgltmr_design_name>
  <voter_design_name>

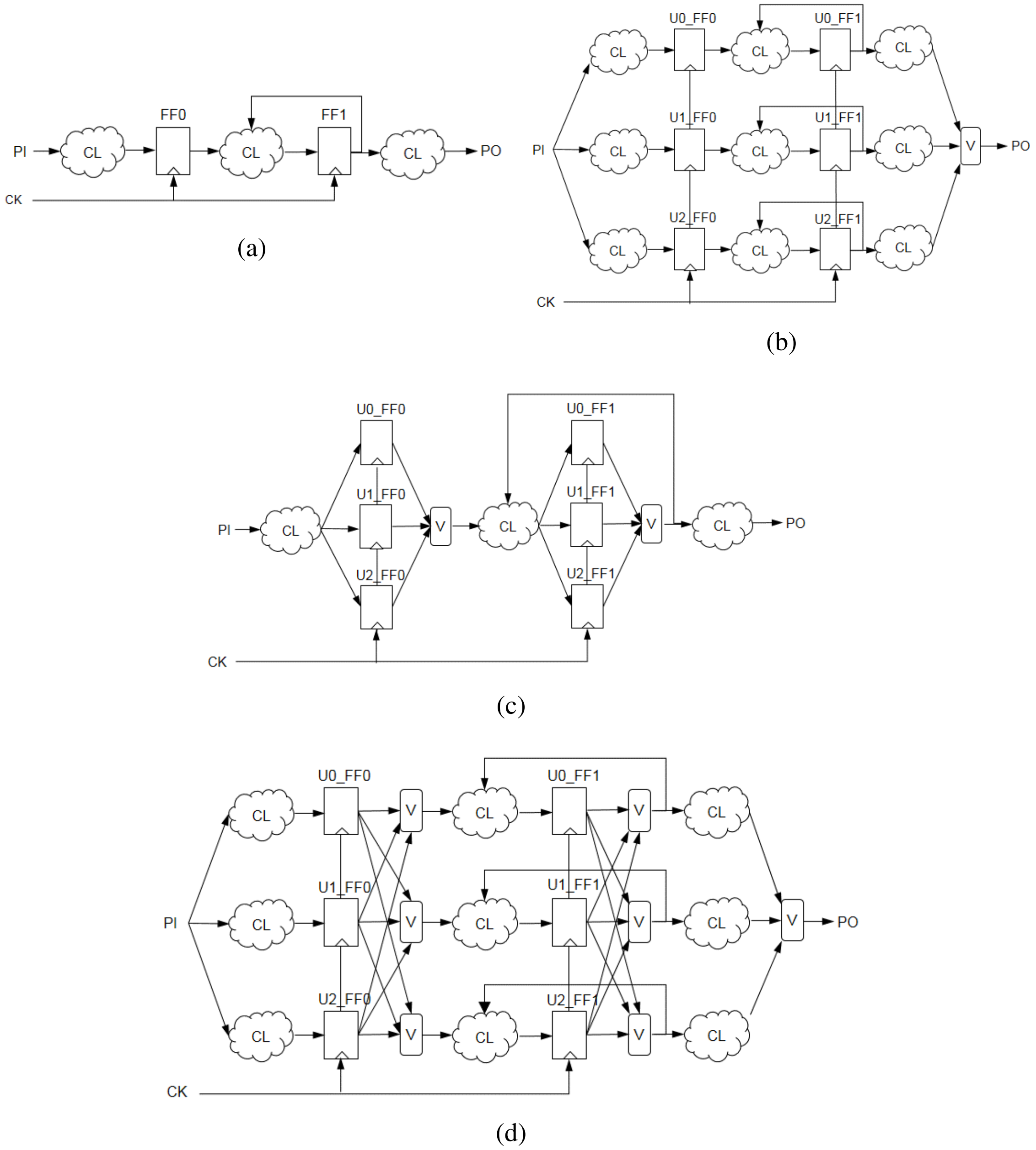
- TMRi_FGDTMR

  <top_level_design_name>

  <top_level_fgdtmr_design_name>

  <voter_design_name>

Figure 4.4: Illustration of a a) simple circuit and its TMR implementations: b) CGTMR, c) FGLTMR and d) FGDTMR



(a)

(b)

(c)

(d)

Source: From the author

The previous commands affect the circuit defined by the first argument. However, in the case of the designer only need to apply the TMR technique over an internal block, it must be extracted from the entire design and apply the command directly to it. Furthermore, a customized design of the voter circuit can be employed. The voter circuit has to be synthesized previously and loaded along with the target design.

The automatic TMR implementation approach employs a structural netlist, the used technology library, the TMRi library, and a voter HDL. According to the custom-design approach, the netlist corresponds to a gate-level netlist (mapped to standard cells) or a generic logic netlist (primitive gates) for the cell-based and FPGA flows respectively. This differentiation is because the implementation of combinational logic is based on LUTs in FPGA designs instead of logic standard cells. By default a generic HDL voter is available and it is included in the TMRi library if an user-customized voter is not utilized.

In this work, the implemented TMR circuit has identical number and type of ports with the aim of avoiding interface mismatches when the circuit is part of an existing system. Thereby, the proposed methodology can protect selectively critical modules in complex systems. Clock and asynchronous reset lines are also not replicated. The main issue of clock replication is clock skew among the redundant clock domains. Also, the synchronous design approach is violated in paths containing voters because of the presence of flip-flops with multiple clock domains. Thus, synchronization among replicas may be cumbersome, affecting substantially the design process (MELANY; LABEL; PELLISH, 2016). Synchronous designs are also characterized by the use of a single-master reset line in order to put the whole circuit into a known state. Furthermore, the implementation and treatment of these signal lines are performed in the back-end stage, where buffers are added to meet timing constraints. So, those elements are available after the synthesis stage and it skips from the scope of this work.

Some library cells include sequential gates with two complementary output pins. Corruption of the stored value in a sequential gate affects both normal and complementary signals because they are typically extracted from the same loop that holds the stored value. Consequently, avoiding the use of the complementary output by the synthesis tool reduces the number of voters in the implemented TMR circuit.

In order to avoid the accumulation of faults in enabled flip-flops, voters are placed before their feedback node. Hence, the transitions of FSMs built with this type of flip-flops are not corrupted by single faults. The feedback node of the enabled flip-flop must be external to the flip-flop's cell in order to implement adequately this feature.

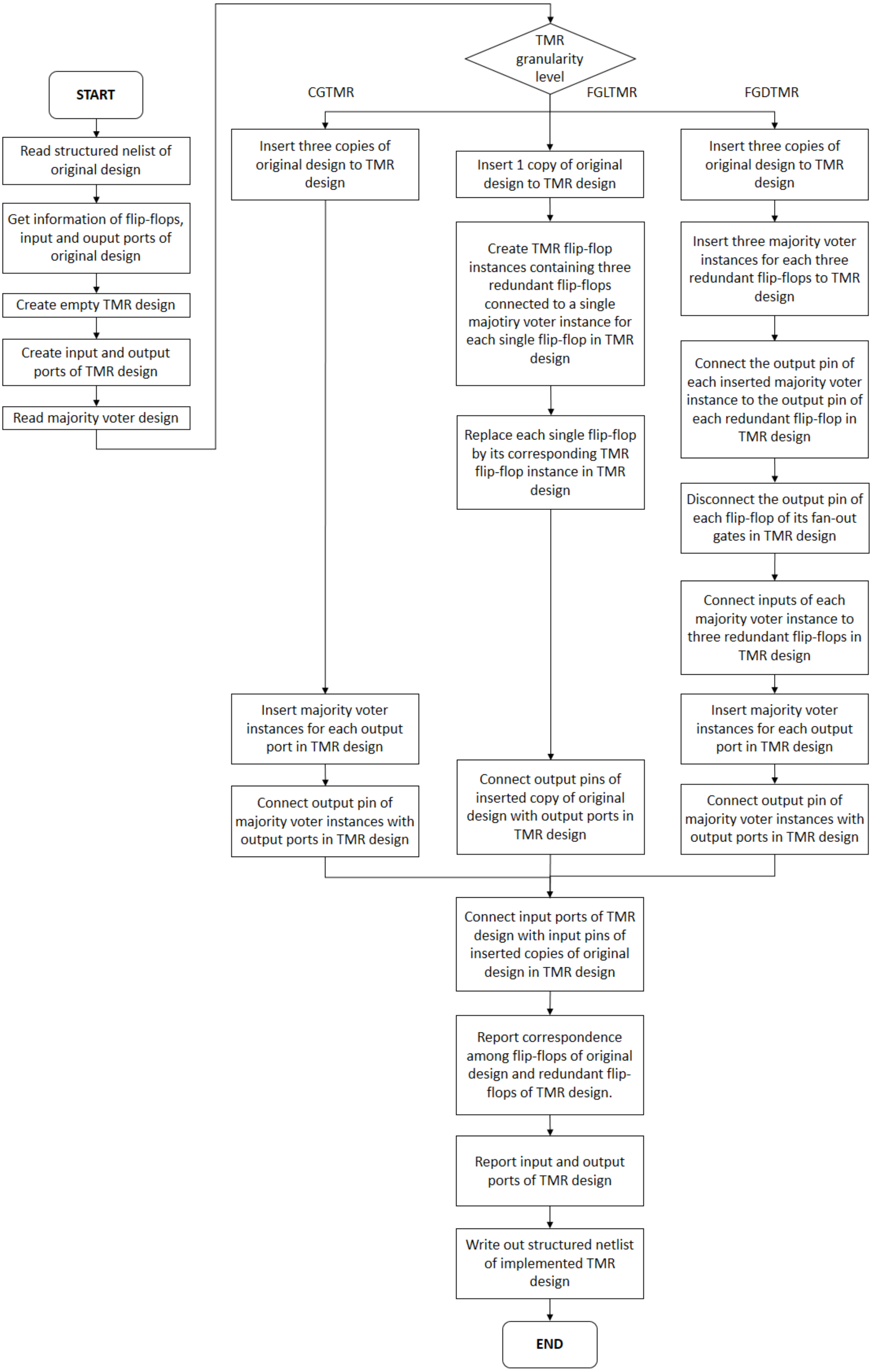
Furthermore, the entry design can contain undefined blocks or black boxes. However, they are replicated in the same way as the combinational logic according to the chosen granularity of the TMR implementation. For instance, this feature is employed when the unmitigated design contains dedicated data paths, multipliers and memory arrays.

The internal steps performed by the use of the aforementioned commands are explained next. Alike the three commands, it is extracted information about input and output ports, and sequential instances (flip-flops) from the post-synthesis netlist. Using the same environment is created a new empty design that will be the container of the desired TMR design. Then, the same number and name of input and output ports of the original circuit are assigned to the TMR design. According to the applied command, one or three instances of the original circuit are inserted within the TMR design. CGTMR and FGDTMR approaches employ three instances to provide both combinational paths and sequential gates in triplicate. FGLTMR version utilizes only one because combinational paths are non-redundant and sequential gates are afterward handled for replication. The following step is the insertion of voters that is performed in different ways for the TMR approaches as commented before. In the FGLTMR version, each sequential gate is replaced by three identical gates of the original thereof and a single voter. Moreover, it has the same input and output pins in order to do not alter connectivity among the gates. For the FGDTMR, triplicated voters are inserted for the existing triplicated sequential elements. CGTMR does not insert voters for the sequential gates. Additionally, single voting elements are collocated at the output pins of the inserted instances of the original circuit for the CGTMR and FGDTMR approaches. For the FGLTMR, there is no need to insert voters for this purpose. Afterward, the input ports of the TMR design are linked with the inputs of the inserted instances. Output ports are connected to the output pins of the inserted instance's outputs or voters according to the TMR approach.

Eventually, the TMR netlist is generated along with information about the correspondence of triplicated sequential gates in the TMR circuit with their original counterparts in the unmitigated circuit. Furthermore, input and output ports are also listed. This information is employed in the verification process of the TMR design.

The aforementioned steps to implements TMR circuits are depicted in Figure 4.5.

Figure 4.5: Flowchart to automate the implementation of TMR circuits


START
Read structured nelist of original design
Get information of flip-flops, input and ouput ports of original design
Create empty TMR design
Create input and output ports of TMR design
Read majority voter design
TMR granularity level
CGTMR
FGLTMR
FGDTMR
Insert three copies of original design to TMR design
Insert 1 copy of original design to TMR design
Create TMR flip-flop instances containing three redundant flip-flops connected to a single majotiry voter instance for each single flip-flop in TMR design
Replace each single flip-flop by its corresponding TMR flip-flop instance in TMR design
Insert three copies of original design to TMR design
Insert three majority voter instances for each three redundant flip-flops to TMR design
Connect the output pin of each inserted majority voter instance to the output pin of each redundant flip-flop in TMR design
Disconnect the output pin of each flip-flop of its fan-out gates in TMR design
Connect inputs of each majority voter instance to three redundant flip-flops in TMR design
Insert majority voter instances for each output port in TMR design
Connect output pin of majority voter instances with output ports in TMR design
Connect output pins of inserted copy of original design with output ports in TMR design
Insert majority voter instances for each output port in TMR design
Connect output pin of majority voter instances with output ports in TMR design
Connect input ports of TMR design with input pins of inserted copies of original design in TMR design
Report correspondence among flip-flops of original design and redundant flip-flops of TMR design.
Report input and output ports of TMR design
Write out structured netlist of implemented TMR design
END

Source: From the author

4.4.2 TMR optimization Process (TMRo)

The use of TMR increases output capacitance and fan-out of gates. Furthermore, the timing of the paths is altered by the voter's delays. In the case of encountering timing violations in the TMR design or with the aim of improving performance, further optimization based on gate sizing is employed. The circuit is easily constrained for this purpose in order to preserve all the gates. On the other hand, if timing issues are not yet resolved after applying optimization, the clock frequency must be redefined. An additional delay budget may be considered in the initial synthesis process to manage this issue.

The circuit is synthesized again in order to perform the sizing of the design. An optimized TMR netlist is generated. This step is only performed for the cell-based flow because the FPGA flow employs generic logic primitives without information of area, timing, and power.

4.4.3 TMR verification process (TMRv)

The optimized TMR implementation must be in compliance with the original circuit functionality. Moreover, it must be proved its capability to mitigate single faults into one copy of the replicas. In other words, redundant logic and voter elements are expected to be successfully inserted. The behavior of a system in presence of faults can be observed inserting intentionally faults into the system and monitoring the responses. That approach is known as fault injection and permits to evaluate the dependability of a system. According to the design stage where is performed the fault injection, it can be classified in simulation-based, emulation-based, and hardware fault injection. At early design stages (before fabrication) is only possible to use the simulation-based approach.

The simulated-based fault injection brings full observation an control of the netlist. However, it needs the use of an exhaustive testbench that can leads to prohibitive long simulation times and big resource utilization. Unlike executing simulations, a powerful approach is the use of formal verification. Formal verification is employed to check as long as the current circuit implementation will function as previously specified at some higher level of abstraction. Equivalence checking is a type of formal verification and it consists in the Boolean comparison of pairs of combinational paths of two different implementations of a design. Combinational paths can be bounded by memory elements (flip-flops) and ports. In the simplest case, it is expected a one-to-one correspondence between these boundary elements for both designs under comparison. All pairs of corresponding next-state logic functions and output functions must be proved to be functionally equal in presence of the same set of stimuli at the corresponding flip-flops and primary inputs. For instance, equivalence checking is performed to verify the correctness of the resulting post-synthesis netlist in relation to the RTL design during the front-end stage.

Some works consider equivalence checking to validate TMR designs. Authors in (BERG; LABEL, 2016) propose a verification method employing gate to gate tracking in order to verify both functional and topology correctness of TMR implementations. in (BELTRAME, 2015), the author uses the equivalence checking to perform fault injection campaigns but without performing gate-level simulations. In this work, the TMR netlist is implemented directly from a post-synthesis netlist so, evaluation of the equivalence between these two designs is simplified because the matching process of registers and primary ports is done by naming convention. The verification process comprises the analysis of the correct functionality and fault tolerance of the automatically generated TMR circuit. A library was developed to perform TMR verification (TMRv library) for the Cadence's Genus and Logic Equivalence Checking (LEC) tools (CADENCE, 2013). Likewise the TMRi library, TMRv library is based on Tcl scripting files.

Early analysis falls in verifying the presence of triplicated flip-flops into the resulting TMR design. To do that, an iterative search is performed and it is reported the number of matched TMR flip-flops. Furthermore, the correctness of redundant combinational logic must be checked also. Combinational logic may represent next-state or output logic functions. In the first case, replication must not share any gate in order to preserve the isolation among the redundant paths as depicted in Figure 4.6. Whereas redundant combinational logic paths that drive output ports have to share uniquely the voter element at the end of the path (Figure 4.7). The amount of common gates encountered between redundant logic paths is reported for each case.

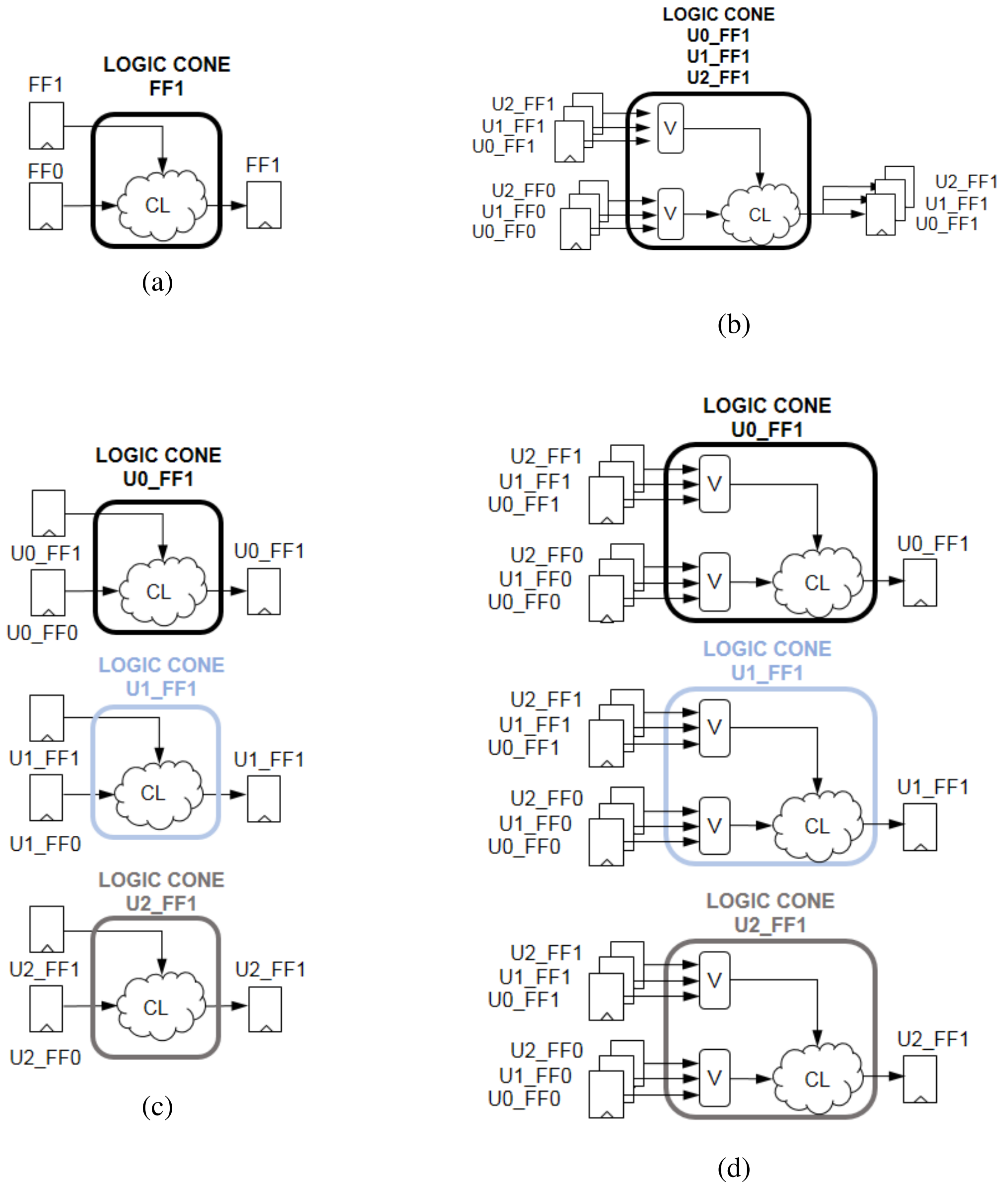
In the aforementioned steps, no logic analysis is done. As a consequence, they are executed using the Genus synthesis tool because of the simplicity to access the circuit's structure information instead of using the equivalence checking tool, which reduces the circuits to their minimum logical forms.

The following three commands belong to the TMRv library and they are employed to verify the presence of TMR flip-flops and the independence of redundant combinational paths that drives flip-flops and output ports respectively:

- TMRv_replication_ff <tmr_design_flipflop_list>
- TMRv_replication_cmb_logic_ff <tmr_design_flipflop_list>
- TMRv_replication_cmb_logic_po <tmr_design_output_port_list>

The arguments tmr_design_flipflop_list and tmr_design_output_port_list receives the target flip-flops and output ports of the TMR circuit intended for verification. They are obtained in the TMRi step.

Figure 4.6: Logic cones of a) unmitigated flip-flop and the resulting TMR logic cones: b) FGLTMR, c) CGTMR and d) FGDTMR



(a)

(b)

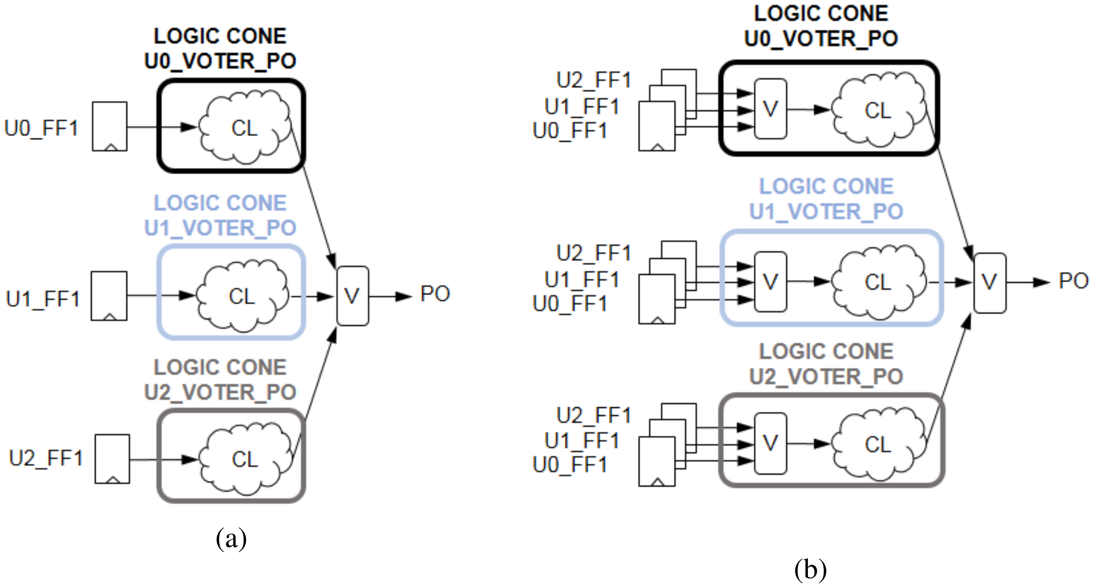
(c)

(d)

Source: From the author

The procedure to find the elements of a logic cone for a chosen flip-flop is illustrated after in Figure 4.9. Furthermore, the structural verification process is shown in

Figure 4.7: Triplicated combinational logic paths of primary output in a) CGTMR and b) FGDTMR implementations respectively



Source: From the author

Figure 4.10 and Figure 4.11 for redundant combinational paths of flip-flops and primary output ports respectively.

Once the structural verification of the replicas is validated, the synthesis tool is left and from this point, the equivalence checking tool is employed. The equivalence checking uses the unmitigated (golden) and the target TMR (revised) designs, the technology library, the TMRv library, and also the information about the matching flip-flops and ports between the golden and the revised designs.

In order to make clear the logic equivalence process, the simple circuit and its TMR implementation depicted in Figure 4.4 are used as an example. Let us take the flip-flop FF1 as a reference. It loads its data from a combinational logic path (also called logic cone), which is stimulated by two flip-flops (FF0 and FF1). Figure 4.6 illustrates the replicated logic cones in the different TMR implementations.

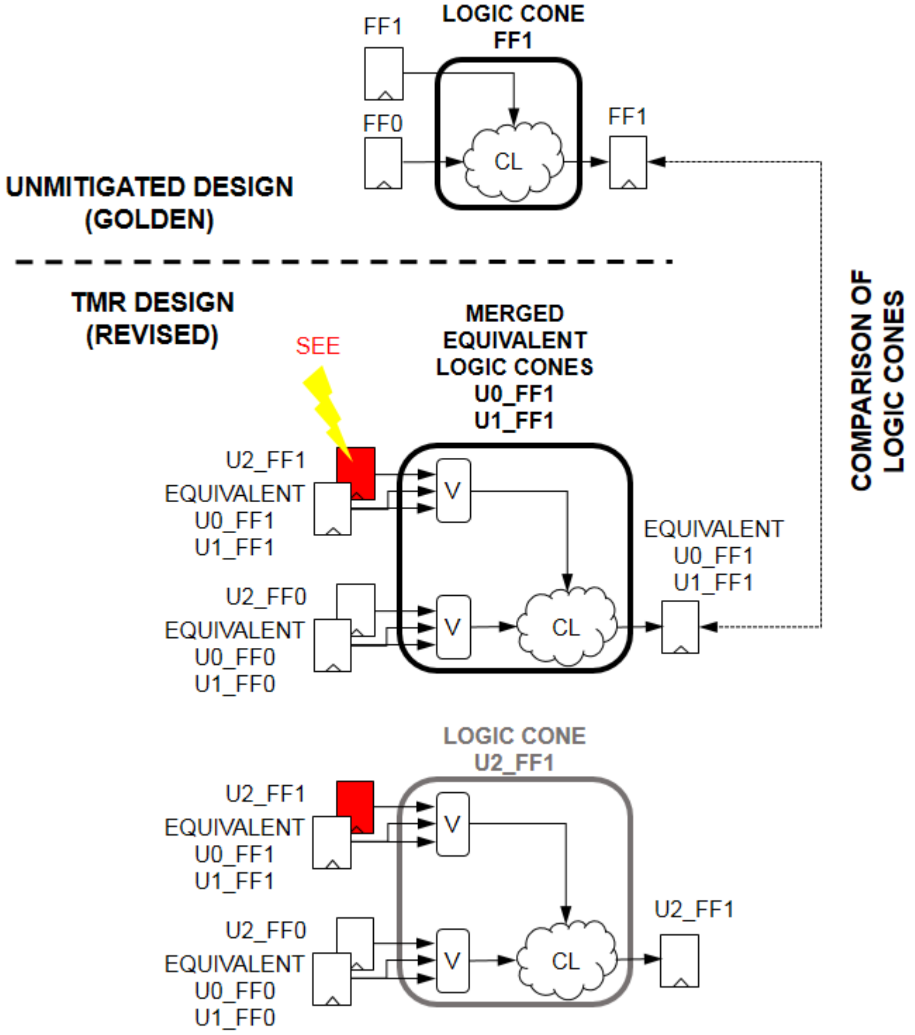
In order to validate the correct implementation of designs implementing fine grain redundancy, the start point is to prove that triplicated sequential gates in the TMR design are driven by equivalent combinational paths. Each one of the combinational logic paths in the FGTMR designs are composed of the original logic path and one or more voters. Moreover, each replicated path is stimulated by the same set of flip-flops and input ports. In the example, each redundant combinational paths is fed by the TMR flip-flops U0_FF0, U1_FF0, and U2_FF0 and U0_FF1, U1_FF1, and U2_FF1. Figure 4.6b and Figure 4.6d depict the resulting logic cones for each TMR flip-flop in fine grain approaches. Three logic cones corresponding to each TMR flip-flop must be proved to be equivalent. In case of mismatch, the process of verification must be interrupted and the revised design must be checked. Debugging begins verifying that redundant combinational paths have the same set of inputs. If there are no missing inputs among the replicas, logic cones must be fixed before going on to the next step.

