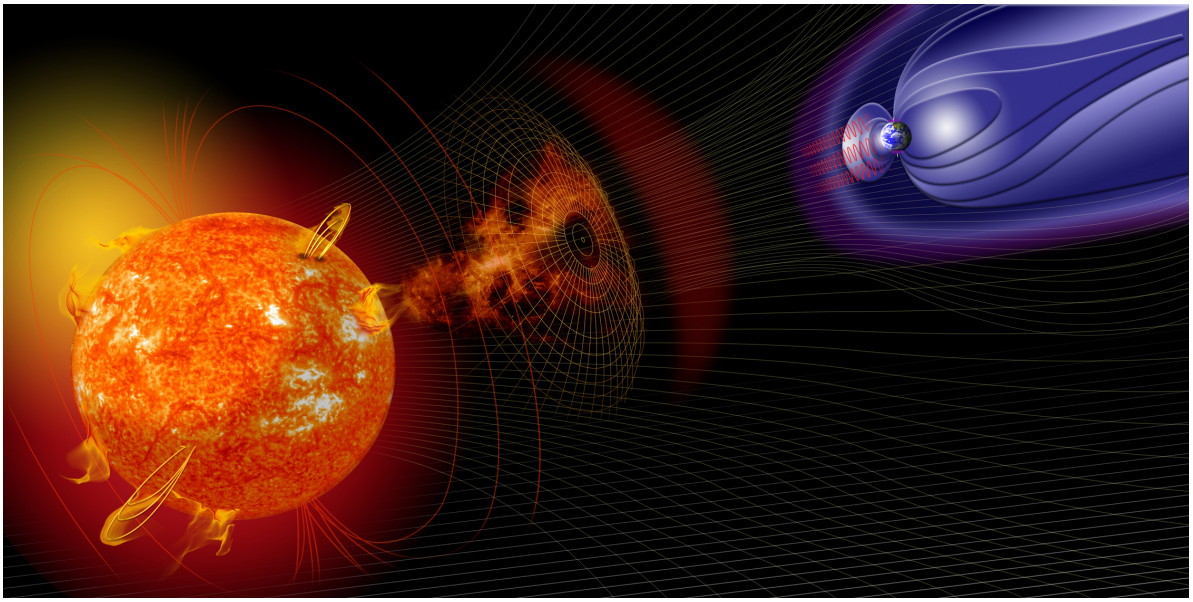


Fault Vulnerability and Countermeasures in Digital Systems for Airborne and Space Applications

Master's thesis in Embedded Electronic System Design

Andrée Centervall
Marko Russegren Sladic

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2016

Master's thesis 2016

Fault Vulnerability and Countermeasures in Digital Systems for Airborne and Space Applications

Andrée Centervall
Marko Russegren Sladic

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2016

Fault Vulnerability and Countermeasures in Digital Systems for Airborne and Space Applications
Marko Russegren Sladic, Andrée Centervall

Supervisor Chalmers: Lena Petersson, Department of Computer Science and Engineering
Supervisor Saab: Johan Friberg, Saab Surveillance
Examiner: Per Larsson-Edefors, Department of Computer Science and Engineering

Master's Thesis 2016
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Artist illustration of events on the sun changing the conditions in Near-Earth space [1].

Typeset in LaTeX
Gothenburg, Sweden 2016

Fault Vulnerability and Countermeasures in Digital Systems for Airborne and Space Applications
Andrée Centervall, Marko Russegren Sladic
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg

Abstract

This thesis project explores and tests different combinations of fault tolerant techniques in order to counteract radiation-induced faults in static random access memory (SRAM)-based field-programmable gate array (FPGA). The use of FPGAs has become very popular for critical airborne systems and space applications during the last decade due to their outstanding performance, high flexibility, low non-recurring engineering (NRE) cost and fast time to market (TTM) compared to other customized approaches.

Radiation-induced faults, such as single event upsets (SEUs) and single event transients (SETs), do not permanently damage transistors or circuits but can still cause severe system failures. These faults can be experienced at both sea level and higher altitudes. However, these type of faults are more common at higher altitudes due to higher radiation levels. This poses a substantial threat to mission-critical FPGA-based applications used in airborne systems.

Many researches conducted over the years have presented mitigation techniques such as triple modular redundancy (TMR) to be successful for reducing the chances of failures in SRAM-based FPGAs due to radiation-induced faults. However TMR comes with an increased area cost since three identical modules are actively running the same design. The purpose of this thesis project is to individually test other fault mitigation techniques as well as combinations of different techniques in order to achieve high reliability while minimizing the overhead area and overall system cost.

The hardware adopted for this thesis project consists of a Kintex-7 FPGA board from Xilinx. The simulated radiation-induced faults are injected into relevant parts of the hardware in order to test the reliability of the proposed fault mitigation techniques.

The simulations show great results and the obtained reliability during tests follow the expected theoretically calculated values. The results also show the impact that the techniques has on design area and overall power consumption. Furthermore, the trade-offs between the different mitigation techniques studied in this thesis are presented.

Keywords: Fault Tolerance, SRAM-based FPGA, Fault Mitigation Techniques, Single Event Effect (SEE), Single Event Upset (SEU), Single Event Transient (SET), Triple modular redundancy (TMR), Error Correcting Code (ECC), Time-redundancy (TR), Active Partial Reconfiguration (APR)

Acknowledgements

We would like to express our gratitude to all whose support made this thesis work possible. Primarily, we would like to thank our supervisor Johan Friberg and line manager Jenny Klinton from Saab Surveillance for their guidance and support during this work.

We would also like to thank our supervisor Lena Petersson and our examiner Per Larsson-Edefors from Chalmers University of Technology as well as Deputy Director Nils Dagås from Saab Surveillance for making this thesis work possible.

Andrée Centervall & Marko Russegren Sladic, Gothenburg, 2016

Contents

List of Figures **xi**

List of Tables **xiii**

Acronyms **xv**

1 Introduction **1**
1.1 Background . . . 1
1.2 SAAB . . . 2
1.3 Single Event Effects . . . 2
1.4 Purpose and Objective . . . 2
1.5 Similar studies . . . 3
1.6 Scope and Limitations . . . 3

2 Background Theory **5**
2.1 Radiation . . . 5
2.1.1 Radiation at different altitudes . . . 6
2.2 Field-Programmable Gate Array Architecture . . . 6
2.2.1 SRAM-based FPGA . . . 7
2.2.2 Flash-based FPGA . . . 8
2.2.3 Antifuse-based FPGA . . . 8
2.3 Single Event Effects . . . 9
2.3.1 Single Event Upset . . . 9
2.3.2 Multi-Cell and Multi-Bit Upset . . . 9
2.3.3 Single Event Transient . . . 10
2.3.4 Single Event Burnout . . . 10
2.3.5 Single Event Latchup . . . 10

3 Evaluation Context **11**
3.1 FPGA . . . 11
3.2 Selection of mitigation techniques . . . 12
3.2.1 General techniques . . . 12
3.3 Multi-word and Multi-bit Upsets . . . 13

4 Mitigation Techniques for FPGAs **15**
4.1 Triple modular redundancy . . . 17
4.2 Error-correcting code . . . 21
4.2.1 Hamming code . . . 21

4.3 Active partial reconfiguration . . . 24
4.4 Scrubbing . . . 25
4.5 Time-redundancy . . . 25

5 Evaluation Method **27**
5.1 Error injection . . . 27
5.1.1 BRAM . . . 27
5.1.2 Distributed memory . . . 27
5.1.3 Configuration memory . . . 28
5.1.4 Single Event Transient . . . 28
5.2 Test designs . . . 28
5.2.1 Test design 1 . . . 28
5.2.2 Test design 2 . . . 29
5.3 Testing . . . 29
5.3.1 BRAM, distributed memory and configuration memory . . . 29
5.3.2 Single event transient pulses . . . 30