Once replicated logic cones in the revised design are proved to be equivalent, the unmitigated and TMR designs are compared. This step pursues to verify that the resulting TMR implementation performs the same functionality that the original circuit. Moreover, it must be tested the fault tolerance against single faults of the revised design. Hence, the presence of soft-errors in flip-flops are modeled to validate the effectiveness of the TMR implementation. The fault model considered during the equivalence checking consists in setting the same output values for two of the triplicated sequential gates while the other one can take any value. Because there are only two valid logic values during the equivalence checking ("1" or "0"), the fault model represents effectively the behavior of an SEU and an SET latched by a flip-flop, where the location of the fault is determined by the redundant flip-flop with opposite value to the other two replicas. There is no need to specify the time when the fault occurs because if the TMR technique is properly implemented the single fault must be masked. In consequence, the fault is not propagated to the next logic state and no error is manifested. On the other side, if the triplicated flip-flops have the same output value, the circuit acts free of faults and therefore, boolean equivalence of both unmitigated and TMR circuits must be achieved as well. Also, in case of no equivalence, correspondence of key points between the golden and the revised design must be observed. If there are no issues of this type, logic cones between designs must be fixed.

Figure 4.8 depicts an SEU in the U2_FF1 flip-flop and the comparison process between the unmitigated and its TMR implementation. The logic cones of two out of the triplicated flip-flops of TMR implementation are merged in the comparison process. That is valid and not lost generality because they take the same value and the equivalence between its logic cones was already verified previously.

Unlike fine grain implementations, the coarse grain version does not link sequential gates of the replicas with voters. Otherwise, it only uses voters for primary outputs. So, each logic cone for a sequential element does not share necessarily the same key points. The logical equivalence between replicated logic cones in the revised design is verified setting the same stimulus for two of the triplicated sequential gates and compar-

Figure 4.8: Comparing process and fault injection for the FGDTMR implementation



Source: From the author

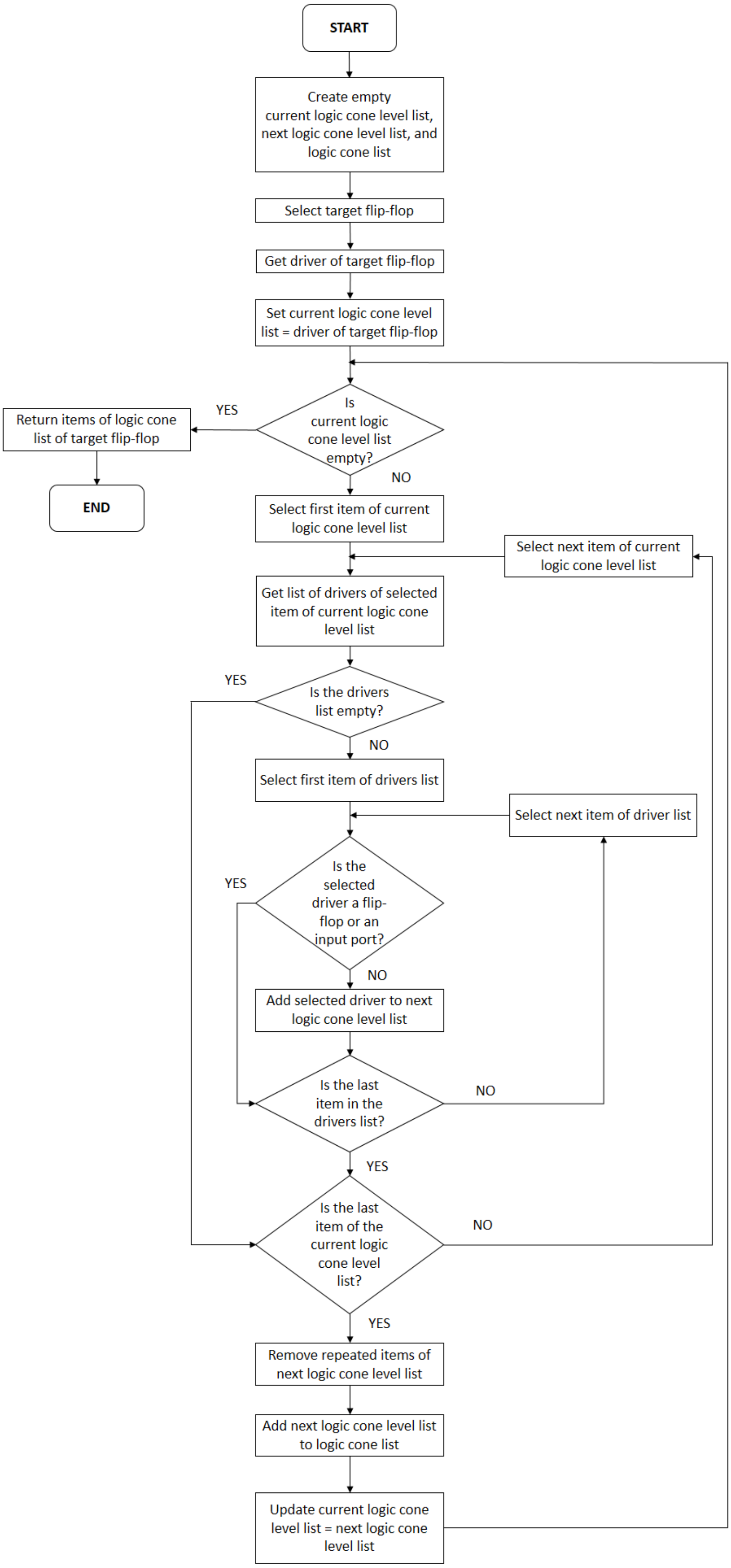
ing the results in its respectively end key points. In case of mismatching, correspondence of key points is analyzed. If the key point mapping is correct, logic cones must be fixed. Equivalence between the revised and golden designs and fault tolerance is verified using the same approach explained before.

The following commands are utilized for the equivalence checking and the verification of fault tolerant capability for each one of the implemented TMR versions according to the selected granularity respectively:

- TMRv_lec_CGTMR
- TMRv_lec_FGLTMR
- TMRv_lec_FGDTMR

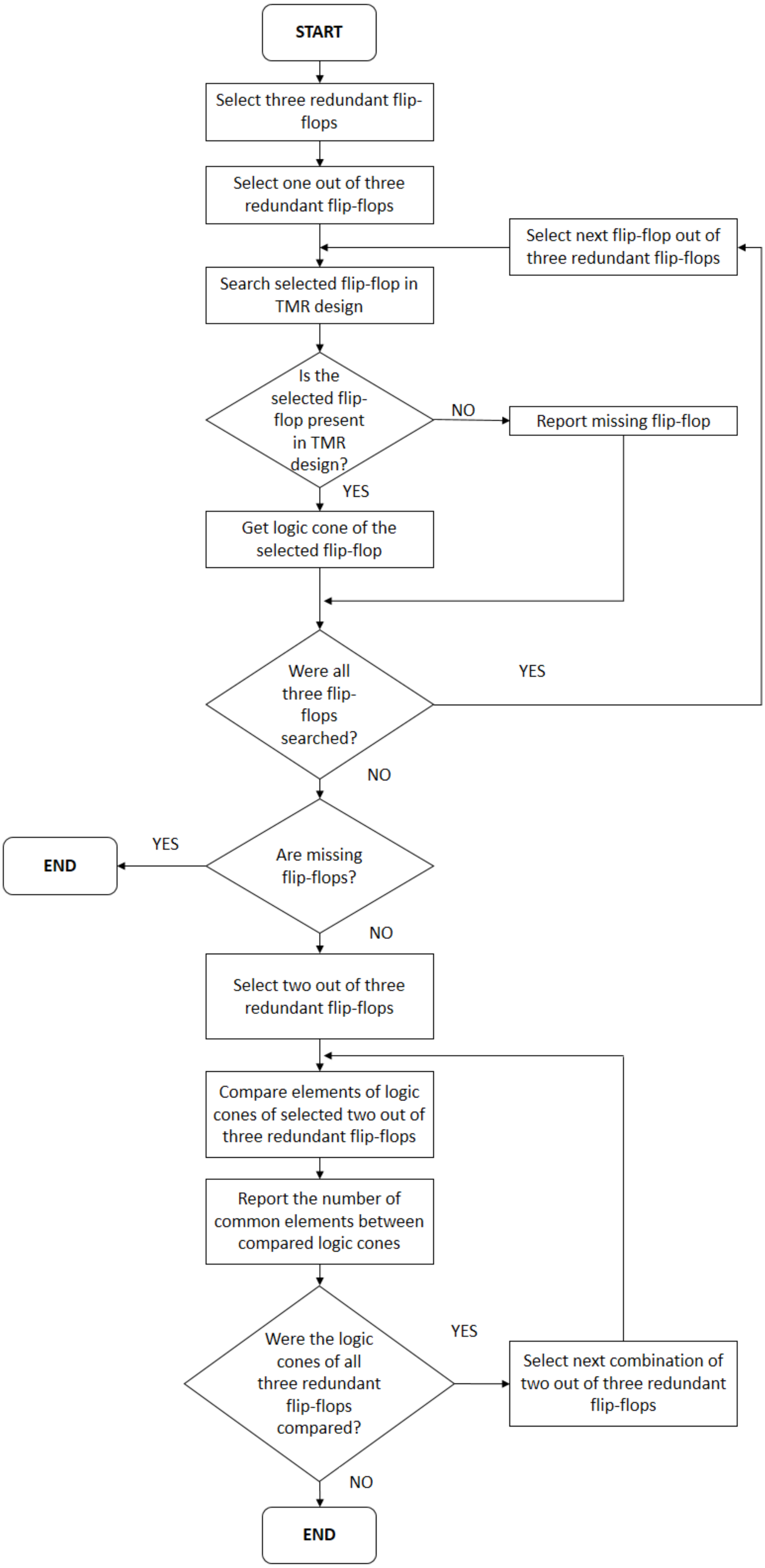
Figure 4.12 illustrates the verification process of the correct functionality and the fault-tolerant capability between the unmitigated circuit and its TMR counterpart.

Figure 4.9: Flowchart to get elements of logic cone of a selected flip-flop


START
Create empty current logic cone level list, next logic cone level list, and logic cone list
Select target flip-flop
Get driver of target flip-flop
Set current logic cone level list = driver of target flip-flop
Is current logic cone level list empty?
YES
Return items of logic cone list of target flip-flop
END
NO
Select first item of current logic cone level list
Select next item of current logic cone level list
Get list of drivers of selected item of current logic cone level list
Is the drivers list empty?
YES
NO
Select first item of drivers list
Select next item of driver list
Is the selected driver a flip-flop or an input port?
YES
NO
Add selected driver to next logic cone level list
Is the last item in the drivers list?
NO
YES
Is the last item of the current logic cone level list?
NO
YES
Remove repeated items of next logic cone level list
Add next logic cone level list to logic cone list
Update current logic cone level list = next logic cone level list

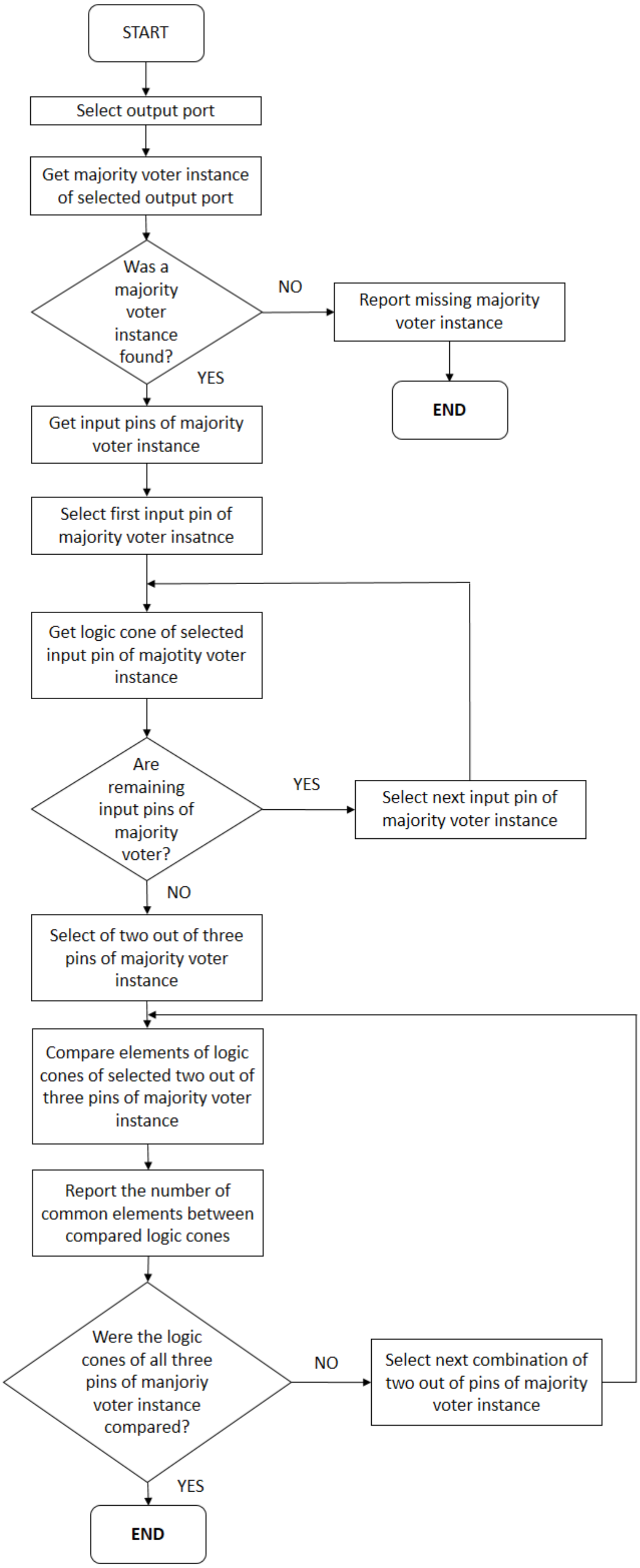
Source: From the author

Figure 4.10: Flowchart to find shared elements in logic cones of three redundant flip-flops


START
Select three redundant flip-flops
Select one out of three redundant flip-flops
Select next flip-flop out of three redundant flip-flops
Search selected flip-flop in TMR design
Is the selected flip-flop present in TMR design?
NO
Report missing flip-flop
YES
Get logic cone of the selected flip-flop
Were all three flip-flops searched?
YES
NO
Are missing flip-flops?
YES
END
NO
Select two out of three redundant flip-flops
Compare elements of logic cones of selected two out of three redundant flip-flops
Report the number of common elements between compared logic cones
Were the logic cones of all three redundant flip-flops compared?
YES
Select next combination of two out of three redundant flip-flops
NO
END

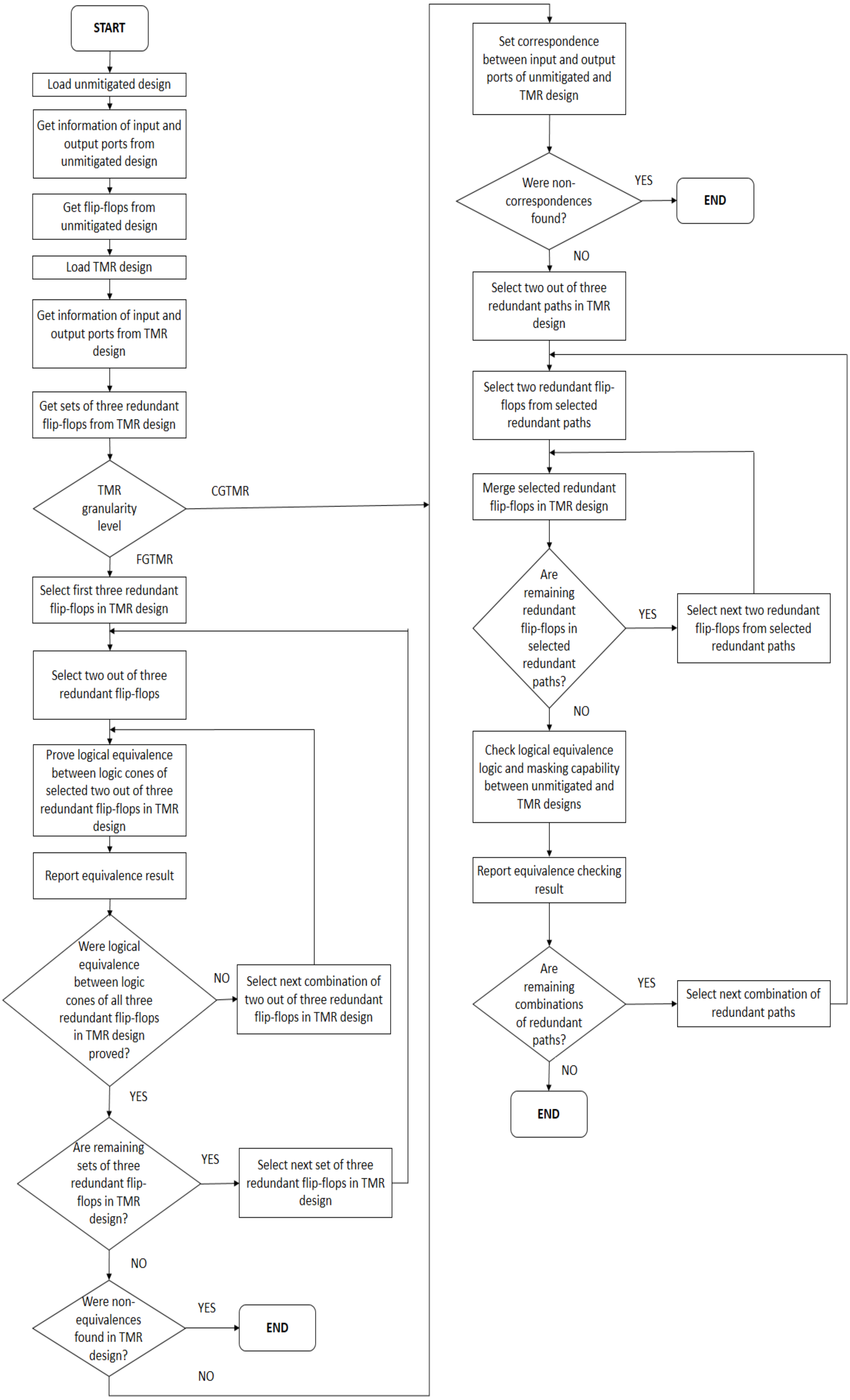
Source: From the author

Figure 4.11: Flowchart to find shared elements in logic cones of a selected primary output


START
Select output port
Get majority voter instance of selected output port
Was a majority voter instance found?
NO
Report missing majority voter instance
END
YES
Get input pins of majority voter instance
Select first input pin of majority voter insatnce
Get logic cone of selected input pin of majotity voter instance
Are remaining input pins of majority voter?
YES
Select next input pin of majority voter instance
NO
Select of two out of three pins of majority voter instance
Compare elements of logic cones of selected two out of three pins of majority voter instance
Report the number of common elements between compared logic cones
Were the logic cones of all three pins of manjoriy voter instance compared?
NO
Select next combination of two out of pins of majority voter instance
YES
END

Source: From the author

Figure 4.12: Flowchart to verify the correct functionality and masking capability of the implemented TMR design


START
Load unmitigated design
Get information of input and output ports from unmitigated design
Get flip-flops from unmitigated design
Load TMR design
Get information of input and output ports from TMR design
Get sets of three redundant flip-flops from TMR design
TMR granularity level
CGTMR
FGTMR
Select first three redundant flip-flops in TMR design
Select two out of three redundant flip-flops
Prove logical equivalence between logic cones of selected two out of three redundant flip-flops in TMR design
Report equivalence result
Were logical equivalence between logic cones of all three redundant flip-flops in TMR design proved?
NO
Select next combination of two out of three redundant flip-flops in TMR design
YES
Are remaining sets of three redundant flip-flops in TMR design?
YES
Select next set of three redundant flip-flops in TMR design
NO
Were non-equivalences found in TMR design?
YES
END
NO
Set correspondence between input and output ports of unmitigated and TMR design
Were non-correspondences found?
YES
END
NO
Select two out of three redundant paths in TMR design
Select two redundant flip-flops from selected redundant paths
Merge selected redundant flip-flops in TMR design
Are remaining redundant flip-flops in selected redundant paths?
YES
Select next two redundant flip-flops from selected redundant paths
NO
Check logical equivalence logic and masking capability between unmitigated and TMR designs
Report equivalence checking result
Are remaining combinations of redundant paths?
YES
Select next combination of redundant paths
NO
END

Source: From the author

4.5 Conclusion

In this chapter was explained the steps of the proposed methodology to automated the design of TMR circuits. The adaptation of a commercial design flow was the goal of the present work in order to ease the design of fault-tolerant circuits. Also, the proposed methodology does not interfere with the use of a high-level description of the circuit and speeds up the design of fault-tolerant circuits. TMR circuits with a single clock and reset domains are automatically implemented in order to reduce the design effort and to reuse designs initially not addressed for critical applications where the effect of radiation is a concern. Moreover, the proposed methodology is not restricted to a specific technology node. That feature is worthy of taking into account the constant miniaturization of the transistor. Furthermore, the proposed methodology deals with the semi-custom design of digital circuits, so TMR circuits can be implemented following the cell-based and FPGA approaches from a mapping netlist or using generic primitive gates respectively. Optimization of the resulting TMR circuit is also applied based on the gate sizing approach with the aim of resolving timing issues. Verification of the implemented TMR circuit is a relevant step in order to detect issues early before the deployment of the circuit. Because tools are also prone to error, the verification step pursues to ensure the correct functionality and fault-tolerant capability of the resulting TMR design.

5 EXPERIMENTS

5.1 Introduction

The aim of this section is to show how the design of fault-tolerant circuits is simplified by the use of the proposed automated methodology. The following guidelines were taken into account to select the circuits intended for experimentation:

- Large-size circuits, where manual insertion of redundancies would be a hard, inefficient and prone to error task.
- Circuits generated from behavioral languages, where the resulting optimized RTL description is generated entirely by the synthesis tool.
- Circuits with restricted information about its implementation or where circuit description is not easily readable.

As a result, two study-case circuits corresponding to a matrix multiplication and a soft-core microprocessor were chosen. They are used to follow the proposed cell-based and the FPGA flows respectively shown in the previous chapter.

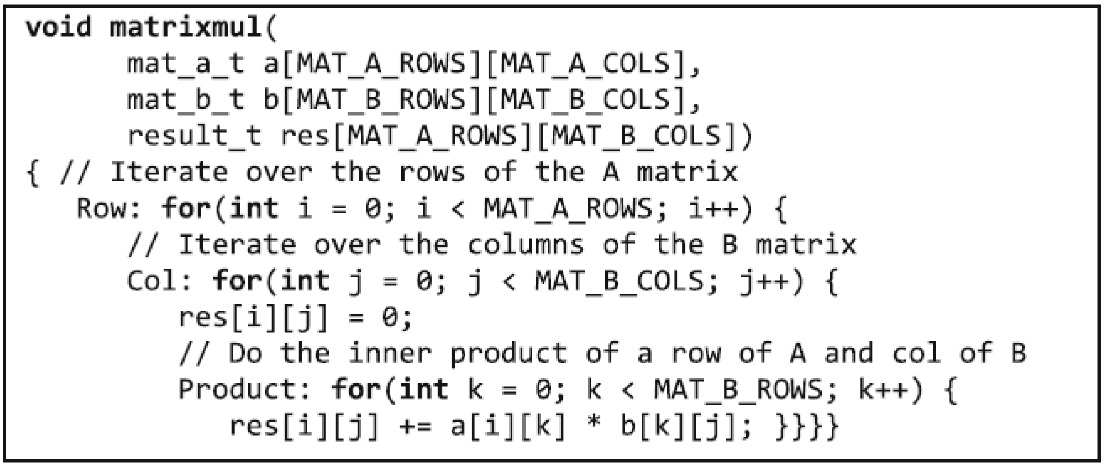
5.2 Fault-tolerant cell-based design flow

The case-study design for the cell-based design flow corresponds to a matrix multiplication. Matrix multiplication circuit is a hardware-implemented algorithm essentially built from blocks of multipliers and adders. These mathematical operations are very common in digital signal processing (DSP) applications and its recurring structure enables the exploration of different implementations. It is considered for the experiments that each input matrix is a 6x6 array of 8-bit vectors and the output thereof is also a 6x6 array composed of 19-bit vectors.

Three architectural versions were generated using the high-level synthesis approach. The Xilinx Vivado HLS tool was used for that purpose. Figure 5.1 shows the C algorithm used to generate the three RTL implementations.

The architectural exploration was based on pipelining the loop iterations in the C code. The first version (MM1) was generated without using optimization directives that lead to a non-pipelining version with a latency of 733 clock cycles. Meanwhile, two optimized versions (MM2 and MM3) were implemented utilizing the pipeline directive.

Figure 5.1: Matrix multiplication algorithm



```
void matrixmul(
      mat_a_t a[MAT_A_ROWS][MAT_A_COLS],
      mat_b_t b[MAT_B_ROWS][MAT_B_COLS],
      result_t res[MAT_A_ROWS][MAT_B_COLS])
{ // Iterate over the rows of the A matrix
   Row: for(int i = 0; i < MAT_A_ROWS; i++) {
      // Iterate over the columns of the B matrix
      Col: for(int j = 0; j < MAT_B_COLS; j++) {
         res[i][j] = 0;
         // Do the inner product of a row of A and col of B
         Product: for(int k = 0; k < MAT_B_ROWS; k++) {
            res[i][j] += a[i][k] * b[k][j]; }}}}
```

Source: (TONFAT et al., 2016)

MM2 is a 21-stage pipeline with a latency of 112 clock cycles, meanwhile, MM3 exhibits a 7-stage pipeline an latency of 133 clock cycles.