6 Results **31**
6.1 Reference Design . . . 31
6.2 Triple modular redundancy . . . 35
6.3 Error-Correcting Code . . . 39
6.4 Active partial reconfiguration . . . 43
6.5 TMR combined with APR . . . 45
6.6 Comparison . . . 47
6.6.1 Performance, utilization and power . . . 48
6.6.2 Reliability . . . 48
6.6.3 Overall system cost . . . 49
6.7 Single Event Transient . . . 50

7 Discussion **51**
7.1 Limitations of test designs and error injection methods . . . 51
7.2 Evaluation of test results . . . 53
7.2.1 Performance, utilization and power . . . 53
7.2.2 Reliability . . . 54
7.3 Future Work . . . 55

8 Conclusion **57**

Bibliography **59**

List of Figures

2.1 Penetrating power of alpha, beta and gamma radiation. 5
2.2 Relative neutron flux depending on altitude. [12] 7
2.3 Example of a CLB cell 8
2.4 SRAM memory cell 8

4.1 Refence circuit with an AND gate and a register where a wire is getting hit by a particle causing an SET 15
4.2 Timing-table for circuit in Fig. 4.1 15
4.3 Reference circuit with an AND gate and a register where the LUT is getting hit by a particle causing an SEU 16
4.4 Timing-table for circuit in Fig. 4.3 16
4.5 Circuit with an AND gate and a register using TMR where a wire is getting hit by a particle, causing an SET 17
4.6 Timing-table for circuit in Fig. 4.5 18
4.7 Circuit with an AND gate and a register using TMR where a LUT is getting hit by a particle, causing an SEU 18
4.8 Timing-table for circuit in Fig. 4.7 with particle hitting the LUT for the AND gate 19
4.9 The reliability of reference design and TMR design without voter 20
4.10 Timing-table for circuit in Fig. 4.3 when APR is active 25
4.11 Circuit with an AND gate and three registers implementing time redundency 26

6.1 Reliability results for the block random access memory (BRAM) test of reference design 1 33
6.2 Reliability results for the configuration memory test of reference design 1 34
6.3 Reliability results for the distributed memory test of reference design 2 34
6.4 Reliability results for the configuration memory test of reference design 2 35
6.5 Reliability results for the BRAM memory test of TMR design 1 37
6.6 Reliability results for the configuration memory test of TMR design 1 37
6.7 Reliability results for the distributed memory test of TMR design 2 38
6.8 Reliability results for the configuration memory test of TMR design 2 38

6.9 Reliability results for the BRAM memory test of design 1 with ECC in BRAM 41
6.10 Reliability results for the configuration memory test of design 1 with ECC in BRAM 41
6.11 Reliability results for the distributed memory test of design 2 with ECC in the distributed memory 42
6.12 Reliability results for the configuration memory test of design 2 with ECC in the distributed memory 42
6.13 Reliability results for the configuration memory test when using active partial reconfiguration (APR) in design 1 44
6.14 Reliability results for the configuration memory test when using APR in design 2 45
6.15 Reliability results for the configuration memory when using TMR combined with APR in design 1 47
6.16 Reliability results for the configuration memory when using TMR combined with APR in design 2 47
6.17 Negative effects for different techniques. 48
6.18 Average failure rate comparison among all tested techniques . . 49
6.19 Average failure rate comparison between the reference and TMR 49