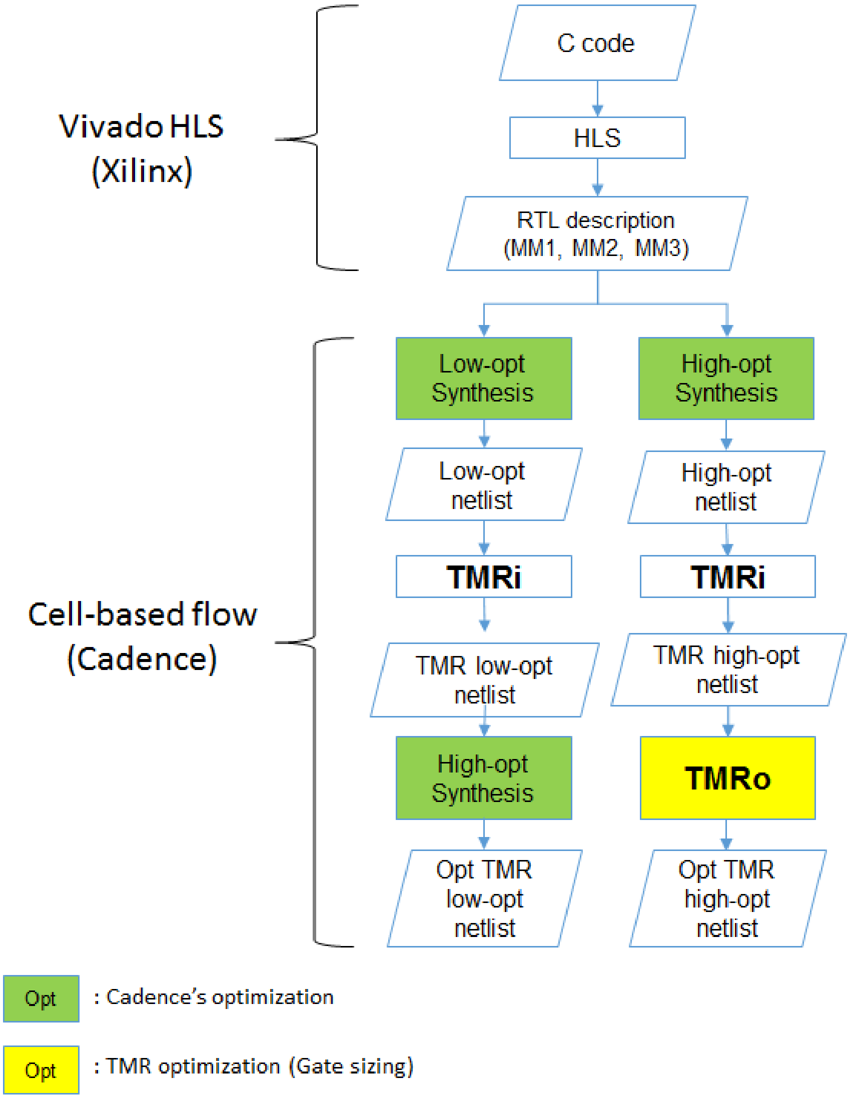
These three versions of the matrix multiplication algorithm were employed to follow the cell-based flow of the proposed methodology. They were synthesized using the NanGate 45nm Open Cell Library and setting the target clock frequency to 125 MHz. The area of each implementation is expressed in terms of the NAND2 Gate Equivalent (GE) size of the technology library and it is not considered interconnection's area and delay at this stage.

In order to analyze the proposed TMR cell-based flow, two approaches were followed after the generation of the RTL implementations as depicted in Figure 5.2. The left branch is followed to evaluate the resulting TMR circuits from a low-optimized synthesis. Further aggressive optimization is performed after the TMR implementation. On the other side, in the right branch is applied the TMR in high-optimized versions of the circuits and afterward, gate sizing optimization is employed.

5.2.1 Analyzing the impact of logic optimization in the designs

Each one of the three RTL descriptions was synthesized setting the Genus synthesis tool to perform low- and high-effort levels of logic optimization. In the case of low-effort synthesis, very little RTL optimization such as constant propagation, resource sharing, logic speculation, MUX optimization and carry-save arithmetic (CSA) optimization is performed. Also, relaxed redundancy identification and removal is done. As a

Figure 5.2: Procedure to evaluate the proposed cell-based flow to design TMR circuits



Source: From the author

result, the post-synthesis netlist retains the behavioral structures of the entry RTL description. On the other hand, high-effort approach performs aggressive utilization of the aforementioned optimization tasks. Table 5.1 summarizes the resulting designs after synthesis.

In the case of the high-optimized implementation of MM1, the number of combinational gates suffers a considerable reduction of 24% and an insignificant variation in the number of flip-flops (1%) in comparison to its low-optimized implementation. Therefore, the area comes down by 8%. Besides, its maximum clock frequency is reduced by 32%. This happens because area optimization is obtained by the use of small, slow gates. High-optimized implementations of MM2 and MM3 exhibit similar results in the diminution of the number of combinational gates (16% and 13% respectively), flip-flops

Table 5.1: Summary of low and high effort synthesis results of circuit versions

Design	**MM1**	**MM2**	**MM3**
Low-optimized implementations			
Combinational gates	523 (1×)	9180 (1×)	1981 (1×)
Flip-flops	73 (1×)	722 (1×)	209 (1×)
Area (GE)	1242.67 (1×)	21592.00 (1×)	4805.67 (1×)
Max. Freq (MHz)	223 (1×)	163 (1×)	131 (1×)
High-optimized implementations			
Combinational gates	395 (0.76×)	7735 (0.84×)	1715 (0.87×)
Flip-flops	72 (0.99×)	709 (0.98×)	206 (0.99×)
Area (GE)	1144.33 (0.92×)	20515.00 (0.95×)	4607.00 (0.96×)
Max. Freq (MHz)	153 (0.68×)	147 (0.90×)	129 (0.98×)

Source: From the author

(2% and 1% respectively) and area (5% and 4% respectively). Maximum clock frequency also decreased by 10% and 2% respectively.

Low-optimized implementation of MM1 leads to a maximum clock frequency greater than the values for MM2 and MM3 (1.4 and 1.7 times respectively). For that reason, performance degradation is more noticeable for MM1.

5.2.2 Analyzing the impact of adding TMR in area and performance

The left sections of Table 5.2 and Table 5.3 show the characteristics of the generated TMR implementations for the low and high-optimized versions respectively. Because the CGTMR triplicates all the elements of the circuit, the resulting area of the replicated elements is 3 times the area of the unmitigated circuit. Likewise the CGTMR, FGDTMR also triplicates all the internal elements of the unmitigated circuit. In the FGLTMR, by contrast, it is only triplicated the flip-flops leaving the combinational logic unaltered. For the low-optimized implementations of MM1, MM2, and MM3, the area overhead caused by the addition of voters at the output ports is 19%, 1%, and 6% respectively. Performance is not affected for CGTMR versions. FGTMR approaches, in contrast, are slightly reduced in 8%, 1%, and 5% respectively. In the case of high-optimized implementations, the area overhead related to the insertion of voters at the output ports is 21%, 2% and 7% for MM1, MM2, and MM3 respectively. Furthermore, the increase of area because of the addition of single voters per each triplicated flip-flop is 33%, 18%, and 24% respectively in both cases. About performance, CGTMR does not suffer impact meanwhile the FGTMR approaches only suffers a reduction of around 5%.

Table 5.2: Summary of TMR implementation results from low-optimized netlist (left section) and further aggressive optimization (right section)

Design	Unmitigated	CGTMR	FGLTMR	FGDTMR	Opt. CGTMR	Opt. FGLTRM	Opt. FGDTMR
Low-optimized implementation of MM1							
Combinational gates	523	1613	596	1832	945	402	402
Flip-flops	73	219	219	219	110	72	72
Area (GE)	1242.67 (**1**×)	3962.67 (**3.19**×)	2459.33 (**1.98**×)	5130.67 (**4.13**×)	2469.00 (**1.99**×)	1142.67 (**0.92**×)	1142.67 (**0.92**×)
Max. Freq (MHz)	223 (1×)	223 (1×)	205 (0.92×)	205 (0.92×)			
Low-optimized implementation of MM2							
Combinational gates	9180	27598	9902	29764	22815	7629	7625
Flip-flops	722	2166	2166	2166	2166	709	709
Area (GE)	21592.00 (**1**×)	65085.33 (**3.01**×)	33674.67 (**1.56**×)	76637.33 (**3.55**×)	61744.67 (**2.86**×)	20461.00 (**0.95**×)	20455.67 (**0.95**×)
Max. Freq (MHz)	163 (1×)	163 (1×)	149 (0.92×)	149 (0.91×)			
Low-optimized implementation of MM3							
Combinational gates	1981	6001	2190	6628	4141	1368	1369
Flip-flops	209	627	627	627	627	206	206
Area (GE)	4805.67 (**1**×)	14726.33 (**3.06**×)	8289.00 (**1.72**×)	18070.33 (**3.76**×)	13550.33 (**2.82**×)	441.33 (**0.92**×)	4442.00 (**0.92**×)
Max. Freq (MHz)	131 (1×)	131 (1×)	125 (0.95×)	125 (0.95×)			

Source: From the author

5.2.3 Analyzing the impact of performing logic optimization after adding TMR

The resulting TMR implementations are also optimized with the goal of evaluating how the synthesis tool treats the automatically inserted redundancies and its relation with the selected TMR granularity. The right sections of Table 5.2 and Table 5.3 show the characteristics of the optimized TMR implementations for the low and high-optimized versions respectively.

First, the TMR implementations obtained from the low-optimized circuits were optimized without using additional constraints. Thus, the synthesis is able to perform aggressive optimization. As a result, in the case of the coarse grain approach, flip-flops were removed partially in MM1 but they were not affected in MM2 and MM3. On the other hand, in the fine grain approaches, the tool excluded all the redundant flip-flops and even it generated post-synthesis circuits with the same number of flip-flops than their high-optimized implementations respectively. The fine grain approaches have voters as common points among the triplicated flip-flops. Therefore, the synthesis tool was able

Table 5.3: Summary of TMR implementation results from high-optimized netlist (left section) and further gate sizing optimization (right section)

Design	**Unmitigated**	**CGTMR**	**FGLTMR**	**FGDTMR**	**Opt.** **CGTMR**	**Opt.** **FGLTRM**	**Opt.** **FGDTMR**
High-optimized implementation of MM1							
Combinational gates	395	1229	467	1445	1265	467	1481
Flip-flops	72	216	216	216	216	216	216
Area (GE)	1144.33 (**1**×)	3667.67 (**3.21**×)	2344.33 (**2.05**×)	4819.67 (**4.21**×)	3691.67 (**3.23**×)	2344.33 (**2.05**×)	4843.67 (**4.23**×)
Max. Freq (MHz)	153 (1×)	153 (1×)	144 (0.94×)	144 (0.94×)	153 (1×)	144 (0.94×)	144 (0.94×)
High-optimized implementation of MM2							
Combinational gates	7735	23263	8444	25390	23331	8485	25581
Flip-flops	709	2127	2127	2127	2127	2127	2127
Area (GE)	20515.00 (**1**×)	61854.33 (**3.02**×)	32386.33 (**1.58**×)	73198.33 (**3.57**×)	61899.67 (**3.02**×)	32359.00 (**1.58**×)	73325.67 (**3.57**×)
Max. Freq (MHz)	147 (1×)	147 (1×)	140 (0.95×)	140 (0.95×)	147 (1×)	142 (0.97×)	141 (0.96×)
High-optimized implementation of MM3							
Combinational gates	1715	5203	1921	5821	5271	1925	5904
Flip-flops	206	618	618	618	618	618	618
Area (GE)	4607.00 (**1**×)	14130.33 (**3.07**×)	8040.33 (**1.75**×)	17426.33 (**3.78**×)	14175.67 (**3.08**×)	8046.33 (**1.75**×)	17492.67 (**3.80**×)
Max. Freq (MHz)	129 (1×)	129 (1×)	124 (.96×)	124 (0.96×)	129 (1×)	125 (0.97×)	125 (0.97×)

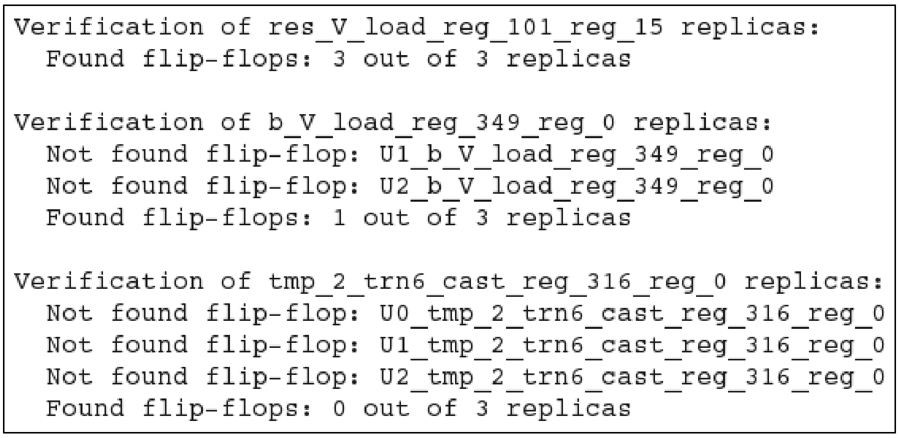
Source: From the author

to merge them. On the other side, the tool was not efficient to optimize the flip-flops without direct interaction with its redundant counterparts in the designs with coarse grain granularity.

Figure 5.3 shows an extract of missing replicas for TMR flip-flops. The amount of removed flip-flops is arbitrary and depends on the analyzed flip-flop.

Reduction in the number of combinational gates can be a consequence of removing partially or completely the redundant logic or replacing simple logic gates by complex logic ones. So, it is not possible to affirm if the fault mitigation capability of the TMR approach is broken or persists. Therefore, it was performed the verification of the structure of combinational paths in the fine grain approaches in order to validate or reject the resulting optimized TMR versions of the low-optimized designs. Figure 5.4 shows an extract of the verification result of the logic cones of flip-flops. It is possible to quantify the number of common gates into the logic cones, as a result, it is observed that the mitigation capability is lost because a fault in those common gates can propagate up to triplicated flip-flops. It remarks the importance of verifying the generated TMR circuit in order to correct the design in early stages of design.

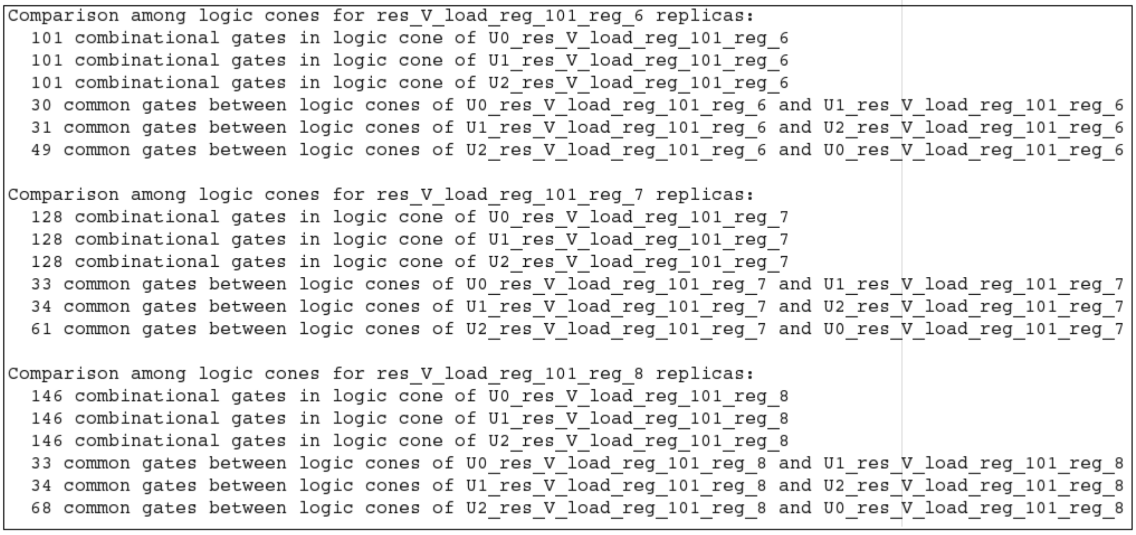
Figure 5.3: Extract from flip-flop's searching report in optimized TMR circuit derived from low-optimized implementation



```
Verification of res_V_load_reg_101_reg_15 replicas:
  Found flip-flops: 3 out of 3 replicas

Verification of b_V_load_reg_349_reg_0 replicas:
  Not found flip-flop: U1_b_V_load_reg_349_reg_0
  Not found flip-flop: U2_b_V_load_reg_349_reg_0
  Found flip-flops: 1 out of 3 replicas

Verification of tmp_2_trn6_cast_reg_316_reg_0 replicas:
  Not found flip-flop: U0_tmp_2_trn6_cast_reg_316_reg_0
  Not found flip-flop: U1_tmp_2_trn6_cast_reg_316_reg_0
  Not found flip-flop: U2_tmp_2_trn6_cast_reg_316_reg_0
  Found flip-flops: 0 out of 3 replicas
```

Source: From the author

Figure 5.4: Extract from logic cones' report of optimized TMR circuit derived from low-optimized implementation



```
Comparison among logic cones for res_V_load_reg_101_reg_6 replicas:
  101 combinational gates in logic cone of U0_res_V_load_reg_101_reg_6
  101 combinational gates in logic cone of U1_res_V_load_reg_101_reg_6
  101 combinational gates in logic cone of U2_res_V_load_reg_101_reg_6
  30 common gates between logic cones of U0_res_V_load_reg_101_reg_6 and U1_res_V_load_reg_101_reg_6
  31 common gates between logic cones of U1_res_V_load_reg_101_reg_6 and U2_res_V_load_reg_101_reg_6
  49 common gates between logic cones of U2_res_V_load_reg_101_reg_6 and U0_res_V_load_reg_101_reg_6

Comparison among logic cones for res_V_load_reg_101_reg_7 replicas:
  128 combinational gates in logic cone of U0_res_V_load_reg_101_reg_7
  128 combinational gates in logic cone of U1_res_V_load_reg_101_reg_7
  128 combinational gates in logic cone of U2_res_V_load_reg_101_reg_7
  33 common gates between logic cones of U0_res_V_load_reg_101_reg_7 and U1_res_V_load_reg_101_reg_7
  34 common gates between logic cones of U1_res_V_load_reg_101_reg_7 and U2_res_V_load_reg_101_reg_7
  61 common gates between logic cones of U2_res_V_load_reg_101_reg_7 and U0_res_V_load_reg_101_reg_7

Comparison among logic cones for res_V_load_reg_101_reg_8 replicas:
  146 combinational gates in logic cone of U0_res_V_load_reg_101_reg_8
  146 combinational gates in logic cone of U1_res_V_load_reg_101_reg_8
  146 combinational gates in logic cone of U2_res_V_load_reg_101_reg_8
  33 common gates between logic cones of U0_res_V_load_reg_101_reg_8 and U1_res_V_load_reg_101_reg_8
  34 common gates between logic cones of U1_res_V_load_reg_101_reg_8 and U2_res_V_load_reg_101_reg_8
  68 common gates between logic cones of U2_res_V_load_reg_101_reg_8 and U0_res_V_load_reg_101_reg_8
```

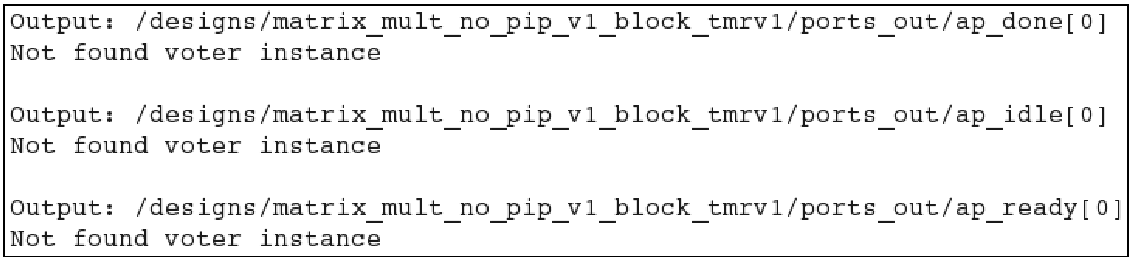
Source: From the author

For output ports, first it is found the voter instances that drive them and the analysis is performed over the logic cones for each one of their input pins. Figure 5.5 shows an extract where no voter instances were found.

In conclusion, it is hard to predict all the situations where the synthesis tool can remove designer's inserted redundancy. Analogously, the implementation of TMR designs from the RTL description depends on the use of properly constraints. Thus, that approach generally results in a greater area overhead and worse performance than employing a full optimized circuit as it was found in the previous section.

The number of combinational gates and flip-flops is coherent with the expected values for the optimized TMR version of the high-optimized implementations. Area over-

Figure 5.5: Extract from voter's searching report in optimized TMR circuit derived from low-optimized implementation



```
Output: /designs/matrix_mult_no_pip_v1_block_tmrv1/ports_out/ap_done[0]
Not found voter instance

Output: /designs/matrix_mult_no_pip_v1_block_tmrv1/ports_out/ap_idle[0]
Not found voter instance

Output: /designs/matrix_mult_no_pip_v1_block_tmrv1/ports_out/ap_ready[0]
Not found voter instance
```

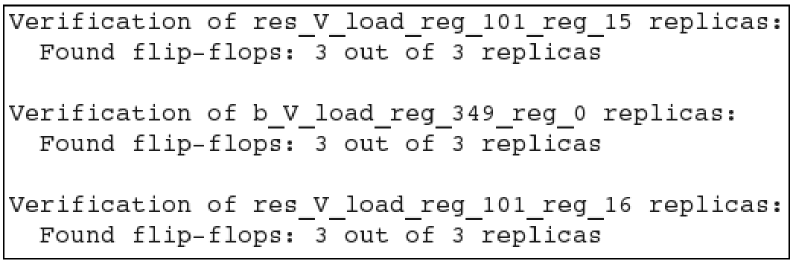
Source: From the author

head and performance after optimization were slightly worsted e improved respectively for around 1% and 2%. The increase in the area happens because the sizing of gates to get a fast response. Also, the synthesis tool added inverters and buffers in order to fix some timing issues.

The resulting optimized TMR implementations were also verified to ensure that no issues were produced by the use of the synthesis tool. Figure 5.6 shows an extract of the full report where it is indicated the presence of all triplicated flip-flops. Common gates among logic cones of TMR flip-flops are also illustrated in Figure 5.7. Surprisingly, several cases with common gates were encountered as in consequence, the fault tolerance capacity was affected. Inspecting these issues, it was found that those common gates were in majority inverters inserted in the gate sizing optimization step. Again, even constraining all the gates for minimum optimization, the synthesis tool corrupted the TMR implementation. Hence, the verification step is a critical task of the proposed automatic flow to guarantee the correctness of the TMR implementation. In this case, the optimization step can be avoided because as it was shown in Table 5.3, it did not make a noticeable impact on performance or area.

On the other side, the logic cones of the output ports were correctly implemented.

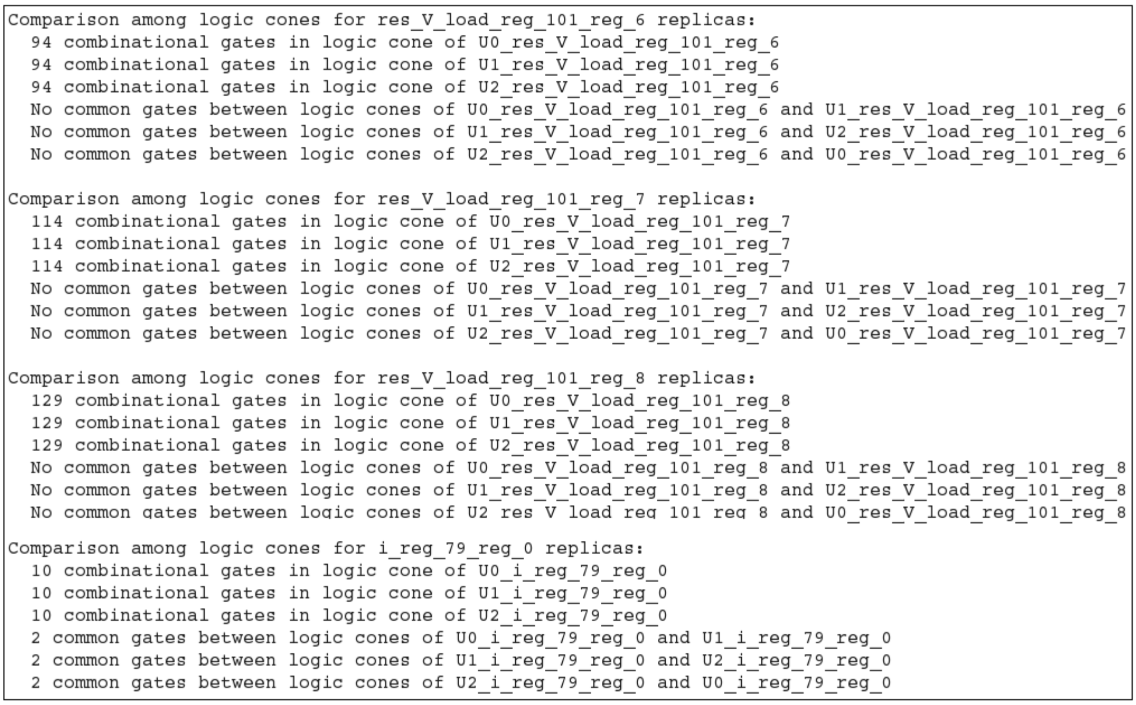
Figure 5.6: Extract from flip-flop's searching report in optimized TMR circuit (gate sizing) derived from high-optimized implementation



```
Verification of res_V_load_reg_101_reg_15 replicas:
  Found flip-flops: 3 out of 3 replicas

Verification of b_V_load_reg_349_reg_0 replicas:
  Found flip-flops: 3 out of 3 replicas

Verification of res_V_load_reg_101_reg_16 replicas:
  Found flip-flops: 3 out of 3 replicas
```

Source: From the author

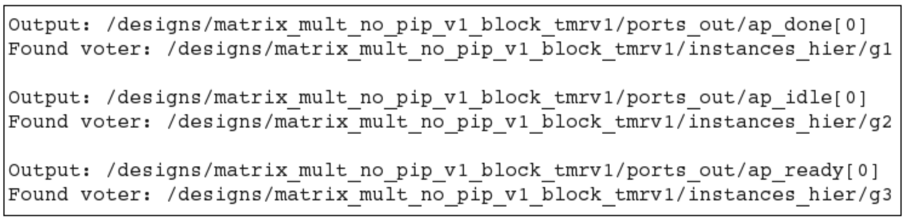
Figure 5.7: Extract from logic cones' report of optimized TMR circuit (gate sizing) derived from high-optimized implementation



```
Comparison among logic cones for res_V_load_reg_101_reg_6 replicas:
  94 combinational gates in logic cone of U0_res_V_load_reg_101_reg_6
  94 combinational gates in logic cone of U1_res_V_load_reg_101_reg_6
  94 combinational gates in logic cone of U2_res_V_load_reg_101_reg_6
  No common gates between logic cones of U0_res_V_load_reg_101_reg_6 and U1_res_V_load_reg_101_reg_6
  No common gates between logic cones of U1_res_V_load_reg_101_reg_6 and U2_res_V_load_reg_101_reg_6
  No common gates between logic cones of U2_res_V_load_reg_101_reg_6 and U0_res_V_load_reg_101_reg_6

Comparison among logic cones for res_V_load_reg_101_reg_7 replicas:
  114 combinational gates in logic cone of U0_res_V_load_reg_101_reg_7
  114 combinational gates in logic cone of U1_res_V_load_reg_101_reg_7
  114 combinational gates in logic cone of U2_res_V_load_reg_101_reg_7
  No common gates between logic cones of U0_res_V_load_reg_101_reg_7 and U1_res_V_load_reg_101_reg_7
  No common gates between logic cones of U1_res_V_load_reg_101_reg_7 and U2_res_V_load_reg_101_reg_7
  No common gates between logic cones of U2_res_V_load_reg_101_reg_7 and U0_res_V_load_reg_101_reg_7

Comparison among logic cones for res_V_load_reg_101_reg_8 replicas:
  129 combinational gates in logic cone of U0_res_V_load_reg_101_reg_8
  129 combinational gates in logic cone of U1_res_V_load_reg_101_reg_8
  129 combinational gates in logic cone of U2_res_V_load_reg_101_reg_8
  No common gates between logic cones of U0_res_V_load_reg_101_reg_8 and U1_res_V_load_reg_101_reg_8
  No common gates between logic cones of U1_res_V_load_reg_101_reg_8 and U2_res_V_load_reg_101_reg_8
  No common gates between logic cones of U2_res_V_load_reg_101_reg_8 and U0_res_V_load_reg_101_reg_8

Comparison among logic cones for i_reg_79_reg_0 replicas:
  10 combinational gates in logic cone of U0_i_reg_79_reg_0
  10 combinational gates in logic cone of U1_i_reg_79_reg_0
  10 combinational gates in logic cone of U2_i_reg_79_reg_0
  2 common gates between logic cones of U0_i_reg_79_reg_0 and U1_i_reg_79_reg_0
  2 common gates between logic cones of U1_i_reg_79_reg_0 and U2_i_reg_79_reg_0
  2 common gates between logic cones of U2_i_reg_79_reg_0 and U0_i_reg_79_reg_0
```

Source: From the author

Figure 5.8: Extract from voter's searching report in optimized TMR circuit (gate sizing) derived from high-optimized implementation



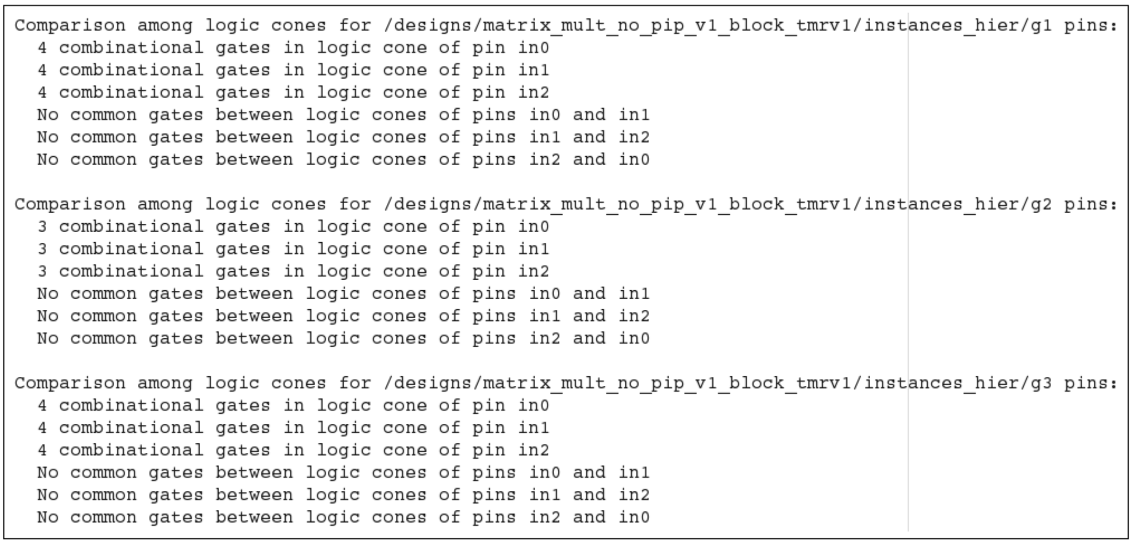
```
Output: /designs/matrix_mult_no_pip_v1_block_tmrv1/ports_out/ap_done[0]
Found voter: /designs/matrix_mult_no_pip_v1_block_tmrv1/instances_hier/g1

Output: /designs/matrix_mult_no_pip_v1_block_tmrv1/ports_out/ap_idle[0]
Found voter: /designs/matrix_mult_no_pip_v1_block_tmrv1/instances_hier/g2

Output: /designs/matrix_mult_no_pip_v1_block_tmrv1/ports_out/ap_ready[0]
Found voter: /designs/matrix_mult_no_pip_v1_block_tmrv1/instances_hier/g3
```

Source: From the author

Figure 5.8 shows an extract where all voter instances were found. Furthermore, Figure 5.9 illustrates that there are not common gates among the redundant logic cones that drives the voter instances' pins.