List of Tables

I	Product specification for the XC7K325T FPGA	11
II	Possible effects on a LUT when hit by a particle.	16
III	Structure and parity coverage for a 7-bit hamming-code with three parity bits	21
IV	Data coverage for each parity bit	22
V	Timing-table for circuit in Fig. 4.11	26
VI	Maximum clock frequency and total on-chip power of reference test designs 1 and 2	31
VII	Utilization percent of reference test designs 1 and 2	32
VIII	Total number of injections, observed failures and MTTF of reference test designs 1 and 2	33
IX	Maximum clock frequency and total on-chip power of TMR-based test designs 1 and 2	35
X	Utilization percent of TMR-based test designs 1 and 2	36
XI	Total number of injections, observed failures and MTTF of TMR-based test designs 1 and 2	36
XII	Maximum clock frequency and total on-chip power of ECC-based test designs 1 and 2	39
XIII	Utilization percent of ECC-based test designs 1 and 2	39
XIV	Total number of injections, observed failures and MTTF of ECC-based test designs 1 and 2	40
XV	Maximum clock frequency and total on-chip power of APR-based test designs 1 and 2	43
XVI	Utilization percent of APR-based test designs 1 and 2	43
XVII	Total number of injections, observed failures and MTTF of APR-based test designs 1 and 2	44
XVIII	Maximum clock frequency and total on-chip power of TMR-based test designs 1 and 2 with APR	45
XIX	Utilization percent of TMR-based test designs 1 and 2 with APR	46
XX	Total number of injections, observed failures and MTTF of reference test designs 1 and 2	46
XXI	Failures observed with 500 injected SETs pulses	50

Acronyms

APR	active partial reconfiguration
ASIC	application-specific integrated circuit
BRAM	block random access memory
CLB	configurable logic block
CMOS	complementary metal oxide semiconductor
CRC	cyclic redundancy check
DSP	digital signal processing
ECC	error correcting code
FET	field-effect transistor
FF	flip-flop
FIT	failures in time
FPGA	field-programmable gate array
HDL	hardware description language
IC	integrated circuit
ICAP	internal configuration access port
IP	intellectual property
LET	linear energy transfer
LFSR	linear-feedback shift register
LUT	look-up table
LUTRAM	look-Up table RAM
MBU	multi-bit upset
MCU	multi-cell upset
MTTF	mean time to failure
MWU	multi-word upset
NRE	non-recurring engineering

RAM	random access memory
SEB	single event burnout
SEE	single event effect
SEL	single event latchup
SEM	soft error mitigation
SET	single event transient
SEU	single event upset
SRAM	static random access memory
TMR	triple modular redundancy
TR	time redundancy
TTM	time to market
UAV	unmanned aerial vehicle

1
Introduction

This chapter presents an introduction and background as well as information about the company supervising the execution of this thesis project.

1.1 Background

As process geometries continue to shrink, embedded digital systems such as field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) become increasingly vulnerable to state changes or temporal voltage pulses due to radiation such as particle radiation and electromagnetic radiation. These changes, known as single event effects (SEEs), have the potential to cause internal states and registers to switch from a logical zero to logical one or vice versa and ultimately even damage the transistors or circuits. These effects become increasingly problematic for applications used in space and airborne systems due to the higher radiation levels at higher altitudes.

SEEs are random and they can have a serious impact on mission-critical applications as they can result in data corruption, data loss and severe system failures. For software processes operating on susceptible components in environments with high radiation levels, it is important to ensure that FPGA and ASIC applications are designed in a manner to reduce the likelihood of radiation-induced system failures. Faults that do not permanently damage transistors or circuits can still have serious impact on the function of the application.

In low-quantity integrated circuit (IC) designs it is often favorable to use FPGAs before ASICs because of the low non-recurring engineering (NRE) cost and short time to market (TTM). For space applications it is also favorable that the design may be updated after launching the device to space. Therefore, an FPGA is more advantageous for some applications since it can be reprogrammed while an ASIC is customized to the design and cannot be changed. For these reasons, FPGAs have become widely used in applications for airborne and space systems. A major drawback with FPGAs is that because they are reprogrammable, the design configuration is stored in a memory known as the configuration memory which can be very sensitive to SEEs. Consequently, an FPGA has a lower reliability than an ASIC when exposed to ionizing radiation.

1.2 SAAB

For companies like SAAB, creating products used at higher altitudes and in environments where a repair is undesired, SEEs could become a big problem. It is therefore favorable to have a design process where the decreased reliability caused by SEE is analyzed. If the decreased reliability is a problem, a mitigation technique should be implemented with minimal effect on characteristics such as area and power consumption of a product.