Figure 5.10, Figure 5.11 and Figure 5.12 show the verification reports that indicate the correct functionality and fault-tolerant capability of the implemented TMR versions for MM1, MM2 and MM3 circuits respectively. The matched flip-flops and primary input and outputs were found equivalent to their unmitigated counterparts. Also, the equivalence between the merged flip-flops was checked in the aforementioned figures. The extra flip-flops represent the ones that can take different values in relation to its merged replicas. Eventually, the final result "PASS" points out the correctness of the TMR implementations.

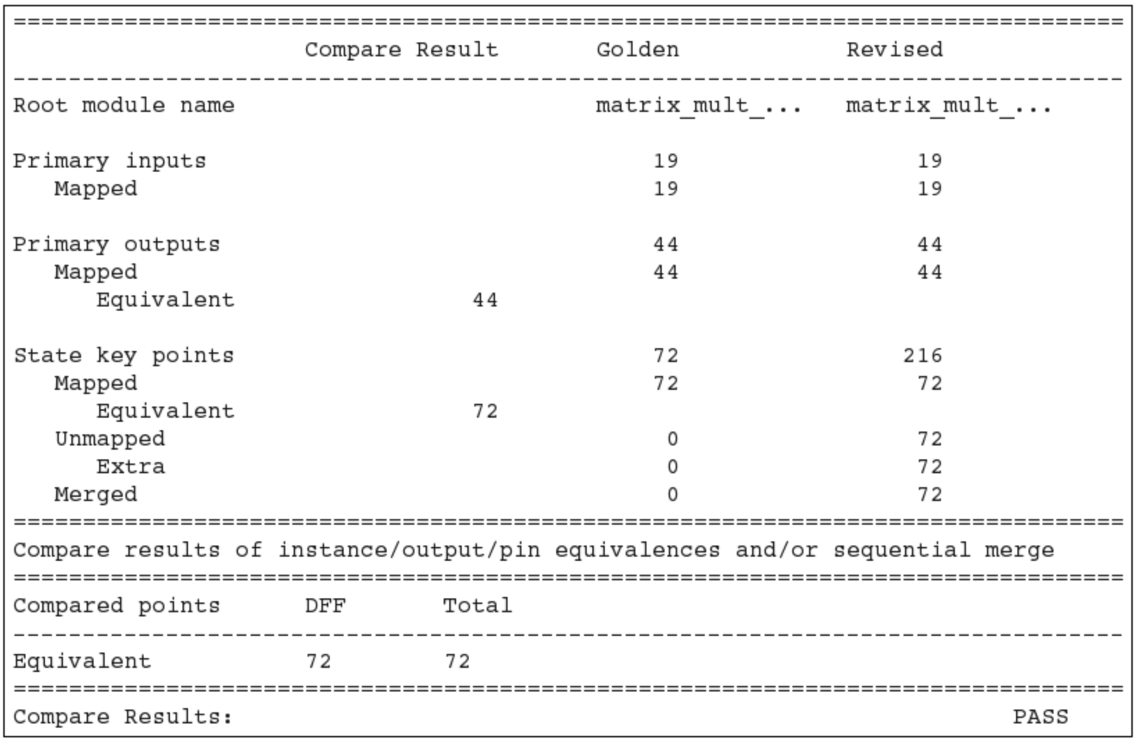
Figure 5.9: Extract from logic cones' searching report of voters in optimized TMR circuit (gate sizing) derived from high-optimized implementation



```
Comparison among logic cones for /designs/matrix_mult_no_pip_v1_block_tmrv1/instances_hier/g1 pins:
  4 combinational gates in logic cone of pin in0
  4 combinational gates in logic cone of pin in1
  4 combinational gates in logic cone of pin in2
  No common gates between logic cones of pins in0 and in1
  No common gates between logic cones of pins in1 and in2
  No common gates between logic cones of pins in2 and in0

Comparison among logic cones for /designs/matrix_mult_no_pip_v1_block_tmrv1/instances_hier/g2 pins:
  3 combinational gates in logic cone of pin in0
  3 combinational gates in logic cone of pin in1
  3 combinational gates in logic cone of pin in2
  No common gates between logic cones of pins in0 and in1
  No common gates between logic cones of pins in1 and in2
  No common gates between logic cones of pins in2 and in0

Comparison among logic cones for /designs/matrix_mult_no_pip_v1_block_tmrv1/instances_hier/g3 pins:
  4 combinational gates in logic cone of pin in0
  4 combinational gates in logic cone of pin in1
  4 combinational gates in logic cone of pin in2
  No common gates between logic cones of pins in0 and in1
  No common gates between logic cones of pins in1 and in2
  No common gates between logic cones of pins in2 and in0
```

Source: From the author

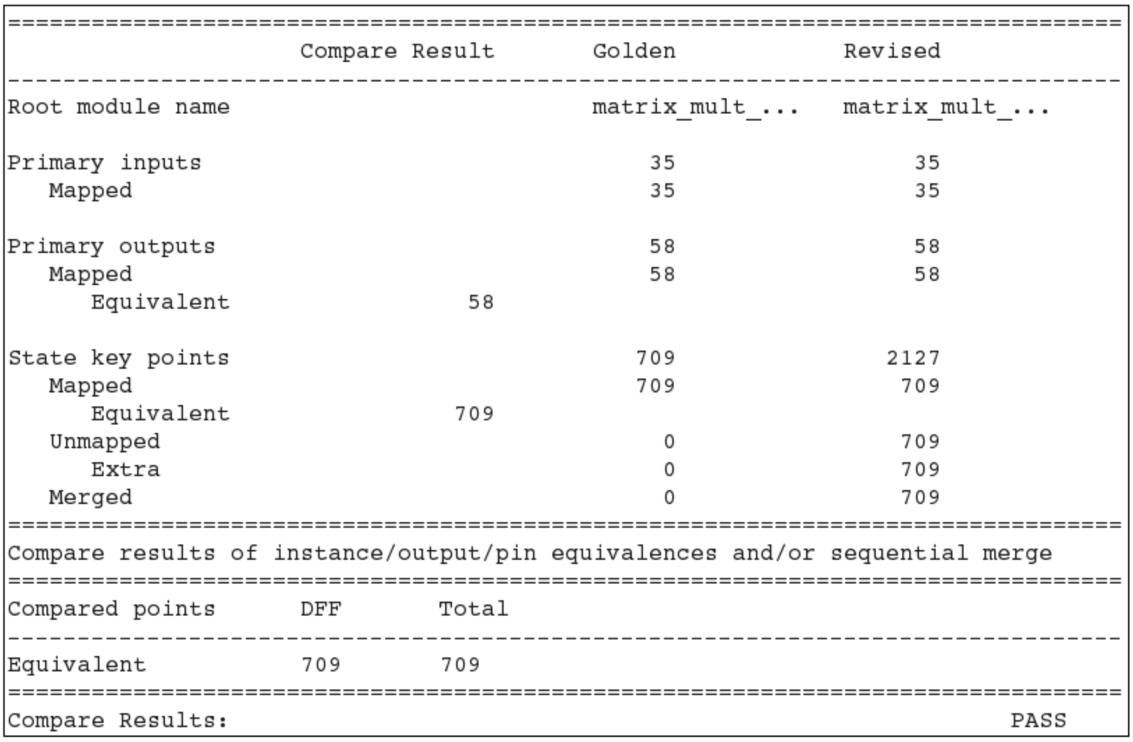
Figure 5.10: Verification report shows the correct implementation of the TMR versions of MM1



```
================================================================================
                     Compare Result      Golden            Revised
--------------------------------------------------------------------------------
Root module name                         matrix_mult_...   matrix_mult_...

Primary inputs                               19                 19
   Mapped                                    19                 19

Primary outputs                              44                 44
   Mapped                                    44                 44
      Equivalent               44

State key points                             72                216
   Mapped                                    72                 72
      Equivalent               72
   Unmapped                                   0                 72
      Extra                                   0                 72
   Merged                                     0                 72
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF       Total
--------------------------------------------------------------------------------
Equivalent           72        72
================================================================================
Compare Results:                                                        PASS
```

Source: From the author

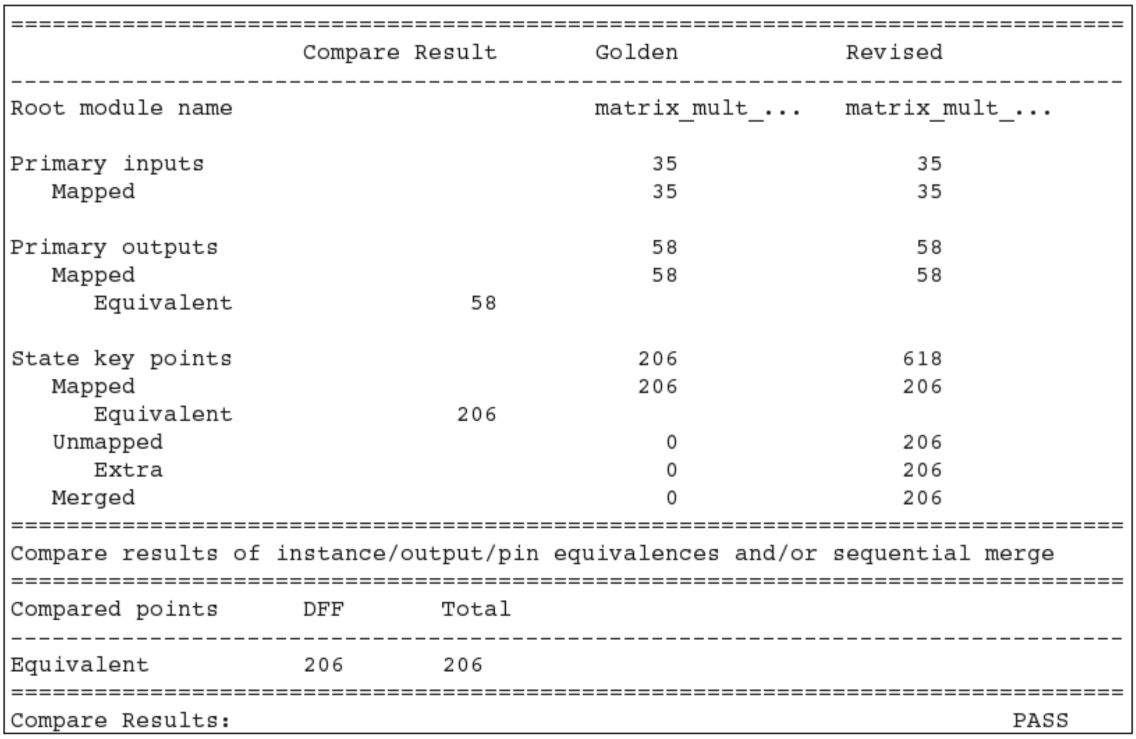
Figure 5.11: Verification report shows the correct implementation of the TMR versions of MM2



```
================================================================================
                     Compare Result       Golden           Revised
--------------------------------------------------------------------------------
Root module name                          matrix_mult_...   matrix_mult_...

Primary inputs                                35                 35
   Mapped                                     35                 35

Primary outputs                               58                 58
   Mapped                                     58                 58
      Equivalent                 58

State key points                             709               2127
   Mapped                                    709                709
      Equivalent                709
   Unmapped                                    0                709
      Extra                                    0                709
   Merged                                      0                709
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF       Total
--------------------------------------------------------------------------------
Equivalent           709       709
================================================================================
Compare Results:                                                        PASS
```

Source: From the author

Figure 5.12: Verification report shows the correct implementation of the TMR versions of MM3



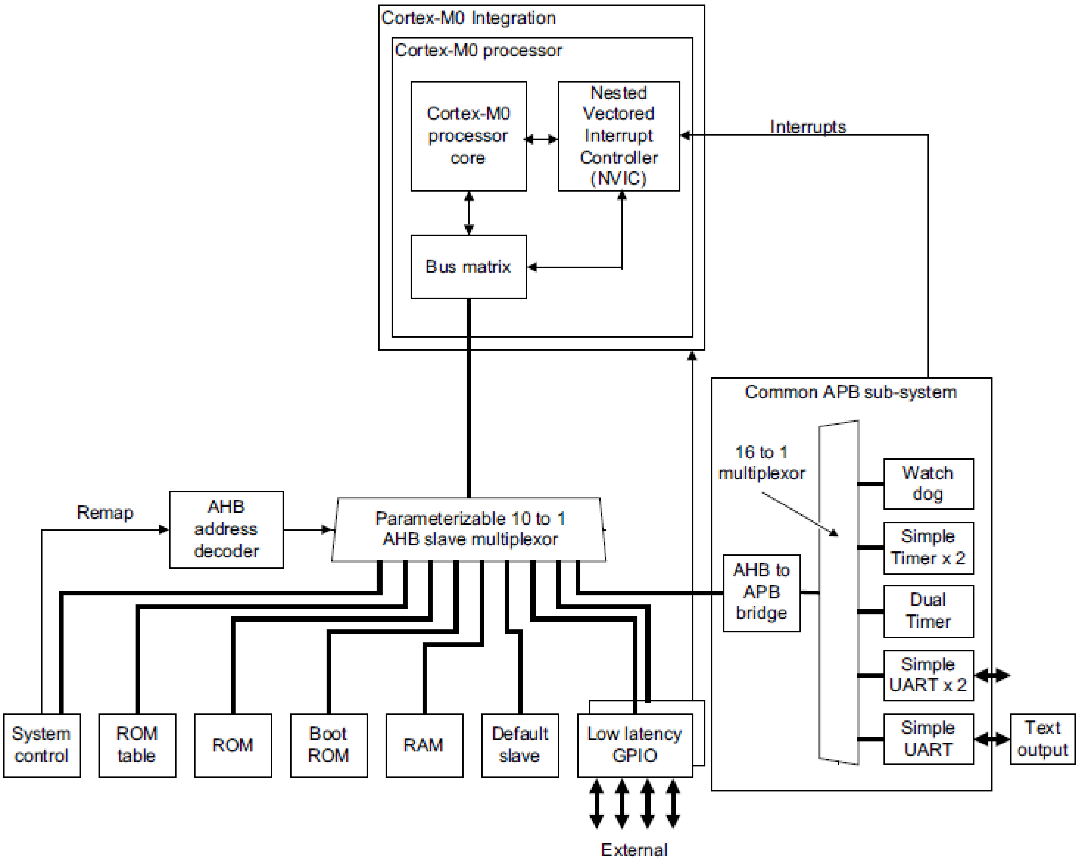
```
================================================================================
                     Compare Result       Golden           Revised
--------------------------------------------------------------------------------
Root module name                          matrix_mult_...   matrix_mult_...

Primary inputs                                35                 35
   Mapped                                     35                 35

Primary outputs                               58                 58
   Mapped                                     58                 58
      Equivalent                 58

State key points                             206                618
   Mapped                                    206                206
      Equivalent                206
   Unmapped                                    0                206
      Extra                                    0                206
   Merged                                      0                206
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF       Total
--------------------------------------------------------------------------------
Equivalent           206       206
================================================================================
Compare Results:                                                        PASS
```

Source: From the author

5.3 Fault-tolerant FPGA design flow

For the FPGA design flow design, a soft-core IP processor was chosen as a case-study. The ARM Cortex-M0 processor is a soft-core processor available in the ARM DesignStart Eval package for free use. The Cortex-M0 processor is a 32-bit processor that implements the Armv6-M architecture. It employs a three-stage pipeline for low-area implementation and it is capable of achieving performance figures of 2.33 CoreMarks/MHz (ARM, 2017). Figure 5.13 illustrates the Cortex-M0 processor in an example system provided in the ARM DesignStart Eval package. The processor communicates with the rest of the system through an AHB Lite interface. The Cortex-M0 processor from DesignStart Eval package features only a master port of the AHB Lite interface. The RTL description of the Cortex-M0 processor is an obfuscated, although synthesizable Verilog code. The obfuscated code brings the opportunity to show how the proposed methodology is able to implement successfully fault-tolerant versions of large designs even without the need of identifying its behavioral description.

Figure 5.13: System example based on the ARM Cortex-M0 processor



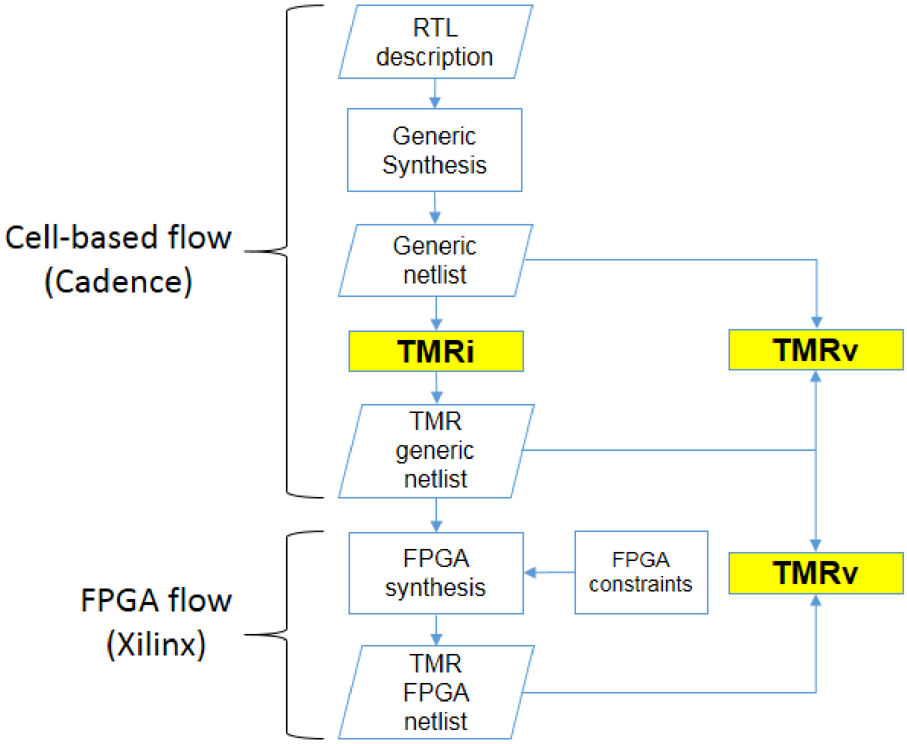
Source: (ARM, 2017)

Other works that implement the TMR technique over the ARM core utilized the FGLTMR approach with triplicated flip-flops or the CGTMR with voters placed at the output port of the entire triplicated circuit (BALLAST et al., 2015; ÖZER; Ghahroodi, M; BULL, 2013). Even a commercial radiation hardened micro-controller based on the ARM Cortex-M0 only uses the FGLTMR approach (BANNATYNE et al., 2016). In the knowledge of the author, there is not past work implementing the FGDTMR version of any family of ARM processors. As part of the experiments, the CGTMR, FGTMR and moreover the FGDMTR of the ARM Cortex-M0 processor were implemented using the proposed automated methodology.

5.3.1 Analyzing synthesis in the cell-based and FPGA design flows

For this work, it was only applied the TMR technique over the Cortex-M0 processor block shown in Figure 5.13. Program memory (PM), data memory (DM), as well as other peripherals present in the ARM processor system were not protected using the TMR technique. The TMR versions were implemented using Cadence tools and then exported to the Xilinx FPGA flow as illustrated in Figure 5.14.

Figure 5.14: Procedure to evaluate the proposed FPGA flow to design TMR circuits



Source: From the author

The synthesis was done restricting the use of generic primitive gates. Furthermore, the TMR circuits employ flip-flops with no more than one asynchronous input. These considerations ensure that the resulting netlist can be synthesized adequately using primitive logic blocks of the target FPGA device. In this work is used the programmed logic of the Xilinx Zynq-7000 Z030 SoC, which is built on 28 nm process technology and mainly uses high-k metal gate (HKMG) (XILINX, 2015). The target system clock frequency was 50 MHz. Because the mapping process was performed considering the FPGA resources, constraints must be used to avoid removing the redundancies in the FPGA design flow.

Table 5.4 shows the synthesized implementation results of the ARM Cortex-M0 processor and its TMR versions following the cell-based and FPGA design flows. Figure 5.15 shows the correctness of the TMR implementations.

In the cell-based section of Table 5.4, the number of flip-flops in the TMR implementations is three times as expected. Also, majority voters inserted at the output ports and for each triplicated flip-flops represent 13% and 25% of the number of gates of the unmitigated design.

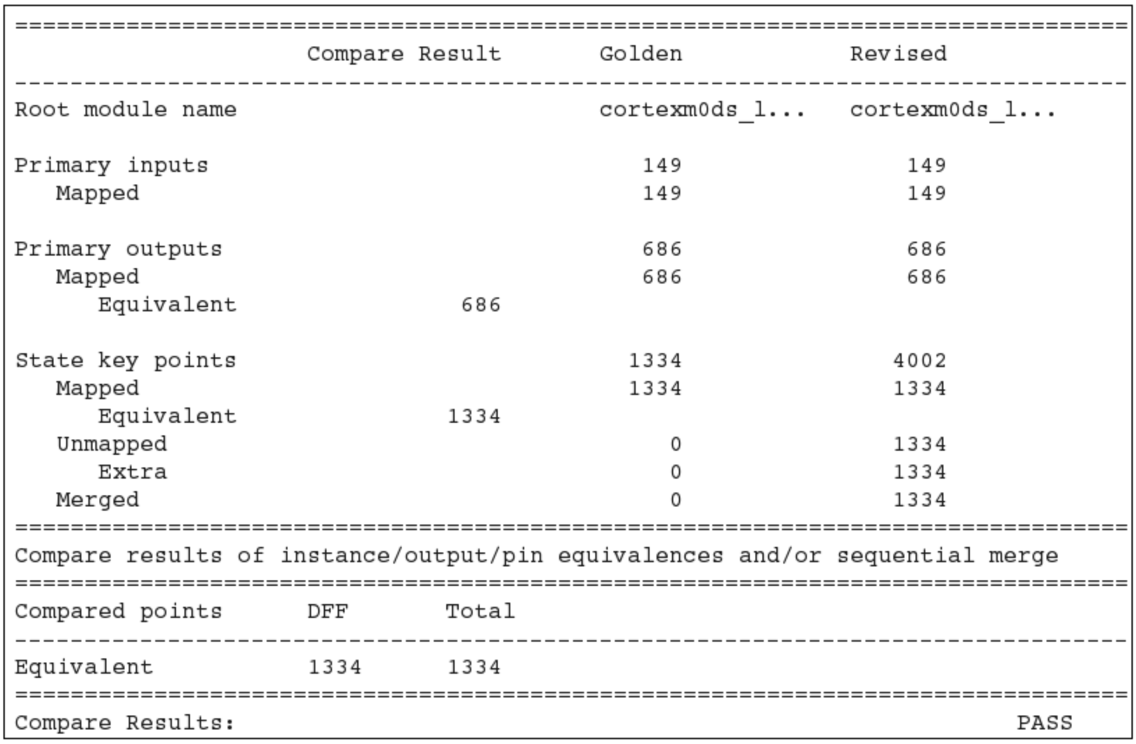
In the case of the FPGA section, the unmitigated circuit was optimized by the Xilinx's synthesis tool whereas, the TMR circuits were constrained to preserve the redundant logic. In consequence, the number of flip-flops is reduced in almost 25% in the unmitigated implementation but it remains the same for TMR implementations in comparison with the cell-based implementations. Thus, the number of flip-flops in the TMR implementations is more than three times. The reduction of flip-flops in the unmitigated circuit also affects the proportion of LUTs between it and its TMR counterparts because of the removal of their logic cones.