SAAB surveillance develops both airborne products as well as products used at sea level. Different products have different reliability demands and for these reasons they wanted to implement a design process that analyzes the effects of SEEs for a wide range of products and environments. The mitigation techniques studied in this report were also chosen with SAAB's products in mind.

1.3 Single Event Effects

When a particle or wave with high energy hits the silicon in an IC the energy will be transferred to the silicon and can cause a fault in the system. These faults can either be of a non-destructive och destructive nature and are called SEEs. A non-destructive fault means that the hardware is not permanently damaged from the event and is either an single event upset (SEU) or an single event transient (SET). Destructive effects will permanently damage the hardware; where single event burnout (SEB) and single event latchup (SEL) are examples of such effects [2].

1.4 Purpose and Objective

The products developed at SAAB Surveillance may be used in critical situations where reliable systems are of great importance. Some products are also critical to human safety and can be used in situations where an unexpected error can have a very dangerous outcome. Evaluating the fault tolerance and sensitivity of different designs as well as testing different mitigation techniques when needed, will help SAAB develop even more reliable systems.

The main purpose of this thesis is to design and implement two different test platforms in a Xilinx FPGA in order to run representative test processes. These test platforms are then used to evaluate different design methods and to counteract system failures caused by SEEs focusing on system reliability, overhead area, performance, power consumption and overall system cost.

The aim is to study both data and configuration upsets as well as transient errors and give recommendations to improve the design process by studying new coding styles, fault-tolerant techniques, verification techniques, etc. These goals will be achieved by individually evaluating different fault mitigation methods as well as combinations of them with the help of different test platforms. Some of the methods studied are error correction encoding, redundancy techniques and advanced mitigation techniques based on the tools provided by the hardware vendor.

Generally, the major goal was to develop, test and study different designs to reduce the risk of failures and improve the overhead area, performance, power efficiency and overall system cost.

1.5 Similar studies

Many researchers and students have studied the effects of SEEs and how to use different mitigation techniques to decrease a design's sensitivity against SEEs [3, 4, 5]. Most of these studies focus on a specific kind of mitigation technique or a specific kind of SEE. This project is about creating a test process that SAAB can use to easily check their designs and if needed suggest mitigation techniques depending on their restrictions in size, performance, power consumption and system cost. In contrast to other previous studies performed, this thesis is about developing test platforms in a generic way in order to evaluate many types of FPGA-based designs as well as to study several kinds of SEEs and mitigation techniques.

1.6 Scope and Limitations

The project studies SEEs only in form of SEUs and SETs. Permanent destructive faults such as SEL and SEB are not considered.

This thesis focuses on testing and evaluating designs implemented on commercial static random access memory (SRAM)-based FPGA devices from Xilinx, more specifically on the Xilinx Kintex-7 board chosen by SAAB. The use of Flash and Antifuse FPGAs is not studied in this work.

Techniques such as triple modular redundancy (TMR), error correcting code (ECC), active partial reconfiguration (APR) and other types of technique combinations are tested for mitigating soft SEEs. There are also other SEE mitigation techniques at different design levels such as SEU hardening techniques for digital CMOS designs. Since these hardening techniques are done at a logic cell design level, these techniques are considered to be outside the project scope.

Reference parameters for fault injection simulations are based on radiation induced faults that occur anywhere between sea level and maximum flight altitude. However, testing and evaluating space applications can be performed in the same manner just by applying corresponding fault injection parameter values.

Multi-cell upsets (MCUs) in form of multi-word upsets (MWUs) are not simulated because they do not have any functional difference for the mitigation techniques chosen. Since MWUs are not simulated, the results obtained are considered to be somewhat optimistic and better than the ones expected in reality.

2
Background Theory

This chapter describes the theoretical background needed to understand this project.

2.1 Radiation

Radiation comes in many forms and is defined as "The process in which energy is emitted as particles or waves" [6]. It can be divided into ionizing or non-ionizing radiation which depends on the amount of energy it carries. Ionizing radiation will carry energy higher than 10 eV and has the possibility of ionizing atoms and molecules as well as breaking chemical bounds. Ionizing radiation occurs either as a particle, for example alpha and beta radiation, or as a wave, electromagnetic radiation [7].