Table 5.4: Summary of TMR synthesis results of generic cell-based and FPGA implementations

Design	**Unmitigated**	**CGTMR**	**FGLTMR**	**FGDTMR**
Generic cell-based flow				
Combinational gates	21522 (1×)	67310 (3.13×)	26858 (1.25×)	83318 (3.87×)
Flip-flops	1334 (1×)	4002 (3.00×)	4002 (3.00×)	4002 (3.00×)
FPGA flow				
LUTs	3104 (1×)	16683 (5.37×)	6681 (2.15×)	20854 (6.72×)
Flip-flops	1027 (1×)	4002 (3.90×)	4002 (3.90×)	4002 (3.90×)
Configuration bits	1110127 (1×)	5983337 (5.39×)	2340227 (2.11×)	7516339 (6.77×)
Max. Frequency (MHz)	54 (1×)	53.6 (0.99×)	54.6 (1.01×)	52.2 (0.97×)

Source: From the author

Figure 5.15: Verification report shows the correct implementation of the TMR versions of the ARM Cortex-M0 processor



```
================================================================================
                     Compare Result       Golden           Revised
--------------------------------------------------------------------------------
Root module name                          cortexm0ds_l...   cortexm0ds_l...

Primary inputs                               149               149
   Mapped                                    149               149

Primary outputs                              686               686
   Mapped                                    686               686
      Equivalent                686

State key points                            1334              4002
   Mapped                                   1334              1334
      Equivalent               1334
   Unmapped                                    0              1334
      Extra                                    0              1334
   Merged                                      0              1334
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF       Total
--------------------------------------------------------------------------------
Equivalent           1334      1334
================================================================================
Compare Results:                                                        PASS
```

Source: From the author

Maximum operating frequency was slightly impacted by the use of TMR in the FPGA flow with a maximum degradation of 3% for the FGDTMR implementation. The FGLTMR exhibits an increment of 1% in the maximum frequency that it may be a result of a better placement of the utilized resources or because a large logic cone was better optimized with regard to the unmitigated circuit.

5.3.2 Fault injection campaigns

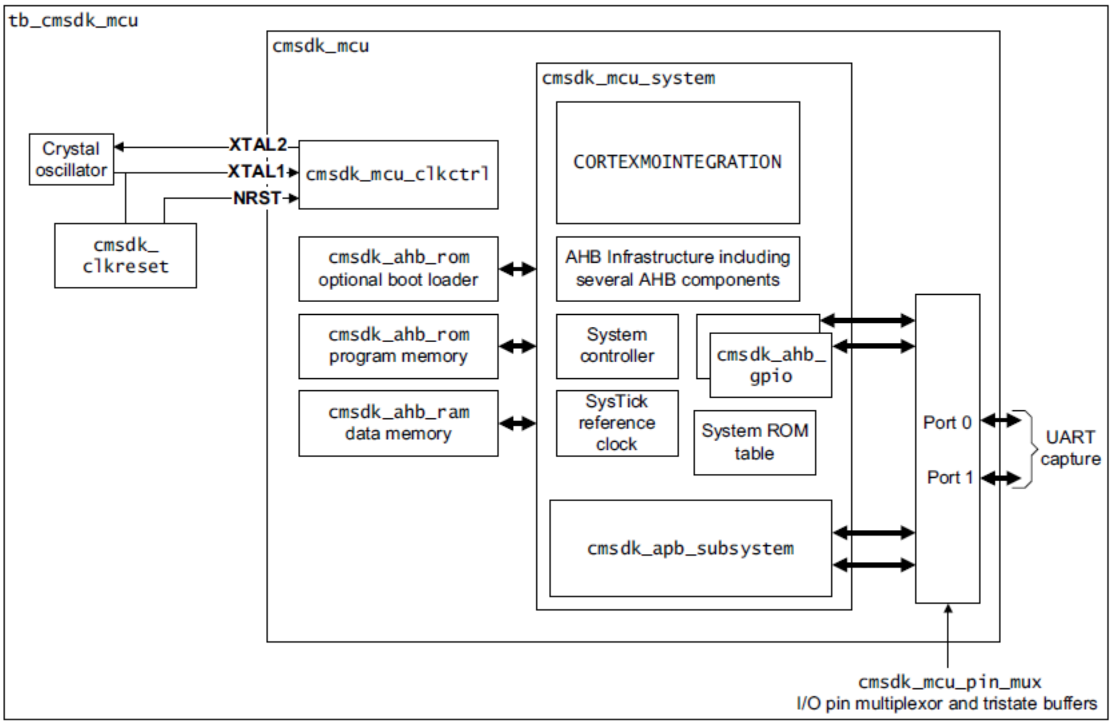
SRAM-based FPGA solutions are very attractive because of their fast time-to-market, high density and reconfiguration capability. However, they are susceptible to the effect of radiation even at ground level. Neutron-induced soft errors in the FPGA's configuration memory may result in unexpected bit-flips that can provoke erroneous operation of the entire system. Thus, it is mandatory to analyze the possible effects of radiation before the deployment of the system. Accelerated radiation test allows to collect data in a much shorter time but it is very expensive. Another alternative is the emulated fault injection approach, which comes up with a suitable and affordable solution to analyze radiation's susceptibility in early stages of development. Emulated fault injection is an accepted approach that speeds up the fault injection campaigns instead of using simulation, which is very time and resource consuming for large systems. In this work, it is employed the fault injector platform developed and detailed in (TONFAT et al., 2015) to analyze the effect of bit-flips in the configuration memory of the unmitigated and TMR implementations of the ARM Cortex-M0 processor. Bit-flips in the configuration memory can affect both interconnection and logic implementation of the circuit and in consequence, its operation can be corrupted. Unlike flip-flops, which update their states in the next triggering clock edge, the configuration memory is usually loaded only once before the deployment of the circuit. Thus, a bit-flip in any of these memory cells remain until the design is updated for maintenance or in case of updating the functionality of the system. Consequently, the configuration memory represents the most vulnerable part of an SRAM-based FPGA system to SEUs because these memory cells are abundant, spread over the entire area and mainly its data is loaded once. For instance, in Table 5.4 it is shown that the number of utilized configuration bits is considerably greater than the number of flip-flops, from around 600 times in FGLTMR to about 2000 times in FGDTMR.

The emulated fault injection campaigns performed in this work consider the accumulative effect of bit-flips in the configuration memory. Figure 5.16 shows the example microcontroller system provided into the Cortex-M0 DesignStart Eval package and employed to perform the fault campaigns.

The software algorithm corresponding to matrix multiplication was executed in the ARM Cortex-M0 processor for experimentation. The matrix multiplication algorithm, unlike the first case-study circuit, employs 32x32 array inputs compound of 32-bit vectors. The matrix operands and result were loaded into the memory data implemented by BRAMs. Single faults are inserted randomly before starting each execution of the application. In case of correct system response, the application is restarted and a new bit-flip is added randomly in order to emulate the persistence of faults in the configuration memory. The accumulated bit-flips remains up to an error is found.

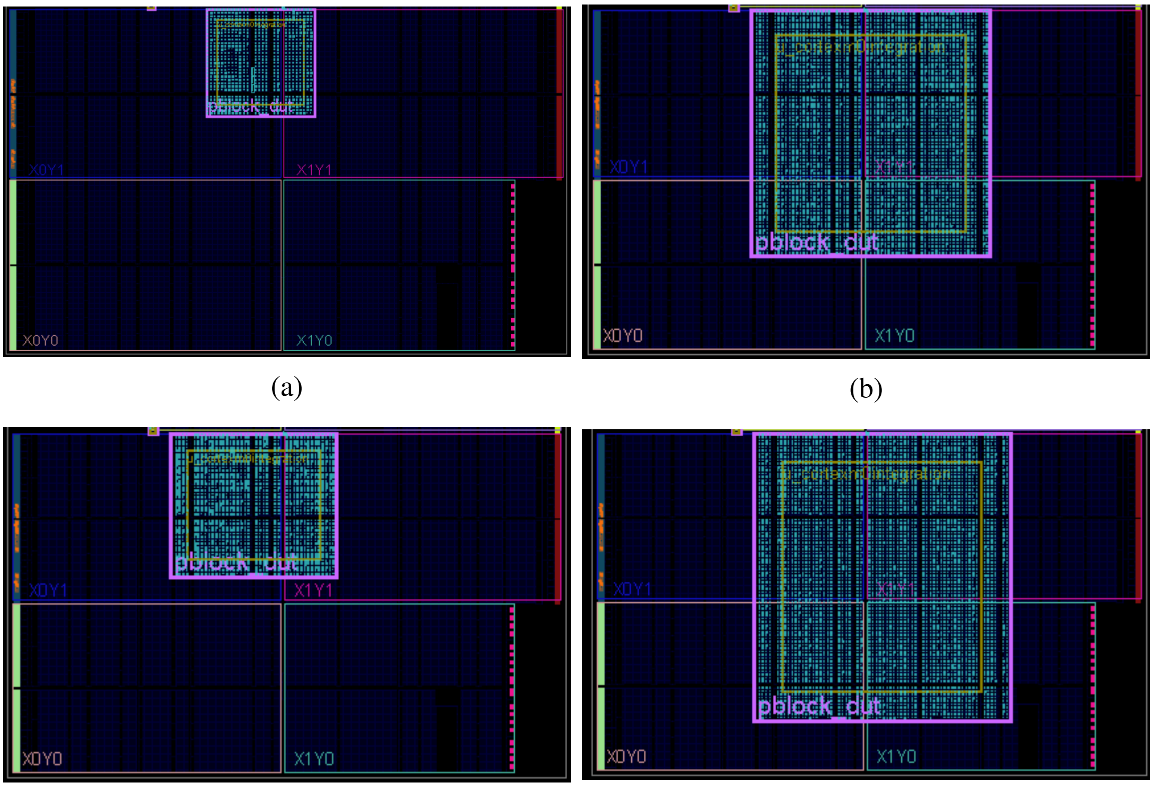
The unmitigated and TMR versions of the ARM Cortex-M0 processor were placed into the same regions of the FPGA in order to limit the frame of configuration bits in the fault injection. That means, the configuration bits used for the biggest TMR implementation (FGDTMR) contains the configuration bit of the rest. So, the fault injection campaigns are over this big section in order to emulate the same flux of particle over the same area. Figure 5.17 illustrates the placement of the unmitigated and TMR versions of the ARM Cortex-M0 processor. The above region represents the safe block where no faults

Figure 5.16: Microcontroller system provided into the ARM Cortex-M0 DesignStart Eval package



Source: (ARM, 2017)

Figure 5.17: Placement of a) unmitigated and its TMR implementations: b) CGTMR, c) FGLTMR and d) FGDTMR



(a)

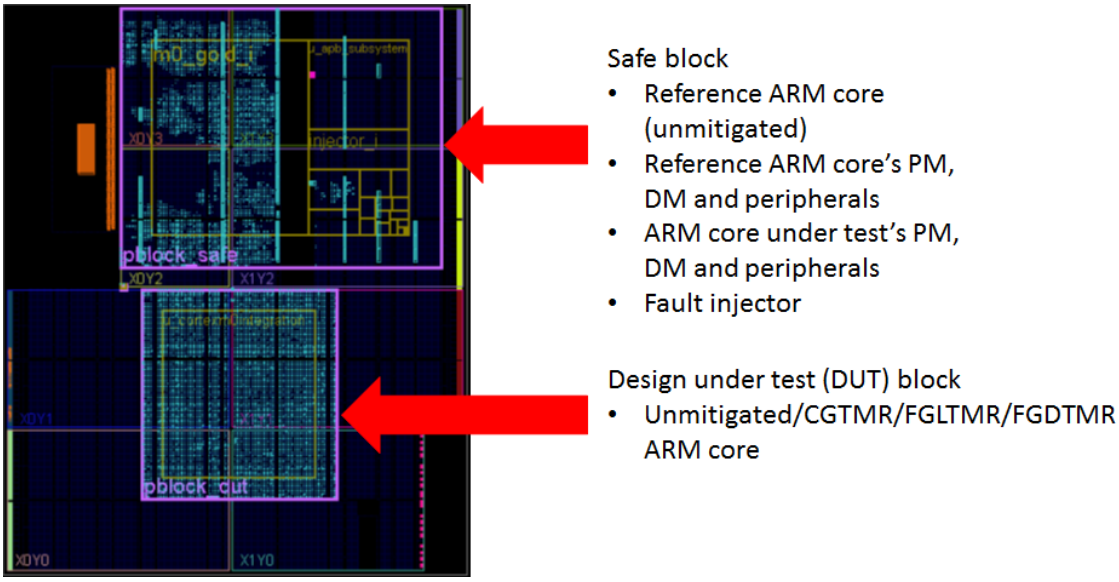
(b)

(c)

(d)

Source: From the author

Figure 5.18: Placement of the reference and the implementations under test of the ARM Cortex-M0 processor



Safe block

- Reference ARM core (unmitigated)
- Reference ARM core's PM, DM and peripherals
- ARM core under test's PM, DM and peripherals
- Fault injector

Design under test (DUT) block

- Unmitigated/CGTMR/FGLTMR/FGDTMR ARM core

Source: From the author

are injected. The upper region represents a safe block where no faults are injected. This region contains an ARM core which brings the correct values of the matrix multiplication application. Also, the data memories, program memories, and peripherals for both the reference design and the design under test (DUT) are placed in the safe block. On the other side, the lower region only contains an ARM processor implementation under test which runs the application in the presence of faults in its configuration memory.

Two separated blocks containing implementations of the ARM core were executed simultaneously the same application with the aim of comparing the normal response against the circuit under the presence of bit-flips in its configuration memory. The unmitigated implementation of the ARM processor was taken into account as the design that provides the correct response in each execution. Figure 5.18 illustrates the placement of the reference and the implementations under test of the ARM processor.

According to the manifestation of the errors in relation with the behavior of the circuit, they can be classified in: timeout, where the faulty circuit takes more time to finish and even stays in an infinite loop without finishing the processing, and silent data corruption (SDC), where the execution time is respected but the result is incorrect. Table 5.5 classifies the errors found in the fault injection campaigns. For all implementation, reaching to 1000 errors was the target to evaluate the rate of error of each type.

The amount of timeout errors in the CGTMR implementation is very close to the results for the unmitigated circuit. That can be explained because the coarse grain

Table 5.5: Classification of errors from the fault injection campaigns

	Unmitigated	**CGTMR**	**FGLTMR**	**FGDTMR**
Total faults	66404	40353	47203	99060
Total errors	1000 (100%)	1000 (100%)	1000 (100%)	1000 (100%)
SDC	385 (39%)	366 (37%)	512 (51%)	825 (82%)
Timeout	615 (61%)	634 (63%)	488 (49%)	175 (18%)

Source: From the author

approach is not suitable to avoid state corruption of the FSM that controls the circuit. Therefore, due to the accumulative effects of bit-flips in the configuration memory, a corrupted state can remain latent for the next execution increasing the probability of error. On the other hand, voters inserted after flip-flops reduce the number of timeout error in fine grain implementations. The rate of timeout errors is reduced by about 20% and a striking 80% for FGLTMR and FGDTMR implementations respectively.

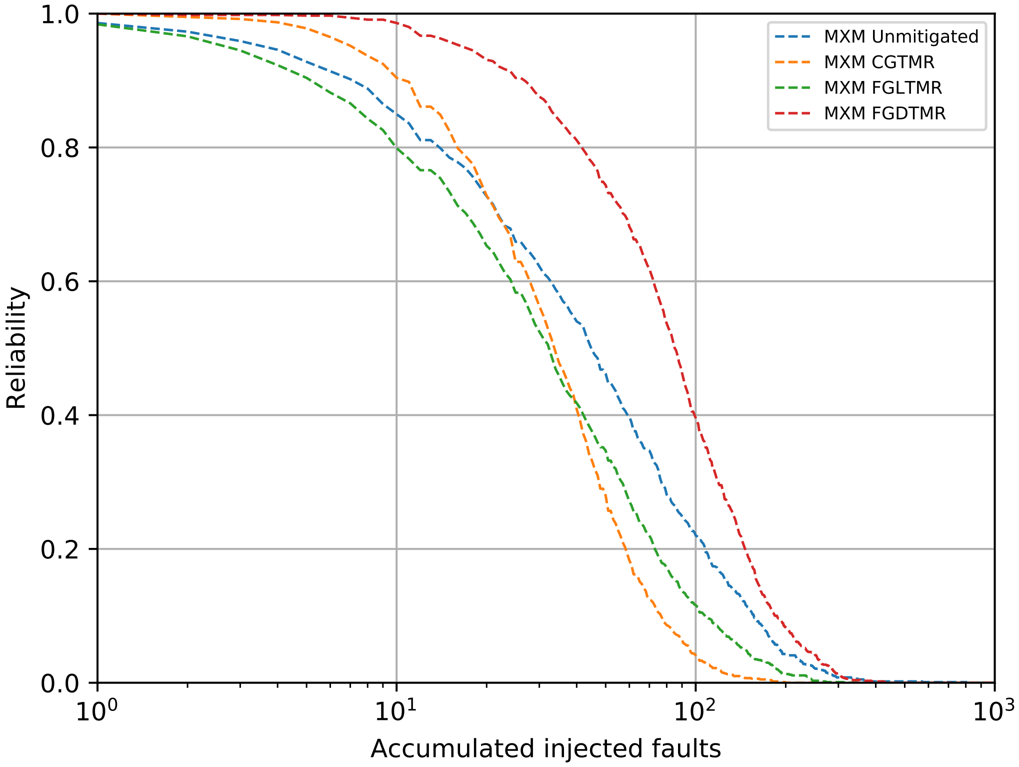
5.3.3 Analyzing reliability of unmitigated and TMR implementations

Reliability is the probability of a system operates correctly in a specified interval of time. It can also be understood as the complement of the probability of error of the system. Thus, the reliability curve can be derived from the classification of the rate of erroneous executions in relation to the amount of accumulated injected faults (BENEVENUTI; KASTENSMIDT, 2018). Figure 5.19 depicts the reliability curve of the ARM Cortex-M0 processor running the matrix multiplication algorithm.

From the reliability curve is remarkable to notice that the CGTMR and FGDTMR approaches could mask all the single injected bit-flips in the configuration memory bits. That also corroborated the correct implementation of the TMR implementations. Meanwhile, the unmitigated and the FGLTMR versions could not mask all the single injected faults as expected. In addition, the FGLTMR and the unmitigated implementations have similar reliability levels under the presence of single faults. Furthermore, it is remarkable to realize the poor reliability levels of the FGLTMR implementation even for a few accumulated faults. This can be attributed to the replication and insertion of voters that increased in twice the number of configuration bits. So, the FGDTMR and the CGTMR implementations were the most proper approaches to protect the circuit from the effects of radiation whereas, the FGLTMR approach was not suitable for this application. These results were obtained for the total number of 1000 errors and injecting random faults. The exhaustively fault injection of the susceptibility would show a better representation of the susceptibility of the unmitigated and FGLTMR implementations.

Comparing the accumulated effect of faults among the four implementations, the FGDTMR presents the best reliability levels. The FGDTMR is suitable to operate at a high level of reliability even for 10 accumulated faults. The CGTMR was more efficient to mitigate up to 20 accumulated faults than the FGLTMR and the unmitigated implementations. From that point, the unmitigated circuit responds better than the CGTMR and the FGLTMR.

Figure 5.19: Reliability curve from accumulated fault injection campaigns



Source: From the author

5.4 Conclusion

In this section was showed the application of the proposed methodology to implement TMR designs from a post-synthesized netlist, as it was detailed in the previous chapter. Two study-case circuits were considered in order to follow the cell-based and the FPGA flow to design semi-custom digital integrated circuits respectively. For the first case-study circuit, a matrix multiplication generated using the HLS approach it is employed to show the advantage of employing a post-synthesis netlist instead of behavioral description to implement its TMR versions. This circuit was synthesized using low-effort levels of optimization in order to make an analogy with the implementation of TMR circuits directly from the RTL description. In both approaches, the circuit is poorly optimized by the synthesis tool. Furthermore, a high optimized implementation is also generated in order to notice the advantages of allowing aggressive optimization before applying the TMR technique. The implementation of the TMR circuits was verified to show how a commercial tool can corrupt the redundant elements. Whereas the verification of the presence of triplicated flip-flops in the resulting TMR circuit is a straightforward task, the correct replication of combinational logic paths must consider different cases in order to guarantee that redundant paths do not share any elements. Moreover, the replicas must perform the same boolean function and provide the expected mitigation capability against single faults. Our proposed methodology also performs further optimization of the generated TMR circuit. However, it was encountered that the TMR implementation was corrupted even constrained the commercial tool to perform only simple sizing of gates. In consequence, the designer must be careful and always verify the correctness of the resulting implementation when consider using commercial tools to implement redundancy-based mitigation techniques.

For the second case-study circuit, TMR versions of the ARM Cortex-M0 processor were generated and implemented in FPGA. A contribution of the present work is to make available the FGDTMR version of the ARM processor. Emulated fault injection campaigns were performed with the finality of analyzing the susceptibility of the ARM processor running a benchmark application. Moreover, the reliability of the application was evaluated through the effects of accumulated faults. The traced reliability curve showed that poor fault tolerant of the FGLTMR approach of the circuit implemented in FPGA. On the other hand, the FGDTMR presented an outstanding reliability level of up to 10 accumulated faults.

6 CONCLUSION

The objective of this work was to develop a methodology to automate the design of TMR circuits for cell-based and FPGA designs. The Cadence's commercial design flow was adapted to design fault-tolerant circuits. Hence, the need for dedicated tools along with the issues of the ITAR regimen are avoided. The outcome of this work was the development of the Tcl libraries TMRi and TMRv to implement a verify the use of the TMR technique in different granularity levels respectively. In spite of these libraries are based on the Cadence's data hierarchy and commands, they can be extended to other commercial tools by replacing the proprietary commands adequately.

The most representative features of the TMRi library are the reuse of existing designs initially not addressed for critical applications, the selection of the granularity level of the TMR technique, the protection of FSMs against accumulated faults, the no dependency of a fabrication process and its utilization for both cell-based and FPGA designs.

Furthermore, the chosen study-case circuits showed how the use of a structural netlist to implement TMR designs exhibits the advantage of non-intervention in the behavioral design. Therefore, the utilization of HLS and RTL descriptions is compatible with the developed approach. In the case-study circuit for the cell-based flow, it was observed that a fully optimized netlist can get a reduction of 8% in its area compared to a circuit with restricted optimization. So, it is more effective to utilize a netlist than a constrained behavioral description to implement TMR circuits. An important observation is a considerable increment in area derived from the insertion of voters, which can reach up to 110% according to the number of flip-flops, output ports and the granularity level of the replication. Thus, the use of small majority voters contributes to the reduction in area overhead.

The use of a structured netlist eases the verification process because of the direct relation between the flip-flops of the unmitigated circuit and its TMR counterparts. The verification process was conveniently divided into two steps: structural verification and the logic equivalence verification. It is remarked that the structural verification of logic cones of output ports is conditioned to the presence of the voters. If they are not identified in the TMR circuit, the structural verification cannot be applied with the resources of the TMRv library.

The proposed fault-tolerant design flow also estates the possibility to apply further optimization in the resulting TMR netlist. Gate sizing optimization was considered for that purpose. However, during the verification of the TMR circuits, it was found that even constraining all the gates for this purpose, the TMR implementation can be corrupted in its structure but not in its functionality. So, that fact highlights the importance of performing both structural and logical verification of the resulting TMR circuit. Also, the use of the optimization step requires special attention.

In the case of the circuit that follows that FPGA design flow, the size of the configuration memory is much more than 3 times for the FGDTMR and CGTMR implementations. That shows a limitation of generating a generic TMR netlist for the FPGA flow. So, the resulting circuit is not totally optimized. However, the use of the proposed approach must be balanced between the desired reliability level and the overhead caused by the use of a generic netlist. This issue is not found in the cell-based approach.

Another contribution of this work is to make available the FGDTMR version of the ARM Cortex-M0 soft-core. Although it was derived from a reduced version of the ARM soft-core, its RTL description is open access and validated by ARM company so further research over this implementation is relevant. The set of TMR implementations of the ARM soft-core was verified and also synthesized in FPGA. Also, their capability to mitigate accumulated errors in the configuration memory was evaluated using emulated-based fault injections. According to the results from the fault injection campaigns, the FGDTMR implementation presented the highest reliability level even in the presence of up to 10 accumulated faults. On the other hand, the FGLTMR version showed the worst reliability level even lower than the unmitigated circuit. The reliability level of CGTMR approach drops after a few accumulated faults but its mitigation capacity is slightly better than the unmitigated circuit.

The fault-injection campaigns applied in the FPGA implementations of the ARM soft-core exhibit the efficacy of the voters inserted for enabled flip-flops because of the low rate of timeout errors in the TMR implementations.

Moreover, the fault-injection campaigns also corroborate the correct implementation of the CGTMR and FGTMR designs, which present full mitigation against single faults.

In order to improve the reliability of the FPGA designs, regular refreshing of the configuration memory can be applied. For instance, authors in (CARMICHAEL; CAFFREY; SALAZAR, 2000) describes the use of partial reconfiguration for the purpose of correcting SEUs in the configuration memory. Authors in (AZAMBUJA et al., 2009) applies dynamic partial reconfiguration with TMR technique in SRAM-based FPGA designs. They employ reconfiguration only in faulty modules with the finality of reducing fault recovery time and energy consumption in CGTMR designs.

A collateral effect of the experiments is related to the fact that the Cadence synthesis tool was not able to eliminate efficiently the redundant paths in the coarse grain approach. That exhibits a limitation of the tool to remove redundant paths.

This work also gave a revision of the radiation effects on electronics in order to offer the background and expose the increased concern of these effects in modern technologies. Also, fault-tolerant techniques were revised to serve as a guide for designers who deal with this type of applications.

6.1 Future work

The main drawback of the TMR technique is the overhead in area caused by the use of replicas and the insertion of voters. Thus, several approaches can be considered to improve the implementation of TMR designs through the proposed approach.

One way to reduce the cost of employing redundancy is to apply it selectively to a portion of the design. This approach is called Partial TMR. Use of this approach requires making a trade-off between improvement in reliability and the cost of the replication. Authors in (RUANO; MAESTRO; REVIRIEGO, 2009) and (IMAGAWA et al., 2013) present methodologies to automatically select critical flip-flops for replication and obtain the desired SEU susceptibility level. On the other hand, authors in (MOHANRAM; TOUBA, 2003) target the partial replication of combinational logic paths. They employ the cluster sharing and the dominant value reduction methods in TMR circuits. The first reduction method carefully selects the nodes with low soft-error susceptibility to avoid performing replication over them. The second method exploits the fact that the soft-error susceptibility is related to the frequency of the determined logic value (0 or 1) in primary outputs. If a primary output exhibits low failure rate and it is frequently asserted to logic 1 (logic 0), only two copies are used along with the replacement of the 2-out-of-3 majority voter for an OR (AND) gate. The author in (TEIFEL, 2008) combines the DMR and TMR techniques in the same design as well. Interestingly, it employs self-voters built from a normal majority voter for the DMR parts.

Approximate TMR also exploits the benefits of logic masking properties to reduce the overhead caused by replication of combinational logic paths. The approximated replicas perform different but closely related logic functions. Consequently, approximate TMR can provide flexibility to get an optimal balance between fault coverage and the overhead in area an power (SANCHEZ-CLEMENTE et al., 2016; SANCHEZ-CLEMENTE; ENTRENA; GARCIA-VALDERAS, 2016; ARIFEEN et al., 2016; ALBANDES et al., 2018). Authors in (SANCHEZ-CLEMENTE et al., 2016) shows a reduction in area overhead of around 150% and reduction of 2% in the fault masking capability of the approximated TMR implementation in relation to the CGTMR implementation.

On the other side, the developed libraries are based on the command of Cadence tools but they can be adapted to be utilized with open-source synthesis tools such as ABC and Yosys (BRAYTON; MISHCHENKO, 2010; WOLF, 2012)

Experiments with the different TMR implementations of the ARM Cortex-M0 synthesized in FPGA were conducted to evaluate the behavior under the impact of heavy ions. The tests were performed using the 8UD Pelletron accelerator at Laboratório Aberto de Física Nuclear (LAFN) at Universidade de São Paulo. The results of the experiments will be exhibited in a future work.

In a collaborative project with the Carnegie Mellon University (Pittsburg, USA), the first case-study circuit and its TMR implementations were used to implement a test system chip that integrated all these circuits. The aim of the project was to expose the chip to accelerated radiation tests in order to analyze the behavior of the different implementations under the effects of radiation. The chip was fabricated using the FinFET 16nm technology. The dimensions of the die are 1.8 mm x 1.8 mm and the chips can operate with a maximum clock frequency of 1 GHz. The fault-free functionality of the chip was successfully tested and we expect to perform the radiation tests in the near future.

REFERENCES

AHLBIN, J. R. et al. Single-event transient pulse quenching in advanced CMOS logic circuits. In: **IEEE Transactions on Nuclear Science**. [S.l.: s.n.], 2009. v. 56, n. 6, p. 3050–3056. ISSN 00189499.

ALBANDES, I. et al. Improving approximate-TMR using multi-objective optimization genetic algorithm. In: **2018 IEEE 19th Latin-American Test Symposium (LATS)**. [S.l.]: IEEE, 2018. p. 1–6. ISBN 978-1-5386-1472-3.

AMUSAN, O. et al. Single Event Upsets in Deep-Submicrometer Technologies Due to Charge Sharing. **IEEE Transactions on Device and Materials Reliability**, v. 8, n. 3, p. 582–589, sep 2008. ISSN 1530-4388.

AMUSAN, O. A. et al. Single event upsets in a 130 nm hardened latch design due to charge sharing. In: **2007 IEEE International Reliability Physics Symposium Proceedings. 45th Annual**. [S.l.: s.n.], 2007. p. 306–311. ISSN 1541-7026.

ARIFEEN, T. et al. Probing Approximate TMR in Error Resilient Applications for Better Design Tradeoffs. In: **2016 Euromicro Conference on Digital System Design (DSD)**. [S.l.]: IEEE, 2016. p. 637–640. ISBN 978-1-5090-2817-7.

ARIMA, Y. et al. Cosmic-ray immune latch circuit for 90nm technology and beyond. In: **Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International**. [S.l.: s.n.], 2004. p. 492–493 Vol.1. ISSN 0193-6530.

ARM. **Arm ® Cortex ® -M0 DesignStart ™ Eval Revision: r2p0 User Guide Arm Cortex-M0 DesignStart Eval User Guide**. [S.l.], 2017.

AUTRAN, J. et al. Real-time soft-error rate measurements: A review. **Microelectronics Reliability**, Pergamon, v. 54, n. 8, p. 1455–1476, aug 2014. ISSN 0026-2714.

AZAMBUJA, J. R. et al. Evaluating large grain TMR and selective partial reconfiguration for soft error mitigation in SRAM based FPGAs. In: **2009 15th IEEE International On-Line Testing Symposium, IOLTS 2009**. [S.l.: s.n.], 2009. p. 101–106. ISBN 9781424445950.

BAGATIN, M. et al. Radiation environment in the ITER neutral beam injector prototype. **IEEE Transactions on Nuclear Science**, v. 59, n. 4 PART 1, p. 1099–1104, 2012. ISSN 00189499.

BAGATIN, M.; GERARDIN, S. **Ionizing radiation effects in electronics : from memories to imagers**. [S.l.: s.n.], 2015. 394 p. ISBN 9781498722605.

BALASUBRAMANIAN, P.; MASTORAKIS, N. Power, delay and area comparisons of majority voters relevant to tmr architectures. **Recent Advances in Circuits, Systems, Signal Processing and Communications**, 2015.

BALLAST, J. et al. A method for efficient Radiation Hardening of multicore processors. In: **IEEE Aerospace Conference Proceedings**. [S.l.: s.n.], 2015. v. 2015-June. ISBN 9781479953790. ISSN 1095323X.

BANNATYNE, R. et al. High temperature / radiation hardened capable ARM ® Cortex ® -M0 microcontrollers. International Microelectronics Assembly and Packaging Society, v. 2016, n. HiTEC, p. 46–50, may 2016. ISSN 2380-4491.

BAUMANN, R. C. Radiation-induced soft errors in advanced semiconductor technologies. **IEEE Transactions on Device and Materials Reliability**, v. 5, n. 3, p. 305–315, 2005. ISSN 15304388.

BELTRAME, G. Triple Modular Redundancy verification via heuristic netlist analysis. **PeerJ Computer Science**, PeerJ Inc., v. 1, p. e21, aug 2015. ISSN 2376-5992.

BENEVENUTI, F.; KASTENSMIDT, F. L. Reliability evaluation on interfacing with axi and axi-s on xilinx zynq-7000 ap-soc. In: **2018 IEEE 19th Latin-American Test Symposium (LATS)**. [S.l.: s.n.], 2018. p. 1–6.

BENITES, L. A. C.; KASTENSMIDT, F. L. Automated design flow for applying Triple Modular Redundancy (TMR) in complex digital circuits. In: **2018 IEEE 19th Latin-American Test Symposium (LATS)**. [S.l.]: IEEE, 2018. p. 1–4. ISBN 978-1-5386-1472-3.

BERG, M.; LABEL, K. A. **Verification of Triple Modular Redundancy (TMR) Insertion for Reliable and Trusted Systems**. [S.l.], 2016. 26p p.

BESSOT, D.; VELAZCO, R. Design of seu-hardened cmos memory cells: the hit cell. In: **Radiation and its Effects on Components and Systems, 1993.,RADECS 93., Second European Conference on**. [S.l.: s.n.], 1993. p. 563–570.

BLOUNT, P. The ITAR Treaty and Its Implications for U.S. Space Exploration Policy and the Commercial Space Industry. **Journal of Air Law and Commerce**, v. 73, n. 705, 2008. Available from Internet: <http://scholar.smu.edu/jalchttp://scholar.smu.edu/jalc>.

BOHR, M.; MISTRY, K. **Intel's Revolutionary 22 nm Transistor Technology**. [S.l.], 2011. Available from Internet: <https://www.intel.com/content/dam/www/public/us/en/documents/presentation/revolutionary-22nm-transistor-technology-presentation.pdf>.

BRAYTON, R.; MISHCHENKO, A. ABC: An Academic Industrial-Strength Verification Tool. In: . [S.l.]: Springer, Berlin, Heidelberg, 2010. p. 24–40.

BURLYAEV, D. **Design, Optimization, and Formal Verification of Circuit Fault-Tolerance Techniques**. Thesis (PhD) — Université de Grenoble, 2015.

CADENCE. **Equivalence Checking User Guide**. San Jose, CA, 2013. 480 p.

CADENCE. **Genus User Guide for Legacy UI**. San Jose, CA, 2016. 1–324 p.

CADENCE. **University Software Program**. 2018. Available from Internet: <https://www.cadence.com/content/cadence-www/global/en_US/home/services/cadence-academic-network/university-software-program.html>.

CALIN, T.; NICOLAIDIS, M.; VELAZCO, R. Upset hardened memory design for submicron cmos technology. **IEEE Transactions on Nuclear Science**, v. 43, n. 6, p. 2874–2878, Dec 1996. ISSN 0018-9499.

CAMPLANI, A. et al. Cmos ic radiation hardening by design. p. 251–258, June 2014.

CAO PAUL LEROUX, M. S. Y. **Radiation-Tolerant Delta-Sigma Time-to-Digital Converters**. [S.l.: s.n.], 2015.

CARMICHAEL, C.; CAFFREY, M.; SALAZAR, A. **Correcting Single-Event Upsets Through Virtex Partial Configuration**. [S.l.], 2000. 1–12 p.

CI-BRASIL. **Design Houses**. 2018. Available from Internet: <http://www.ci-brasil.gov.br/index.php/pt/design-houses>.

CLAEYS, C.; SIMOEN, E. **Radiation Effects in Advanced Semiconductor Materials and Devices**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. (Springer Series in Materials Science, v. 57). ISBN 978-3-642-07778-4.

CLEMENS, M. A. et al. The effects of neutron energy and high-Z materials on single event upsets and multiple cell upsets. In: **IEEE Transactions on Nuclear Science**. [S.l.: s.n.], 2011. v. 58, n. 6 PART 1, p. 2591–2598. ISBN 0018-9499. ISSN 00189499.

CRESSLER, J.; MANTOOTH, H. **Extreme Environment Electronics**. [S.l.]: Taylor & Francis, 2012. (Industrial Electronics). ISBN 9781439874301.

DIRK, J. et al. Terrestrial thermal neutrons. **IEEE Transactions on Nuclear Science**, v. 50, n. 6, p. 2060–2064, dec 2003. ISSN 0018-9499.

DODD, P. E.; MASSENGILL, L. W. Basic mechanisms and modeling of single-event upset in digital microelectronics. **IEEE Transactions on Nuclear Science**, v. 50 III, n. 3, p. 583–602, 2003. ISSN 00189499.

ENTRENA, L.; LÓPEZ, C.; OLÍAS, E. Automatic generation of fault tolerant VHDL designs in RTL. In: **Forum on Design Languages (2001)**. [S.l.: s.n.], 2001. p. 1–5.

FANG, Y.-p.; OATES, A. S. Characterization of Single Bit and Multiple Cell Soft Error Events in Planar and FinFET SRAMs. **IEEE Transactions on Device and Materials Reliability**, v. 16, n. 2, p. 132–137, June 2016. ISSN 1530-4388.

GORDON, M. et al. Measurement of the flux and energy spectrum of cosmic-ray induced neutrons on the ground. **IEEE Transactions on Nuclear Science**, v. 51, n. 6, p. 3427–3434, dec 2004. ISSN 0018-9499.

HABINC, S. **Functional Triple Modular Redundancy (FTMR)**. [S.l.], 2002. 1–56 p.

HAN, J. et al. Toward hardware-redundant, fault-tolerant logic for nanoelectronics. **IEEE Design Test of Computers**, v. 22, n. 4, p. 328–339, July 2005. ISSN 0740-7475.

HUGHES, H.; BENEDETTO, J. Radiation effects and hardening of MOS technology: devices and circuits. **IEEE Transactions on Nuclear Science**, v. 50, n. 3, p. 500–521, jun 2003. ISSN 0018-9499.

IBE, E. H. **Terrestrial Radiation Effects in ULSI Devices and Electronic Systems**. [S.l.: s.n.], 2015. 1–268 p. ISBN 9781118479308.

IMAGAWA, T. et al. A Cost-Effective Selective TMR for Heterogeneous Coarse-Grained Reconfigurable Architectures Based on DFG-Level Vulnerability Analysis. In: **Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013**. New Jersey: IEEE Conference Publications, 2013. p. 701–706. ISBN 9781467350716.

JEDEC. Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Error in Semiconductor Devices. **JEDEC Standard JESD89A**, n. October, p. 1–85, 2006.

KAJIHARA, S. et al. Combinationally irredundant ISCAS-89 benchmark circuits. In: **1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96**. [S.l.]: IEEE, 1989. v. 4, p. 632–634. ISBN 0-7803-3073-0.

KONONCHUK, O.; NGUYEN, B. **Silicon-On-Insulator (SOI) Technology: Manufacture and Applications**. [S.l.]: Elsevier Science, 2014. (Woodhead Publishing Series in Electronic and Optical Materials). ISBN 9780857099259.

KOREN, I.; KRISHNA, C. **Fault-Tolerant Systems**. [S.l.]: Elsevier Science, 2010. ISBN 9780080492681.

KULIS, S. Single Event Effects mitigation with TMRG tool. In: **Journal of Instrumentation**. [S.l.: s.n.], 2017. v. 12, n. 1. ISSN 17480221.

LALA, P. **Self-checking and Fault-tolerant Digital Design**. [S.l.]: Morgan Kaufmann, 2001. (The Morgan Kaufmann Series in Computer Architecture and Design Series). ISBN 9780124343702.

LEE, G. et al. TLegUp: A TMR code generation tool for SRAM-based FPGA applications using HLS. In: **Proceedings - IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2017**. [S.l.: s.n.], 2017. p. 129–132. ISBN 9781538640364.

LEVEUGLE, R. Automatic modifications of high level VHDL descriptions for fault detection or tolerance. In: **Proceedings -Design, Automation and Test in Europe, DATE**. [S.l.: s.n.], 2002. p. 837–841. ISBN 0-7695-1471-5. ISSN 15301591.

MAHATME, N. N. et al. Impact of strained-Si PMOS transistors on SRAM soft error rates. **IEEE Transactions on Nuclear Science**, v. 59, n. 4 PART 1, p. 845–850, 2012. ISSN 00189499.

MAVIS, D. G.; EATON, P. H. Soft error rate mitigation techniques for modern microcircuits. In: **2002 IEEE International Reliability Physics Symposium. Proceedings. 40th Annual (Cat. No.02CH37320)**. [S.l.: s.n.], 2002. p. 216–225.

MELANY, B.; LABEL, K.; PELLISH, J. **Susceptibility of Redundant Versus Singular Clock Domains Implemented in SRAM-Based FPGA TMR Designs**. Monterey, 2016.

MENTOR. **Precision® Hi-Rel Advanced FPGA Synthesis Datasheet**. [S.l.], 2018.

MITRA, S. et al. Soft error resilient system design through error correction. In: **2006 IFIP International Conference on Very Large Scale Integration**. [S.l.: s.n.], 2006. p. 332–337. ISSN 2324-8432.

MOHANRAM, K.; TOUBA, N. Partial error masking to reduce soft error failure rate in logic circuits. In: **Proceedings. 16th IEEE Symposium on Computer Arithmetic**. [S.l.]: IEEE Comput. Soc, 2003. p. 433–440. ISBN 0-7695-2042-1.

MONTENEGRO, S.; PETROVIC, V.; SCHOOF, G. Network Centric Systems for Space Applications. In: **2010 Second International Conference on Advances in Satellite and Space Communications**. IEEE, 2010. p. 146–150. ISBN 978-1-4244-7275-8. Available from Internet: <http://ieeexplore.ieee.org/document/5502778/>.

MOOR, J. P.; DE, P. Ionizing Radiation Effects in Electronics: From Memories to Imagers. **Solid State Technology**, p. 394, 2011.

NASA. **The van Allen Belts**. 2018. Available from Internet: <https://image.gsfc.nasa.gov/poetry/tour/AAvan.html>.

NICOLAIDIS, M. Design for soft error mitigation. **IEEE Transactions on Device and Materials Reliability**, v. 5, n. 3, p. 405–418, sep 2005. ISSN 1530-4388. Available from Internet: <http://ieeexplore.ieee.org/document/1545900/>.

NIKNAHAD, M. **Using Fine Grain Approaches for highly reliable Design of FPGA-based Systems in Space**. [S.l.]: KIT Scientific Publishing, 2012. ISBN 9783731500384.

NORMAND, E.; BAKER, T. Altitude and latitude variations in avionics SEU and atmospheric neutron flux. **IEEE Transactions on Nuclear Science**, v. 40, n. 6, p. 1484–1490, 1993. ISSN 00189499.

OSADA, K. et al. SRAM immunity to cosmic-ray-induced multierrors based on analysis of an induced parasitic bipolar effect. **IEEE Journal of Solid-State Circuits**, v. 39, n. 5, p. 827–833, may 2004. ISSN 0018-9200.

OUSTERHOUT, J. K. et al. **Tcl and the Tk Toolkit**. 2. ed. Upper Saddle River, New Jersey: Addison-Wesley, 2010. (Addision-Wesley Professional Computing Series). ISBN 978-0-321-33633-0.

ÖZER, E.; Ghahroodi, M, M.; BULL, D. SEU and SET-tolerant ARM Cortex-R4 CPU for Space and Avionics Applications. 2013.

PRATT, B. et al. Improving FPGA Design Robustness with Partial TMR. In: **2005 MAPLD International Conference**. Washington, D.C.: [s.n.], 2005.

RABAEY, J. M.; CHANDRAKASAN, A. P.; NIKOLIC, B. **Digital integrated circuits: a design perspective**. [S.l.]: Pearson Education, 2003. (Prentice Hall electronics and VLSI series).

RAO, R. R.; BLAAUW, D.; SYLVESTER, D. Soft error reduction in combinational logic using gate resizing and flipflop selection. In: **IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD**. [S.l.: s.n.], 2006. p. 502–509. ISBN 1595933891. ISSN 10923152.

RUANO, O.; MAESTRO, J. A.; REVIRIEGO, P. A methodology for automatic insertion of selective TMR in digital circuits affected by SEUs. In: **IEEE Transactions on Nuclear Science**. [S.l.: s.n.], 2009. v. 56, n. 4, p. 2091–2102. ISBN 1550-5774. ISSN 00189499.

SANCHEZ-CLEMENTE, A. J.; ENTRENA, L.; GARCIA-VALDERAS, M. Partial TMR in FPGAs Using Approximate Logic Circuits. **IEEE Transactions on Nuclear Science**, v. 63, n. 4, p. 2233–2240, aug 2016. ISSN 0018-9499.

SANCHEZ-CLEMENTE, A. J. et al. Error Mitigation Using Approximate Logic Circuits: A Comparison of Probabilistic and Evolutionary Approaches. **IEEE Transactions on Reliability**, v. 65, n. 4, p. 1871–1883, dec 2016. ISSN 0018-9529.

SANTOS, A. F. dos et al. Applying TMR in Hardware Accelerators Generated by High-Level Synthesis Design Flow for Mitigating Multiple Bit Upsets in SRAM-Based FPGAs. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2017. v. 10216 LNCS, p. 202–213.

SCHRIMPF, R.; FLEETWOOD, D. **Radiation Effects and Soft Errors in Integrated Circuits and Electronic Devices**. [S.l.]: World Scientific Pub., 2004. (International journal of high speed electronics and systems). ISBN 9789812389404.

SCHWANK, J. R. et al. Radiation effects in soi technologies. **IEEE Transactions on Nuclear Science**, v. 50, n. 3, p. 522–538, June 2003. ISSN 0018-9499.

SCHWANK, J. R. et al. Radiation effects in MOS oxides. In: **IEEE Transactions on Nuclear Science**. [S.l.: s.n.], 2008. v. 55, n. 4, p. 1833–1853. ISBN 0018-9499. ISSN 00189499.

SECRETARY, E. **Technique for radiation effects mitigation in ASICs and FPGAs handbook**. [S.l.: s.n.], 2016.

SESHIA, S. A.; LI, W.; MITRA, S. Verification-Guided Soft Error Resilience. In: **2007 Design, Automation & Test in Europe Conference & Exhibition**. IEEE, 2007. p. 1–6. ISBN 978-3-9810801-2-4. Available from Internet: <http://ieeexplore.ieee.org/document/4212011/>.

SHELDON, D. Selection and qualification of foundries for cmos integrated circuits. p. 1–10, May 2005.

SHULER, R. L. Porting and scaling strategies for nanoscale cmos rhbd. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v. 62, n. 12, p. 2856–2863, Dec 2015. ISSN 1549-8328.

SHULER, R. L. et al. Comparison of dual-rail and tmr logic cost effectiveness and suitability for fpgas with reconfigurable seu tolerance. **IEEE Transactions on Nuclear Science**, v. 56, n. 1, p. 214–219, Feb 2009. ISSN 0018-9499.

SROUR, J.; MARSHALL, C.; MARSHALL, P. Review of displacement damage effects in silicon devices. **IEEE Transactions on Nuclear Science**, v. 50, n. 3, p. 653–670, 2003. ISSN 0018-9499.

STAMENKOVIć, Z.; PETROVIć, V.; SCHOOF, G. Design flow and techniques for fault-tolerant asic. In: **Proceedings of the 20th IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)**. [S.l.: s.n.], 2013. p. 93–98. ISSN 1946-1542.

STAMENKOVIC, Z.; PETROVIC, V.; SCHOOF, G. Design flow and techniques for fault-tolerant ASIC. In: **Proceedings of the 20th IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)**. IEEE, 2013. p. 93–98. ISBN 978-1-4799-0480-8. Available from Internet: <http://ieeexplore.ieee.org/document/6599133/>.

SYNOPSYS. **Latest Synplify FPGA Synthesis Software Offers New High-Reliability Features and Improves Productivity for FPGA-Based Prototyping**. 2012. Available from Internet: <https://dac.com/media-center/exhibitor-news/latest-synplify-fpga-synthesis-software-offers-new-high-reliability>.

TAKEDA, E.; HISAMOTO, D.; TOYABE, T. A new soft-error phenomenon in VLSIs: the alpha-particle-induced source/drain penetration (ALPEN) effect. In: **26th Annual Proceedings Reliability Physics Symposium 1988**. [S.l.]: IEEE, 1988. p. 109–112.

TEIFEL, J. Self-voting dual-modular-redundancy circuits for single-event-transient mitigation. **IEEE Transactions on Nuclear Science**, v. 55, n. 6, p. 3435–3439, Dec 2008. ISSN 0018-9499.

TONFAT, J. et al. Method to analyze the susceptibility of HLS designs in SRAM-based FPGAs under soft errors. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2016. v. 9625, p. 132–143. ISBN 9783319304809. ISSN 16113349.

TONFAT, J. et al. Multiple fault injection platform for SRAM-based FPGA based on ground-level radiation experiments. In: **FPGAs and Parallel Architectures for Aerospace Applications: Soft Errors and Fault-Tolerant Design**. [S.l.: s.n.], 2015. p. 135–151. ISBN 9783319143521.

UEMURA, T. et al. Seila: Soft error immune latch for mitigating multi-node-seu and local-clock-set. In: **Reliability Physics Symposium (IRPS), 2010 IEEE International**. [S.l.: s.n.], 2010. p. 218–223. ISSN 1541-7026.

Van Allen, J. A.; FRANK, L. A. Radiation around the earth to a radial distance of 107,400 km. **Nature**, v. 183, n. 4659, p. 430–434, 1959. ISSN 00280836.

WANG, W.; GONG, H. Edge triggered pulse latch design with delayed latching edge for radiation hardened application. **IEEE Transactions on Nuclear Science**, v. 51, n. 6, p. 3626–3630, Dec 2004. ISSN 0018-9499.

WEAVER, H. T. et al. An seu tolerant memory cell derived from fundamental studies of seu mechanisms in sram. **IEEE Transactions on Nuclear Science**, v. 34, n. 6, p. 1281–1286, 1987. ISSN 15581578.

WESTE, N. H. E.; HARRIS, D. **CMOS VLSI design: a circuits and systems perspective**. [S.l.]: Pearson Education, 2015.

WILKINSON, J. **Neutron Flux Calculation**. 2006. Available from Internet: <http://www.seutest.com/cgi-bin/FluxCalculator.cgi>.

WILSON, A. et al. Radiation and reliability characterization of a multiplexer family using a 0.35 um triple-well cmos technology. In: **2011 IEEE Radiation Effects Data Workshop**. [S.l.: s.n.], 2011. p. 1–7. ISSN 2154-0519.

WOLF, C. **Yosys Manual**. [S.l.], 2012.

XILINX. **Zynq-7000 All Programmable SoC Architecture Porting Quick Start Guide**. [S.l.], 2015. UG1181, 1–26 p. Available from Internet: <https://www.xilinx.com/support/documentation/user{_}guides/ug1181-zynq-7000-architecture-porting.>

XILINX. **Xilinx TMRTool User Guide: TMRTool Software Version 13.2 User Guide (UG156)**. [S.l.], 2017. Available from Internet: <https://www.xilinx.com/support/documentation/user{_}guides/ug156-tmrtool.>

ZHANG, M. Thermal neutron SER testing and analysis: findings from a 32nm HKMG SRAM case study. In: **2011 IEEE Workshop on Silicon Errors in Logic – System Effects, Champaign, IL,**. [S.l.: s.n.], 2011. p. 7–10.

ZIEGLER, J. et al. IBM experiments in soft fails in computer electronics (1978-1994). **IBM Journal of Research and Development**, v. 40, n. 1, p. 3–18, 1996. ISSN 0018-8646.

APPENDIX A — TMR DESIGN EXAMPLE

The aim of this design example is to introduce the use of the proposed approach in this master thesis to automate the design of TMR circuits employing Cadence tools. Simple scripts are shown for the Genus and LEC tools in order to perform the implementation and verification steps in the automated TMR design. Furthermore, the generated files through the steps of implementations and verification are included in order to show the data used in this work.

The circuit s27 from the ISCAS'89 benchmark set (KAJIHARA et al., 1989) was utilized in order to illustrate the steps of implementation and verification of the devised automated TMR design. It was chosen because of its reduce number of gates in order to ease the visualization of the information contained in the generated files.

Implementation and structural verification of TMR designs

```
#########################################################
# Template file for Cadence Genus synthesis tool
#########################################################

# Name of top module (unmitigated circuit)
set top_design "s27"

# Path to library
set_attribute lib_search_path "$DIR_LIB_NANGATE_CMOS_45nm" /

# Path to netlist directory
set_attribute hdl_search_path "$DIR_NETLIST" /

# Read library
set_attribute library "$LIB_NANGATE_CMOS_45nm" /

# Read netlists (unmitigated and voter circuits)
read_netlist "$top_design.v"
read_netlist "myvoter.v"

#########################################################
# TMR implementation (TMRi)
#########################################################

# Load TMRi library
include "TMRi_library/*.tcl"

# TMR granularity (cgtmr/fgltmr/fgdtmr)
TMRi_CGTMR "$top_design" "$top_design\_cgtmr" "myvoter"
TMRi_FGLTMR "$top_design" "$top_design\_fgltmr" "myvoter"
TMRi_FGDTMR "$top_design" "$top_design\_fgdtmr" "myvoter"

# Write out generated TMR netlists
write_hdl -v2001 $top_design\_cgtmr > "$top_design\_cgtmr.v"
write_hdl -v2001 $top_design\_fgltmr > "$top_design\_fgltmr.v"
write_hdl -v2001 $top_design\_fgdtmr > "$top_design\_fgdtmr.v"

#########################################################
# TMR verification (TMRv): structural verification
#########################################################

# Load TMRv library for Genus tool
include "TMRv_library_genus/*.tcl"

# Structural verification of cgtmr and fgdtmr
TMRv_replication_ff [array get "$top_design\_cgtmr_fgdtmr_ff"]
TMRv_replication_cmb_logic_ff [array get "$top_design\_cgtmr_fgdtmr_ff"]
TMRv_replication_cmb_logic_po "$top_design\_list_ports_out_top"

# Structural verification of fgltmr
TMRv_replication_ff [array get "$top_design\_fgltmr_ff"]
```

```
TMRv_replication_cmb_logic_ff [array get "$top_design\_fgltmr_ff"]
TMRv_replication_cmb_logic_po "$top_design\_list_ports_out_top"
```

scripts/genus_script.tcl

Logical and masking capability verification of TMR designs

```tcl
#########################################################
# Template for Cadence LEC tool
#########################################################

# LEC tool in tcl mode
tclmode

# Name of top module (unmitigated circuit)
set top_design "s27"

# TMR granularity (cgtmr/fgltmr/fgdtmr)
set tmr_version "fgdtmr"
#set tmr_version "fgltmr"
#set tmr_version "fgdtmr"

# Path to library directory
vpx add search path -library "$DIR_LIB_NANGATE_CMOS_45nm"

# Path to netlist directory
vpx add search path -design -both "$DIR_NETLIST"

# Read library
vpx read library -statetable -liberty -both "$LIB_NANGATE_CMOS_45nm"

# Read netlists (unmitigated and TMR circuits)
vpx read design -verilog2k -golden -lastmod -noelab "$top_module.v"
vpx read design -verilog2k -revised -lastmod -noelab "$top_module\_$tmr_version.v"

# Elaborate designs (unmitigated and TMR designs)
vpx elaborate design -golden -root "$top_module"
vpx elaborate design -revised -root "$top_module\_$tmr_version"

#########################################################
# TMR verification (TMRv):
#   logical and masking capability verification
#########################################################

# Load TMRv library for LEC tool
source "TMRv_library_lec/*.tcl"

# TMR granularity (CGTMR/FGLTMR/FGDTMR)
TMRv_lec_CGTMR
#TMRv_lec_FGLTMR
#TMRv_lec_FGDTMR
```

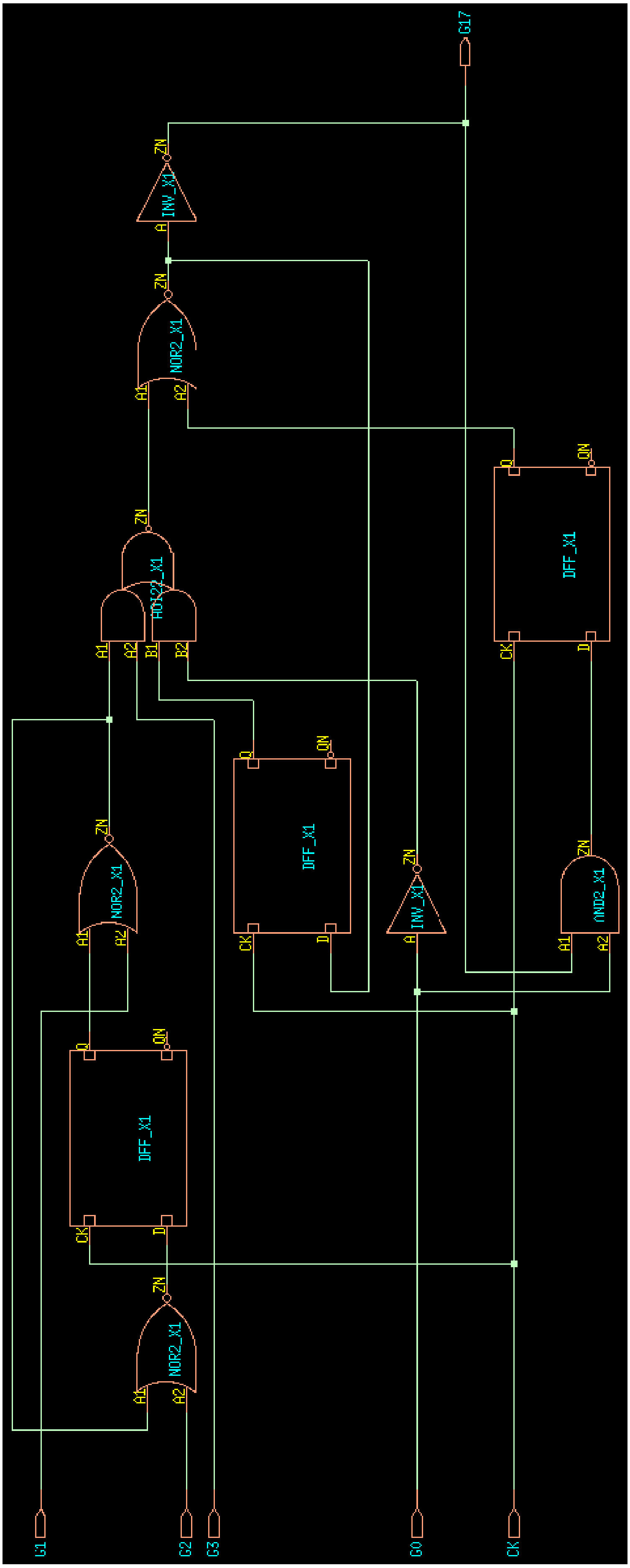
scripts/lec_script.tcl

Unmitigated netlist

```
// Generated by Cadence Genus(TM) Synthesis Solution GENUS15.22 - 15.20-s024_1

module s27(CK, G0, G1, G17, G2, G3);
  input CK, G0, G1, G2, G3;
  output G17;
  wire CK, G0, G1, G2, G3;
  wire G17;
  wire G5, G6, G7, n_0, n_1, n_2, n_3, n_4;
  wire n_6;
  DFF_X1 DFF_0_I1_Q_reg(.CK (CK), .D (n_6), .Q (G5), .QN ());
  AND2_X1 g76(.A1 (G17), .A2 (G0), .ZN (n_6));
  DFF_X1 DFF_1_I1_Q_reg(.CK (CK), .D (n_4), .Q (G6), .QN ());
  INV_X1 g77(.A (n_4), .ZN (G17));
  NOR2_X1 g78(.A1 (n_3), .A2 (G5), .ZN (n_4));
  DFF_X1 DFF_2_I1_Q_reg(.CK (CK), .D (n_2), .Q (G7), .QN ());
  AOI22_X1 g80(.A1 (n_1), .A2 (G3), .B1 (G6), .B2 (n_0), .ZN (n_3));
  NOR2_X1 g81(.A1 (n_1), .A2 (G2), .ZN (n_2));
  NOR2_X1 g82(.A1 (G7), .A2 (G1), .ZN (n_1));
  INV_X1 g83(.A (G0), .ZN (n_0));
endmodule
```

benchmark/s27/s27.v

Figure A.1: Unmitigated s27 circuit



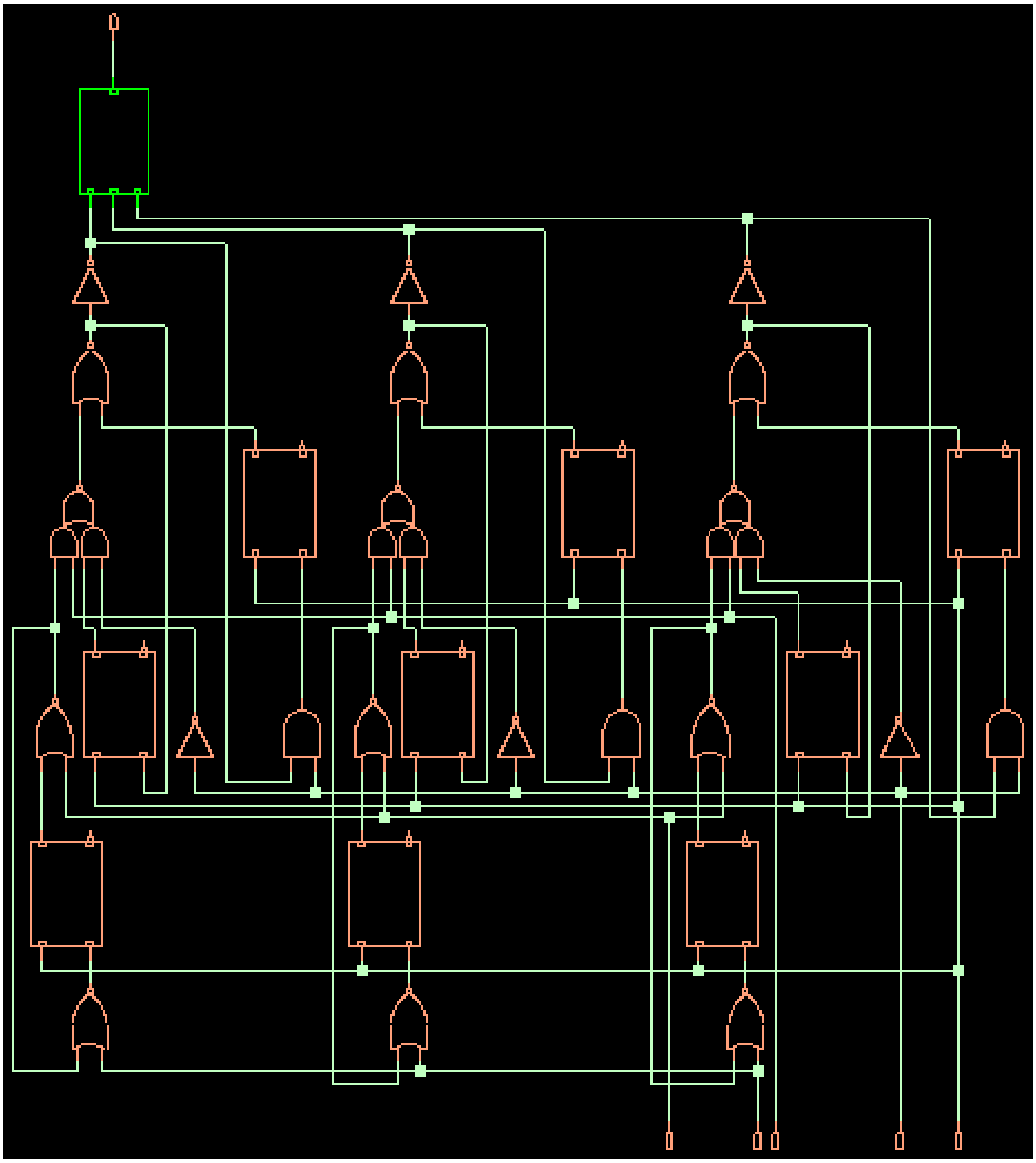
Source: From the author

Resulting CGTMR netlist

```
// Generated by Cadence Genus(TM) Synthesis Solution GENUS15.22 - 15.20-s024_1

module s27_cgtmr(CK, G0, G1, G2, G3, G17);
  input CK, G0, G1, G2, G3;
  output G17;
  wire CK, G0, G1, G2, G3;
  wire G17;
  wire U0_G5, U0_G6, U0_G7, U0_n_0, U0_n_1, U0_n_2, U0_n_3, U0_n_4;
  wire U0_n_6, U1_G5, U1_G6, U1_G7, U1_n_0, U1_n_1, U1_n_2, U1_n_3;
  wire U1_n_4, U1_n_6, U2_G5, U2_G6, U2_G7, U2_n_0, U2_n_1, U2_n_2;
  wire U2_n_3, U2_n_4, U2_n_6, n_6, n_7, n_8;
  myvoter g1(.in0 (n_6), .in1 (n_7), .in2 (n_8), .out (G17));
  DFF_X1 U0_DFF_0_I1_Q_reg(.CK (CK), .D (U0_n_6), .Q (U0_G5), .QN ());
  AND2_X1 U0_g76(.A1 (n_6), .A2 (G0), .ZN (U0_n_6));
  DFF_X1 U0_DFF_1_I1_Q_reg(.CK (CK), .D (U0_n_4), .Q (U0_G6), .QN ());
  INV_X1 U0_g77(.A (U0_n_4), .ZN (n_6));
  NOR2_X1 U0_g78(.A1 (U0_n_3), .A2 (U0_G5), .ZN (U0_n_4));
  DFF_X1 U0_DFF_2_I1_Q_reg(.CK (CK), .D (U0_n_2), .Q (U0_G7), .QN ());
  AOI22_X1 U0_g80(.A1 (U0_n_1), .A2 (G3), .B1 (U0_G6), .B2 (U0_n_0),
       .ZN (U0_n_3));
  NOR2_X1 U0_g81(.A1 (U0_n_1), .A2 (G2), .ZN (U0_n_2));
  NOR2_X1 U0_g82(.A1 (U0_G7), .A2 (G1), .ZN (U0_n_1));
  INV_X1 U0_g83(.A (G0), .ZN (U0_n_0));
  DFF_X1 U1_DFF_0_I1_Q_reg(.CK (CK), .D (U1_n_6), .Q (U1_G5), .QN ());
  AND2_X1 U1_g76(.A1 (n_7), .A2 (G0), .ZN (U1_n_6));
  DFF_X1 U1_DFF_1_I1_Q_reg(.CK (CK), .D (U1_n_4), .Q (U1_G6), .QN ());
  INV_X1 U1_g77(.A (U1_n_4), .ZN (n_7));
  NOR2_X1 U1_g78(.A1 (U1_n_3), .A2 (U1_G5), .ZN (U1_n_4));
  DFF_X1 U1_DFF_2_I1_Q_reg(.CK (CK), .D (U1_n_2), .Q (U1_G7), .QN ());
  AOI22_X1 U1_g80(.A1 (U1_n_1), .A2 (G3), .B1 (U1_G6), .B2 (U1_n_0),
       .ZN (U1_n_3));
  NOR2_X1 U1_g81(.A1 (U1_n_1), .A2 (G2), .ZN (U1_n_2));
  NOR2_X1 U1_g82(.A1 (U1_G7), .A2 (G1), .ZN (U1_n_1));
  INV_X1 U1_g83(.A (G0), .ZN (U1_n_0));
  DFF_X1 U2_DFF_0_I1_Q_reg(.CK (CK), .D (U2_n_6), .Q (U2_G5), .QN ());
  AND2_X1 U2_g76(.A1 (n_8), .A2 (G0), .ZN (U2_n_6));
  DFF_X1 U2_DFF_1_I1_Q_reg(.CK (CK), .D (U2_n_4), .Q (U2_G6), .QN ());
  INV_X1 U2_g77(.A (U2_n_4), .ZN (n_8));
  NOR2_X1 U2_g78(.A1 (U2_n_3), .A2 (U2_G5), .ZN (U2_n_4));
  DFF_X1 U2_DFF_2_I1_Q_reg(.CK (CK), .D (U2_n_2), .Q (U2_G7), .QN ());
  AOI22_X1 U2_g80(.A1 (U2_n_1), .A2 (G3), .B1 (U2_G6), .B2 (U2_n_0),
       .ZN (U2_n_3));
  NOR2_X1 U2_g81(.A1 (U2_n_1), .A2 (G2), .ZN (U2_n_2));
  NOR2_X1 U2_g82(.A1 (U2_G7), .A2 (G1), .ZN (U2_n_1));
  INV_X1 U2_g83(.A (G0), .ZN (U2_n_0));
endmodule

module myvoter(in0, in1, in2, out);
  input in0, in1, in2;
  output out;
  wire in0, in1, in2;
  wire out;
  wire n_0, n_1;
  NAND2_X1 g28(.A1 (n_1), .A2 (n_0), .ZN (out));
  OAI21_X1 g29(.A (in2), .B1 (in0), .B2 (in1), .ZN (n_1));
  NAND2_X1 g30(.A1 (in0), .A2 (in1), .ZN (n_0));
endmodule
```

benchmark/s27/s27_cgtmr.v

Figure A.2: CGTMR s27 circuit



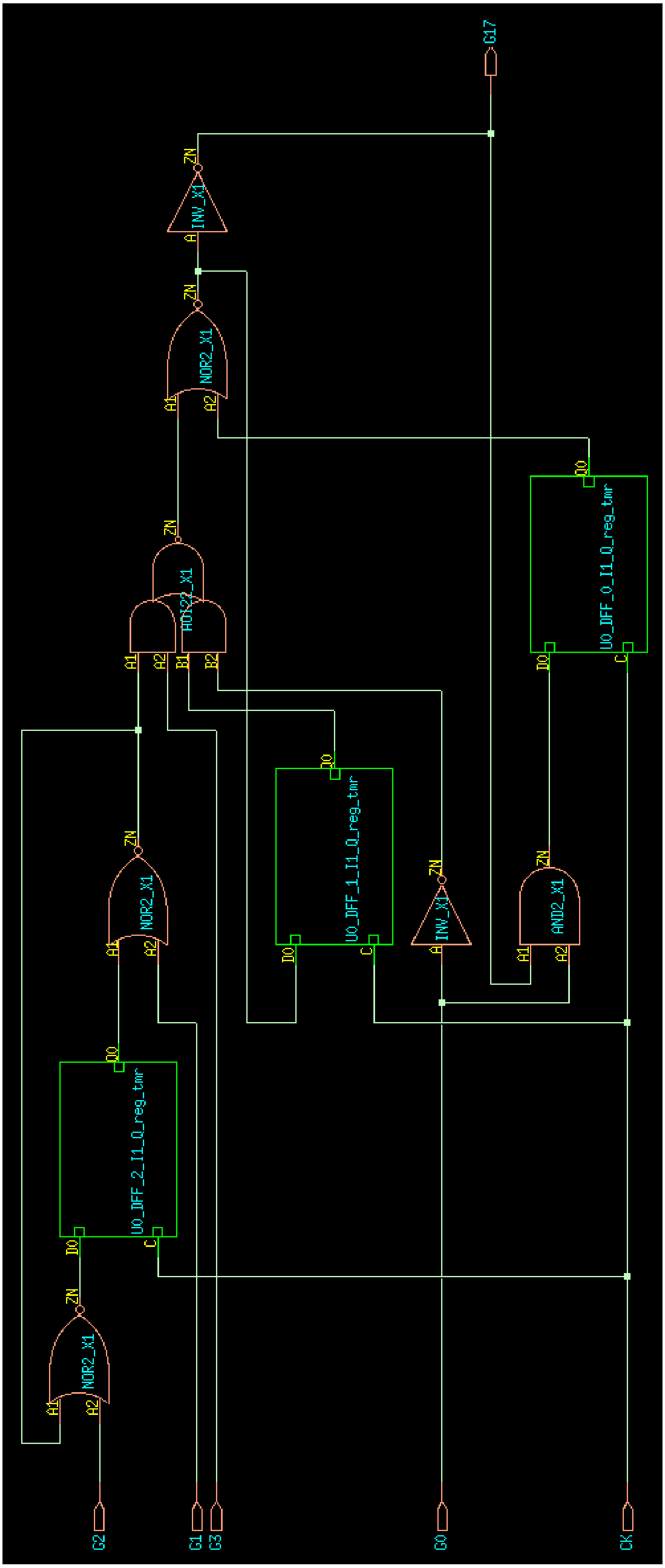
Source: From the author

Resulting FGLTMR netlist

```verilog
// Generated by Cadence Genus(TM) Synthesis Solution GENUS15.22 - 15.20-s024_1

module s27_fgltmr(CK, G0, G1, G2, G3, G17);
  input CK, G0, G1, G2, G3;
  output G17;
  wire CK, G0, G1, G2, G3;
  wire G17;
  wire U0_G5, U0_G6, U0_G7, U0_n_0, U0_n_1, U0_n_2, U0_n_3, U0_n_4;
  wire U0_n_6;
  libcell_TMR_0 U0_DFF_0_I1_Q_reg_tmr(.D0 (U0_n_6), .C (CK), .Q0
       (U0_G5));
  libcell_TMR_1 U0_DFF_1_I1_Q_reg_tmr(.D0 (U0_n_4), .C (CK), .Q0
       (U0_G6));
  libcell_TMR_2 U0_DFF_2_I1_Q_reg_tmr(.D0 (U0_n_2), .C (CK), .Q0
       (U0_G7));
  AND2_X1 U0_g76(.A1 (G17), .A2 (G0), .ZN (U0_n_6));
  INV_X1 U0_g77(.A (U0_n_4), .ZN (G17));
  NOR2_X1 U0_g78(.A1 (U0_n_3), .A2 (U0_G5), .ZN (U0_n_4));
  AOI22_X1 U0_g80(.A1 (U0_n_1), .A2 (G3), .B1 (U0_G6), .B2 (U0_n_0),
       .ZN (U0_n_3));
  NOR2_X1 U0_g81(.A1 (U0_n_1), .A2 (G2), .ZN (U0_n_2));
  NOR2_X1 U0_g82(.A1 (U0_G7), .A2 (G1), .ZN (U0_n_1));
  INV_X1 U0_g83(.A (G0), .ZN (U0_n_0));
endmodule

module myvoter(in0, in1, in2, out);
  input in0, in1, in2;
  output out;
  wire in0, in1, in2;
  wire out;
  wire n_0, n_1;
  NAND2_X1 g73(.A1 (n_1), .A2 (n_0), .ZN (out));
  OAI21_X1 g74(.A (in2), .B1 (in0), .B2 (in1), .ZN (n_1));
  NAND2_X1 g75(.A1 (in0), .A2 (in1), .ZN (n_0));
endmodule

module libcell_TMR_0(D0, C, Q0);
  input D0, C;
  output Q0;
  wire D0, C;
  wire Q0;
  wire n_1, n_2, n_3;
  myvoter voter_0(.in0 (n_1), .in1 (n_2), .in2 (n_3), .out (Q0));
  DFF_X1 ff_0(.CK (C), .D (D0), .Q (n_1), .QN ());
  DFF_X1 ff_1(.CK (C), .D (D0), .Q (n_2), .QN ());
  DFF_X1 ff_2(.CK (C), .D (D0), .Q (n_3), .QN ());
endmodule

module libcell_TMR_1(D0, C, Q0);
  input D0, C;
  output Q0;
  wire D0, C;
  wire Q0;
  wire n_1, n_2, n_3;
  myvoter voter_0(.in0 (n_1), .in1 (n_2), .in2 (n_3), .out (Q0));
  DFF_X1 ff_0(.CK (C), .D (D0), .Q (n_1), .QN ());
  DFF_X1 ff_1(.CK (C), .D (D0), .Q (n_2), .QN ());
  DFF_X1 ff_2(.CK (C), .D (D0), .Q (n_3), .QN ());
endmodule

module libcell_TMR_2(D0, C, Q0);
  input D0, C;
  output Q0;
  wire D0, C;
  wire Q0;
  wire n_1, n_2, n_3;
  myvoter voter_0(.in0 (n_1), .in1 (n_2), .in2 (n_3), .out (Q0));
  DFF_X1 ff_0(.CK (C), .D (D0), .Q (n_1), .QN ());
  DFF_X1 ff_1(.CK (C), .D (D0), .Q (n_2), .QN ());
  DFF_X1 ff_2(.CK (C), .D (D0), .Q (n_3), .QN ());
endmodule
```

benchmark/s27/s27_fgltmr.v

Figure A.3: FGLTMR s27 circuit



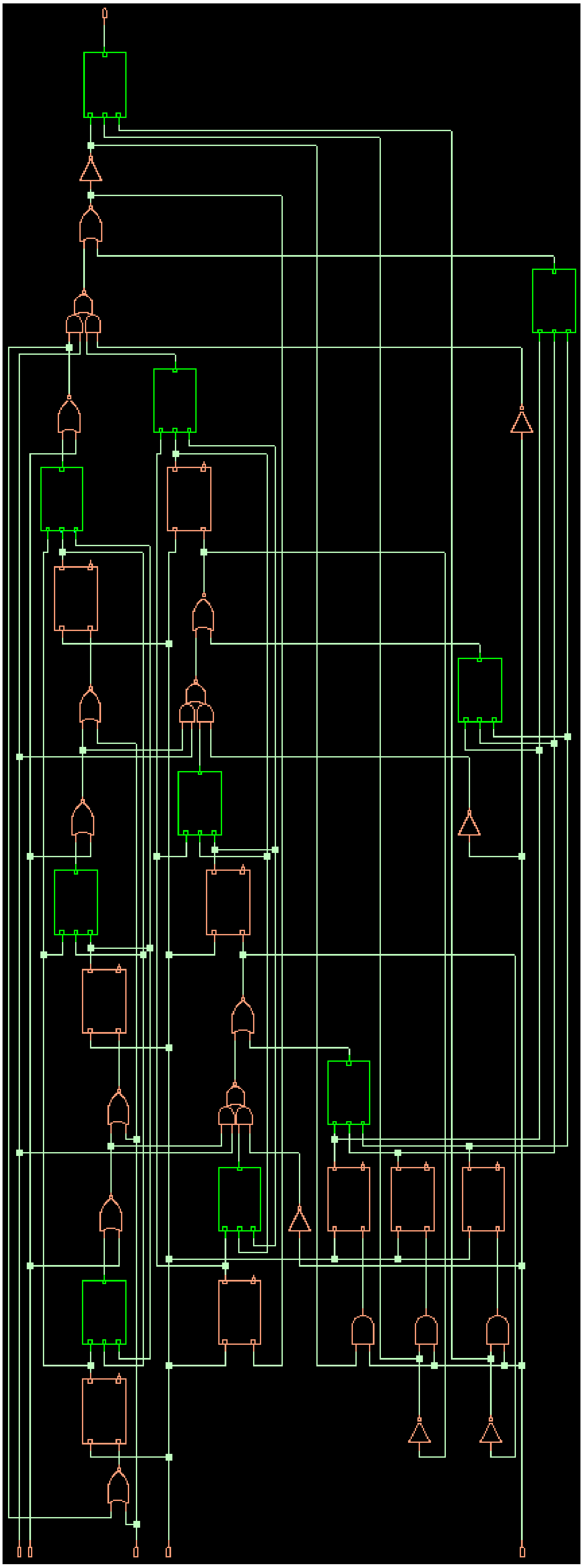
Source: From the author

Resulting FGDTMR netlist

```
// Generated by Cadence Genus(TM) Synthesis Solution GENUS15.22 - 15.20-s024_1

module s27_fgdtmr(CK, G0, G1, G2, G3, G17);
  input CK, G0, G1, G2, G3;
  output G17;
  wire CK, G0, G1, G2, G3;
  wire G17;
  wire U0_n_0, U0_n_1, U0_n_2, U0_n_3, U0_n_4, U0_n_6, U1_n_0, U1_n_1;
  wire U1_n_2, U1_n_3, U1_n_4, U1_n_6, U2_n_0, U2_n_1, U2_n_2, U2_n_3;
  wire U2_n_4, U2_n_6, n_6, n_7, n_8, n_10, n_11, n_12;
  wire n_13, n_14, n_15, n_16, n_17, n_18, n_19, n_20;
  wire n_21, n_22, n_23, n_24, n_25, n_26, n_27;
  myvoter g1(.in0 (n_6), .in1 (n_7), .in2 (n_8), .out (G17));
  myvoter g2(.in0 (n_13), .in1 (n_14), .in2 (n_15), .out (n_10));
  myvoter g3(.in0 (n_13), .in1 (n_14), .in2 (n_15), .out (n_11));
  myvoter g4(.in0 (n_13), .in1 (n_14), .in2 (n_15), .out (n_12));
  myvoter g5(.in0 (n_19), .in1 (n_20), .in2 (n_21), .out (n_16));
  myvoter g6(.in0 (n_19), .in1 (n_20), .in2 (n_21), .out (n_17));
  myvoter g7(.in0 (n_19), .in1 (n_20), .in2 (n_21), .out (n_18));
  myvoter g8(.in0 (n_25), .in1 (n_26), .in2 (n_27), .out (n_22));
  myvoter g9(.in0 (n_25), .in1 (n_26), .in2 (n_27), .out (n_23));
  myvoter g10(.in0 (n_25), .in1 (n_26), .in2 (n_27), .out (n_24));
  DFF_X1 U0_DFF_0_I1_Q_reg(.CK (CK), .D (U0_n_6), .Q (n_13), .QN ());
  AND2_X1 U0_g76(.A1 (n_6), .A2 (G0), .ZN (U0_n_6));
  DFF_X1 U0_DFF_1_I1_Q_reg(.CK (CK), .D (U0_n_4), .Q (n_19), .QN ());
  INV_X1 U0_g77(.A (U0_n_4), .ZN (n_6));
  NOR2_X1 U0_g78(.A1 (U0_n_3), .A2 (n_10), .ZN (U0_n_4));
  DFF_X1 U0_DFF_2_I1_Q_reg(.CK (CK), .D (U0_n_2), .Q (n_25), .QN ());
  AOI22_X1 U0_g80(.A1 (U0_n_1), .A2 (G3), .B1 (n_16), .B2 (U0_n_0), .ZN
       (U0_n_3));
  NOR2_X1 U0_g81(.A1 (U0_n_1), .A2 (G2), .ZN (U0_n_2));
  NOR2_X1 U0_g82(.A1 (n_22), .A2 (G1), .ZN (U0_n_1));
  INV_X1 U0_g83(.A (G0), .ZN (U0_n_0));
  DFF_X1 U1_DFF_0_I1_Q_reg(.CK (CK), .D (U1_n_6), .Q (n_14), .QN ());
  AND2_X1 U1_g76(.A1 (n_7), .A2 (G0), .ZN (U1_n_6));
  DFF_X1 U1_DFF_1_I1_Q_reg(.CK (CK), .D (U1_n_4), .Q (n_20), .QN ());
  INV_X1 U1_g77(.A (U1_n_4), .ZN (n_7));
  NOR2_X1 U1_g78(.A1 (U1_n_3), .A2 (n_11), .ZN (U1_n_4));
  DFF_X1 U1_DFF_2_I1_Q_reg(.CK (CK), .D (U1_n_2), .Q (n_26), .QN ());
  AOI22_X1 U1_g80(.A1 (U1_n_1), .A2 (G3), .B1 (n_17), .B2 (U1_n_0), .ZN
       (U1_n_3));
  NOR2_X1 U1_g81(.A1 (U1_n_1), .A2 (G2), .ZN (U1_n_2));
  NOR2_X1 U1_g82(.A1 (n_23), .A2 (G1), .ZN (U1_n_1));
  INV_X1 U1_g83(.A (G0), .ZN (U1_n_0));
  DFF_X1 U2_DFF_0_I1_Q_reg(.CK (CK), .D (U2_n_6), .Q (n_15), .QN ());
  AND2_X1 U2_g76(.A1 (n_8), .A2 (G0), .ZN (U2_n_6));
  DFF_X1 U2_DFF_1_I1_Q_reg(.CK (CK), .D (U2_n_4), .Q (n_21), .QN ());
  INV_X1 U2_g77(.A (U2_n_4), .ZN (n_8));
  NOR2_X1 U2_g78(.A1 (U2_n_3), .A2 (n_12), .ZN (U2_n_4));
  DFF_X1 U2_DFF_2_I1_Q_reg(.CK (CK), .D (U2_n_2), .Q (n_27), .QN ());
  AOI22_X1 U2_g80(.A1 (U2_n_1), .A2 (G3), .B1 (n_18), .B2 (U2_n_0), .ZN
       (U2_n_3));
  NOR2_X1 U2_g81(.A1 (U2_n_1), .A2 (G2), .ZN (U2_n_2));
  NOR2_X1 U2_g82(.A1 (n_24), .A2 (G1), .ZN (U2_n_1));
  INV_X1 U2_g83(.A (G0), .ZN (U2_n_0));
endmodule

module myvoter(in0, in1, in2, out);
  input in0, in1, in2;
  output out;
  wire in0, in1, in2;
  wire out;
  wire n_0, n_1;
  NAND2_X1 g118(.A1 (n_1), .A2 (n_0), .ZN (out));
  OAI21_X1 g119(.A (in2), .B1 (in0), .B2 (in1), .ZN (n_1));
  NAND2_X1 g120(.A1 (in0), .A2 (in1), .ZN (n_0));
endmodule
```

benchmark/s27/s27_fgdtmr.v

Figure A.4: FGDTMR s27 circuit



Source: From the author

Generated list of three redundant flip-flops and original counterpart for CGTMR and FGDTMR circuits

```
array set s27_cgtmr_fgdtmr_ff {
0 {DFF_0_I1_Q_reg/U$1 U0_DFF_0_I1_Q_reg/U$1 U1_DFF_0_I1_Q_reg/U$1 U2_DFF_0_I1_Q_reg/U$1}
1 {DFF_1_I1_Q_reg/U$1 U0_DFF_1_I1_Q_reg/U$1 U1_DFF_1_I1_Q_reg/U$1 U2_DFF_1_I1_Q_reg/U$1}
2 {DFF_2_I1_Q_reg/U$1 U0_DFF_2_I1_Q_reg/U$1 U1_DFF_2_I1_Q_reg/U$1 U2_DFF_2_I1_Q_reg/U$1}
}
```

benchmark/s27/s27_build_tri_ff_cgtmr_fgdtmr.txt

Generated list of three redundant flip-flops and original counterpart for FGLTMR circuit

```
array set s27_fgltmr_ff {
0 {DFF_0_I1_Q_reg/U$1 U0_DFF_0_I1_Q_reg_tmr/ff_0/U$1 U0_DFF_0_I1_Q_reg_tmr/ff_1/U$1
     U0_DFF_0_I1_Q_reg_tmr/ff_2/U$1}
1 {DFF_1_I1_Q_reg/U$1 U0_DFF_1_I1_Q_reg_tmr/ff_0/U$1 U0_DFF_1_I1_Q_reg_tmr/ff_1/U$1
     U0_DFF_1_I1_Q_reg_tmr/ff_2/U$1}
2 {DFF_2_I1_Q_reg/U$1 U0_DFF_2_I1_Q_reg_tmr/ff_0/U$1 U0_DFF_2_I1_Q_reg_tmr/ff_1/U$1
     U0_DFF_2_I1_Q_reg_tmr/ff_2/U$1}
}
```

benchmark/s27/s27_build_tri_ff_fgltmr.txt

List of input ports of TMR circuit

```
set s27_list_ports_in_top {\
CK G0 G1 G2 G3
}
```

benchmark/s27/s27_ports_input.txt

List of output ports of TMR circuit

```
set s27_list_ports_out_top {\
G17
}
```

benchmark/s27/s27_ports_output.txt

Reports of structural verification step of CGTMR circuit

```
Verification of DFF_0_I1_Q_reg replicas:
  Found flip-flops: 3 out of 3 replicas

Verification of DFF_1_I1_Q_reg replicas:
  Found flip-flops: 3 out of 3 replicas

Verification of DFF_2_I1_Q_reg replicas:
  Found flip-flops: 3 out of 3 replicas

Comparison among logic cones for DFF_0_I1_Q_reg replicas:
  6 combinational gates in logic cone of U0_DFF_0_I1_Q_reg
  6 combinational gates in logic cone of U1_DFF_0_I1_Q_reg
  6 combinational gates in logic cone of U2_DFF_0_I1_Q_reg
  No common gates between logic cones of U0_DFF_0_I1_Q_reg and U1_DFF_0_I1_Q_reg
  No common gates between logic cones of U1_DFF_0_I1_Q_reg and U2_DFF_0_I1_Q_reg
  No common gates between logic cones of U2_DFF_0_I1_Q_reg and U0_DFF_0_I1_Q_reg

Comparison among logic cones for DFF_1_I1_Q_reg replicas:
  4 combinational gates in logic cone of U0_DFF_1_I1_Q_reg
  4 combinational gates in logic cone of U1_DFF_1_I1_Q_reg
  4 combinational gates in logic cone of U2_DFF_1_I1_Q_reg
  No common gates between logic cones of U0_DFF_1_I1_Q_reg and U1_DFF_1_I1_Q_reg
  No common gates between logic cones of U1_DFF_1_I1_Q_reg and U2_DFF_1_I1_Q_reg
  No common gates between logic cones of U2_DFF_1_I1_Q_reg and U0_DFF_1_I1_Q_reg

Comparison among logic cones for DFF_2_I1_Q_reg replicas:
  2 combinational gates in logic cone of U0_DFF_2_I1_Q_reg
  2 combinational gates in logic cone of U1_DFF_2_I1_Q_reg
  2 combinational gates in logic cone of U2_DFF_2_I1_Q_reg
  No common gates between logic cones of U0_DFF_2_I1_Q_reg and U1_DFF_2_I1_Q_reg
  No common gates between logic cones of U1_DFF_2_I1_Q_reg and U2_DFF_2_I1_Q_reg
  No common gates between logic cones of U2_DFF_2_I1_Q_reg and U0_DFF_2_I1_Q_reg
```

benchmark/s27/s27_cgtmr_str_verify.rpt

```
Output: /designs/s27_cgtmr/ports_out/G17
Found voter: /designs/s27_cgtmr/instances_hier/g1

Comparison among logic cones for /designs/s27_cgtmr/instances_hier/g1 pins:
  5 combinational gates in logic cone of pin in0
  5 combinational gates in logic cone of pin in1
  5 combinational gates in logic cone of pin in2
  No common gates between logic cones of pins in0 and in1
  No common gates between logic cones of pins in1 and in2
  No common gates between logic cones of pins in2 and in0
```

benchmark/s27/s27_cgtmr_str_verify_po.rpt

Reports of structural verification step of FGLTMR circuit

```
Verification of DFF_0_I1_Q_reg replicas:
  Found flip-flops: 3 out of 3 replicas

Verification of DFF_1_I1_Q_reg replicas:
  Found flip-flops: 3 out of 3 replicas

Verification of DFF_2_I1_Q_reg replicas:
  Found flip-flops: 3 out of 3 replicas

Comparison among logic cones for DFF_0_I1_Q_reg replicas:
  15 combinational gates in logic cone of U0_DFF_0_I1_Q_reg_tmr
  15 combinational gates in logic cone of U0_DFF_0_I1_Q_reg_tmr
  15 combinational gates in logic cone of U0_DFF_0_I1_Q_reg_tmr
  15 common gates between logic cones of U0_DFF_0_I1_Q_reg_tmr and U0_DFF_0_I1_Q_reg_tmr
  15 common gates between logic cones of U0_DFF_0_I1_Q_reg_tmr and U0_DFF_0_I1_Q_reg_tmr
  15 common gates between logic cones of U0_DFF_0_I1_Q_reg_tmr and U0_DFF_0_I1_Q_reg_tmr

Comparison among logic cones for DFF_1_I1_Q_reg replicas:
  13 combinational gates in logic cone of U0_DFF_1_I1_Q_reg_tmr
  13 combinational gates in logic cone of U0_DFF_1_I1_Q_reg_tmr
  13 combinational gates in logic cone of U0_DFF_1_I1_Q_reg_tmr
  13 common gates between logic cones of U0_DFF_1_I1_Q_reg_tmr and U0_DFF_1_I1_Q_reg_tmr
  13 common gates between logic cones of U0_DFF_1_I1_Q_reg_tmr and U0_DFF_1_I1_Q_reg_tmr
  13 common gates between logic cones of U0_DFF_1_I1_Q_reg_tmr and U0_DFF_1_I1_Q_reg_tmr

Comparison among logic cones for DFF_2_I1_Q_reg replicas:
  5 combinational gates in logic cone of U0_DFF_2_I1_Q_reg_tmr
  5 combinational gates in logic cone of U0_DFF_2_I1_Q_reg_tmr
  5 combinational gates in logic cone of U0_DFF_2_I1_Q_reg_tmr
  5 common gates between logic cones of U0_DFF_2_I1_Q_reg_tmr and U0_DFF_2_I1_Q_reg_tmr
  5 common gates between logic cones of U0_DFF_2_I1_Q_reg_tmr and U0_DFF_2_I1_Q_reg_tmr
  5 common gates between logic cones of U0_DFF_2_I1_Q_reg_tmr and U0_DFF_2_I1_Q_reg_tmr
```

benchmark/s27/s27_fgltmr_str_verify.rpt

```
Output: /designs/s27_fgltmr/ports_out/G17
Not found voter instace - Found libcell
```

benchmark/s27/s27_fgltmr_str_verify_po.rpt

Reports of structural verification step of FGDTMR circuit

```
Verification of DFF_0_I1_Q_reg replicas:
  Found flip-flops: 3 out of 3 replicas

Verification of DFF_1_I1_Q_reg replicas:
  Found flip-flops: 3 out of 3 replicas

Verification of DFF_2_I1_Q_reg replicas:
  Found flip-flops: 3 out of 3 replicas

Comparison among logic cones for DFF_0_I1_Q_reg replicas:
  15 combinational gates in logic cone of U0_DFF_0_I1_Q_reg
  15 combinational gates in logic cone of U1_DFF_0_I1_Q_reg
  15 combinational gates in logic cone of U2_DFF_0_I1_Q_reg
  No common gates between logic cones of U0_DFF_0_I1_Q_reg and U1_DFF_0_I1_Q_reg
  No common gates between logic cones of U1_DFF_0_I1_Q_reg and U2_DFF_0_I1_Q_reg
  No common gates between logic cones of U2_DFF_0_I1_Q_reg and U0_DFF_0_I1_Q_reg

Comparison among logic cones for DFF_1_I1_Q_reg replicas:
  13 combinational gates in logic cone of U0_DFF_1_I1_Q_reg
  13 combinational gates in logic cone of U1_DFF_1_I1_Q_reg
  13 combinational gates in logic cone of U2_DFF_1_I1_Q_reg
  No common gates between logic cones of U0_DFF_1_I1_Q_reg and U1_DFF_1_I1_Q_reg
  No common gates between logic cones of U1_DFF_1_I1_Q_reg and U2_DFF_1_I1_Q_reg
  No common gates between logic cones of U2_DFF_1_I1_Q_reg and U0_DFF_1_I1_Q_reg

Comparison among logic cones for DFF_2_I1_Q_reg replicas:
  5 combinational gates in logic cone of U0_DFF_2_I1_Q_reg
  5 combinational gates in logic cone of U1_DFF_2_I1_Q_reg
  5 combinational gates in logic cone of U2_DFF_2_I1_Q_reg
  No common gates between logic cones of U0_DFF_2_I1_Q_reg and U1_DFF_2_I1_Q_reg
  No common gates between logic cones of U1_DFF_2_I1_Q_reg and U2_DFF_2_I1_Q_reg
  No common gates between logic cones of U2_DFF_2_I1_Q_reg and U0_DFF_2_I1_Q_reg
```

benchmark/s27/s27_fgdtmr_str_verify.rpt

```
Output: /designs/s27_fgdtmr/ports_out/G17
Found voter: /designs/s27_fgdtmr/instances_hier/g1

Comparison among logic cones for /designs/s27_fgdtmr/instances_hier/g1 pins:
  14 combinational gates in logic cone of pin in0
  14 combinational gates in logic cone of pin in1
  14 combinational gates in logic cone of pin in2
  No common gates between logic cones of pins in0 and in1
  No common gates between logic cones of pins in1 and in2
  No common gates between logic cones of pins in2 and in0

```

benchmark/s27/s27_fgdtmr_str_verify_po.rpt

Reports of logical and masking capability verification of CGTMR circuit

```
Compared points are: Equivalent
  (R) + 7    DFF   /U0_DFF_0_I1_Q_reg/U$1
  (R) + 10   DFF   /U1_DFF_0_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 8    DFF   /U0_DFF_1_I1_Q_reg/U$1
  (R) + 11   DFF   /U1_DFF_1_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 9    DFF   /U0_DFF_2_I1_Q_reg/U$1
  (R) + 12   DFF   /U1_DFF_2_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 10   DFF   /U1_DFF_0_I1_Q_reg/U$1
  (R) + 13   DFF   /U2_DFF_0_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 11   DFF   /U1_DFF_1_I1_Q_reg/U$1
  (R) + 14   DFF   /U2_DFF_1_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 12   DFF   /U1_DFF_2_I1_Q_reg/U$1
  (R) + 15   DFF   /U2_DFF_2_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 13   DFF   /U2_DFF_0_I1_Q_reg/U$1
  (R) + 7    DFF   /U0_DFF_0_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 14   DFF   /U2_DFF_1_I1_Q_reg/U$1
  (R) + 8    DFF   /U0_DFF_1_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 15   DFF   /U2_DFF_2_I1_Q_reg/U$1
  (R) + 9    DFF   /U0_DFF_2_I1_Q_reg/U$1
```

benchmark/s27/s27_cgtmr_prove_equi_rev.rpt

```
################################################################################
# Fault Injection
################################################################################
Faulty sequential gates:
   U2_DFF_0_I1_Q_reg/U$1
   U2_DFF_1_I1_Q_reg/U$1
   U2_DFF_2_I1_Q_reg/U$1
Mapping and compare statistics
================================================================================
                     Compare Result     Golden           Revised
--------------------------------------------------------------------------------
Root module name                        s27              s27_cgtmr

Primary inputs                               5                5
   Mapped                                    5                5

Primary outputs                              1                1
   Mapped                                    1                1
      Equivalent                  1

State key points                             3                9
   Mapped                                    3                3
      Equivalent                  3
   Unmapped                                  0                3
      Extra                                  0                3
   Merged                                    0                3
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF       Total
--------------------------------------------------------------------------------
Equivalent           3         3
================================================================================
Compare Results:                                                        PASS


################################################################################
# Fault Injection
################################################################################
Faulty sequential gates:
   U0_DFF_0_I1_Q_reg/U$1
   U0_DFF_1_I1_Q_reg/U$1
   U0_DFF_2_I1_Q_reg/U$1
Mapping and compare statistics
================================================================================
                     Compare Result     Golden           Revised
--------------------------------------------------------------------------------
Root module name                        s27              s27_cgtmr

Primary inputs                               5                5
   Mapped                                    5                5
```

```

Primary outputs                                1                  1
   Mapped                                      1                  1
      Equivalent                  1

State key points                               3                  9
   Mapped                                      3                  3
      Equivalent                  3
   Unmapped                                    0                  3
      Extra                                    0                  3
   Merged                                      0                  3
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF       Total
--------------------------------------------------------------------------------
Equivalent           3         3
================================================================================
Compare Results:                                                        PASS


################################################################################
# Fault Injection
################################################################################
Faulty sequential gates:
   U1_DFF_0_I1_Q_reg/U$1
   U1_DFF_1_I1_Q_reg/U$1
   U1_DFF_2_I1_Q_reg/U$1
Mapping and compare statistics
================================================================================
                     Compare Result      Golden            Revised
--------------------------------------------------------------------------------
Root module name                         s27               s27_cgtmr

Primary inputs                                 5                  5
   Mapped                                      5                  5

Primary outputs                                1                  1
   Mapped                                      1                  1
      Equivalent                  1

State key points                               3                  9
   Mapped                                      3                  3
      Equivalent                  3
   Unmapped                                    0                  3
      Extra                                    0                  3
   Merged                                      0                  3
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF       Total
--------------------------------------------------------------------------------
```

```
Equivalent              3          3
==========================================================================================
Compare Results:                                                                 PASS
```

benchmark/s27/s27_cgtmr_comp_equi.rpt

Reports of logical and masking capability verification of FGLTMR circuit

```
Compared points are: Equivalent
  (R) + 7   DFF  /U0_DFF_0_I1_Q_reg_tmr/ff_0/U$1
  (R) + 8   DFF  /U0_DFF_0_I1_Q_reg_tmr/ff_1/U$1
Compared points are: Equivalent
  (R) + 8   DFF  /U0_DFF_0_I1_Q_reg_tmr/ff_1/U$1
  (R) + 9   DFF  /U0_DFF_0_I1_Q_reg_tmr/ff_2/U$1
Compared points are: Equivalent
  (R) + 9   DFF  /U0_DFF_0_I1_Q_reg_tmr/ff_2/U$1
  (R) + 7   DFF  /U0_DFF_0_I1_Q_reg_tmr/ff_0/U$1
Compared points are: Equivalent
  (R) + 10  DFF  /U0_DFF_1_I1_Q_reg_tmr/ff_0/U$1
  (R) + 11  DFF  /U0_DFF_1_I1_Q_reg_tmr/ff_1/U$1
Compared points are: Equivalent
  (R) + 11  DFF  /U0_DFF_1_I1_Q_reg_tmr/ff_1/U$1
  (R) + 12  DFF  /U0_DFF_1_I1_Q_reg_tmr/ff_2/U$1
Compared points are: Equivalent
  (R) + 12  DFF  /U0_DFF_1_I1_Q_reg_tmr/ff_2/U$1
  (R) + 10  DFF  /U0_DFF_1_I1_Q_reg_tmr/ff_0/U$1
Compared points are: Equivalent
  (R) + 13  DFF  /U0_DFF_2_I1_Q_reg_tmr/ff_0/U$1
  (R) + 14  DFF  /U0_DFF_2_I1_Q_reg_tmr/ff_1/U$1
Compared points are: Equivalent
  (R) + 14  DFF  /U0_DFF_2_I1_Q_reg_tmr/ff_1/U$1
  (R) + 15  DFF  /U0_DFF_2_I1_Q_reg_tmr/ff_2/U$1
Compared points are: Equivalent
  (R) + 15  DFF  /U0_DFF_2_I1_Q_reg_tmr/ff_2/U$1
  (R) + 13  DFF  /U0_DFF_2_I1_Q_reg_tmr/ff_0/U$1
```

benchmark/s27/s27_fgltmr_prove_equi_rev.rpt

```
##########################################################################
# Fault Injection
##########################################################################
Faulty sequential gates:
   U0_DFF_0_I1_Q_reg_tmr/ff_2/U$1
   U0_DFF_1_I1_Q_reg_tmr/ff_2/U$1
   U0_DFF_2_I1_Q_reg_tmr/ff_2/U$1
Mapping and compare statistics
==========================================================================
                    Compare Result      Golden           Revised
--------------------------------------------------------------------------
Root module name                        s27              s27_fgltmr

Primary inputs                              5                 5
   Mapped                                   5                 5

Primary outputs                             1                 1
   Mapped                                   1                 1
      Equivalent                1

State key points                            3                 9
   Mapped                                   3                 3
      Equivalent                3
   Unmapped                                 0                 3
      Extra                                 0                 3
   Merged                                   0                 3
==========================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
==========================================================================
Compared points     DFF       Total
--------------------------------------------------------------------------
Equivalent          3         3
==========================================================================
Compare Results:                                                  PASS


##########################################################################
# Fault Injection
##########################################################################
Faulty sequential gates:
   U0_DFF_0_I1_Q_reg_tmr/ff_0/U$1
   U0_DFF_1_I1_Q_reg_tmr/ff_0/U$1
   U0_DFF_2_I1_Q_reg_tmr/ff_0/U$1
Mapping and compare statistics
==========================================================================
                    Compare Result      Golden           Revised
--------------------------------------------------------------------------
Root module name                        s27              s27_fgltmr

Primary inputs                              5                 5
   Mapped                                   5                 5
```

```

Primary outputs                               1                   1
   Mapped                                     1                   1
      Equivalent                 1

State key points                              3                   9
   Mapped                                     3                   3
      Equivalent                 3
   Unmapped                                   0                   3
      Extra                                   0                   3
   Merged                                     0                   3
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF       Total
--------------------------------------------------------------------------------
Equivalent           3         3
================================================================================
Compare Results:                                                        PASS


################################################################################
# Fault Injection
################################################################################
Faulty sequential gates:
   U0_DFF_0_I1_Q_reg_tmr/ff_1/U$1
   U0_DFF_1_I1_Q_reg_tmr/ff_1/U$1
   U0_DFF_2_I1_Q_reg_tmr/ff_1/U$1
Mapping and compare statistics
================================================================================
                     Compare Result      Golden            Revised
--------------------------------------------------------------------------------
Root module name                         s27               s27_fgltmr

Primary inputs                                5                   5
   Mapped                                     5                   5

Primary outputs                               1                   1
   Mapped                                     1                   1
      Equivalent                 1

State key points                              3                   9
   Mapped                                     3                   3
      Equivalent                 3
   Unmapped                                   0                   3
      Extra                                   0                   3
   Merged                                     0                   3
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF       Total
--------------------------------------------------------------------------------
```

```
Equivalent               3          3
================================================================================
Compare Results:                                                       PASS
```

benchmark/s27/s27_fgltmr_comp_equi.rpt

Reports of logical and masking capability verification of FGDTMR circuit

```
Compared points are: Equivalent
  (R) + 7    DFF   /U0_DFF_0_I1_Q_reg/U$1
  (R) + 10   DFF   /U1_DFF_0_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 10   DFF   /U1_DFF_0_I1_Q_reg/U$1
  (R) + 13   DFF   /U2_DFF_0_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 13   DFF   /U2_DFF_0_I1_Q_reg/U$1
  (R) + 7    DFF   /U0_DFF_0_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 8    DFF   /U0_DFF_1_I1_Q_reg/U$1
  (R) + 11   DFF   /U1_DFF_1_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 11   DFF   /U1_DFF_1_I1_Q_reg/U$1
  (R) + 14   DFF   /U2_DFF_1_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 14   DFF   /U2_DFF_1_I1_Q_reg/U$1
  (R) + 8    DFF   /U0_DFF_1_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 9    DFF   /U0_DFF_2_I1_Q_reg/U$1
  (R) + 12   DFF   /U1_DFF_2_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 12   DFF   /U1_DFF_2_I1_Q_reg/U$1
  (R) + 15   DFF   /U2_DFF_2_I1_Q_reg/U$1
Compared points are: Equivalent
  (R) + 15   DFF   /U2_DFF_2_I1_Q_reg/U$1
  (R) + 9    DFF   /U0_DFF_2_I1_Q_reg/U$1
```

benchmark/s27/s27_fgdtmr_prove_equi_rev.rpt

```
################################################################################
# Fault Injection
################################################################################
Faulty sequential gates:
   U2_DFF_0_I1_Q_reg/U$1
   U2_DFF_1_I1_Q_reg/U$1
   U2_DFF_2_I1_Q_reg/U$1
Mapping and compare statistics
================================================================================
                    Compare Result      Golden              Revised
--------------------------------------------------------------------------------
Root module name                        s27                 s27_fgdtmr

Primary inputs                              5                   5
   Mapped                                   5                   5

Primary outputs                             1                   1
   Mapped                                   1                   1
      Equivalent                1

State key points                            3                   9
   Mapped                                   3                   3
      Equivalent                3
   Unmapped                                 0                   3
      Extra                                 0                   3
   Merged                                   0                   3
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points     DFF       Total
--------------------------------------------------------------------------------
Equivalent          3         3
================================================================================
Compare Results:                                                     PASS


################################################################################
# Fault Injection
################################################################################
Faulty sequential gates:
   U0_DFF_0_I1_Q_reg/U$1
   U0_DFF_1_I1_Q_reg/U$1
   U0_DFF_2_I1_Q_reg/U$1
Mapping and compare statistics
================================================================================
                    Compare Result      Golden              Revised
--------------------------------------------------------------------------------
Root module name                        s27                 s27_fgdtmr

Primary inputs                              5                   5
   Mapped                                   5                   5
```

```

Primary outputs                                         1                     1
   Mapped                                               1                     1
      Equivalent                          1

State key points                                        3                     9
   Mapped                                               3                     3
      Equivalent                          3
   Unmapped                                             0                     3
      Extra                                             0                     3
   Merged                                               0                     3
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF        Total
--------------------------------------------------------------------------------
Equivalent           3          3
================================================================================
Compare Results:                                                          PASS


################################################################################
# Fault Injection
################################################################################
Faulty sequential gates:
   U1_DFF_0_I1_Q_reg/U$1
   U1_DFF_1_I1_Q_reg/U$1
   U1_DFF_2_I1_Q_reg/U$1
Mapping and compare statistics
================================================================================
                        Compare Result        Golden                Revised
--------------------------------------------------------------------------------
Root module name                              s27                   s27_fgdtmr

Primary inputs                                          5                     5
   Mapped                                               5                     5

Primary outputs                                         1                     1
   Mapped                                               1                     1
      Equivalent                          1

State key points                                        3                     9
   Mapped                                               3                     3
      Equivalent                          3
   Unmapped                                             0                     3
      Extra                                             0                     3
   Merged                                               0                     3
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF        Total
--------------------------------------------------------------------------------
```

```
 Equivalent             3              3
================================================================================
Compare  Results:                                                         PASS
```

benchmark/s27/s27_fgdtmr_comp_equi.rpt