An alpha radiation particle is an atomic nuclei consisting of two protons and two neutrons. These particles collide with other forms of matter very easily due to their big size and relatively low speed. When this collision happens, these particles tend to lose their energy very quickly. Alpha radiation has therefore a very low penetrating power and can be stopped by just a sheet of paper. In the case of a human body the first layers of skin will stop the particle if the source is outside the body [8].

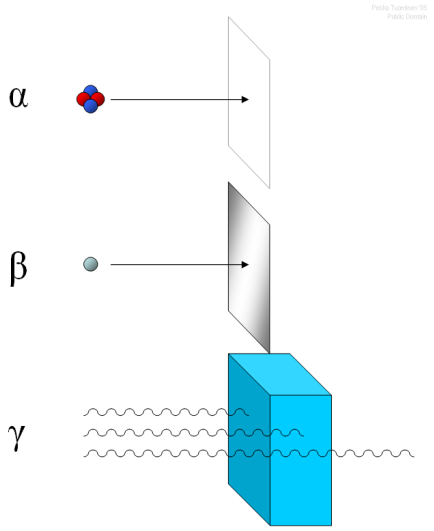


Figure 2.1: Penetrating power of alpha, beta and gamma radiation.

In beta radiation, the particles are electrons ejected from a nuclei moving at a very high speed. Their penetrating power is higher than for alpha particles and can penetrate about 1-2 cm of water. However, the penetration can be stopped by using just a couple of mm thick aluminum [8]. A beta particle can penetrate human skin to the layer where new cells are produced which can result in skin injury.

Electromagnetic radiation is the energy emitted in the form of waves with different wavelengths. Heat and radio waves are examples of electromagnetic radiation with a long wavelength while X-rays and gamma-rays are examples of electromagnetic radiation with a short wavelength [7]. Visible light is also categorized as electromagnetic radiation with wavelengths between 400 nm and 700 nm [9] compared to less than 1 pm for gamma-rays. The difference between x-rays and gamma-rays is that x-rays are produced artificially while gamma-rays occur naturally [8].

Gamma-ray has great penetrating power and compared to alpha and beta, which can be stopped by a sheet of paper and aluminum, gamma-rays can only be stopped by denser materials such as lead or concrete, see Fig. 2.1 [8].

2.1.1 Radiation at different altitudes

Earth's atmosphere and magnetic field protect Earth against cosmic radiation resulting in higher radiation levels at the poles and at higher altitudes. The radiation is not constant, instead the variations can be large during for example solar flares and can even be lethal for spacecraft crew during larger solar flares. Some radiation is deflected by Earth's atmosphere and magnetic field, while some is absorbed by the atmosphere causing secondary radiation [10, 11]. A common way to measure radiation at higher altitudes is by measuring the neutron radiation such as the rate of flow of neutrons known as neutron flux. In Fig. 2.2, it can be seen how neutron flux varies with altitude. The values from the plot represent New York City where the relative radiation level at the surface is 1. It can be seen how the neutron flux levels exponentially increase with altitude.

2.2 Field-Programmable Gate Array Architecture

FPGAs are ICs based on semiconductor devices that can be programmed and reprogrammed to specific application designs by the use of hardware description language (HDL). The most common type of building blocks in FPGAs are configurable logic blocks (CLBs) that typically consist of look-up tables (LUTs), Full Adders and D-type flip-flops as shown in Fig. 2.3. CLBs along with a network of programmable interconnects and I/O Blocks enable the development of complex yet flexible digital circuits.

FPGAs are programmed using configuration bitstreams that hold configuration data. The configuration memory stores configuration data of an FPGA, with SRAM-based configuration memory being the most common technology used for commercial FPGA devices [13]. However, other methods for configuration storage, such as flash- and antifuse-based FPGAs, are also used.

Usually, the configuration memory of Xilinx FPGAs is organized as an array of frames. The entire array of frames is protected by a cyclic redundancy check (CRC)

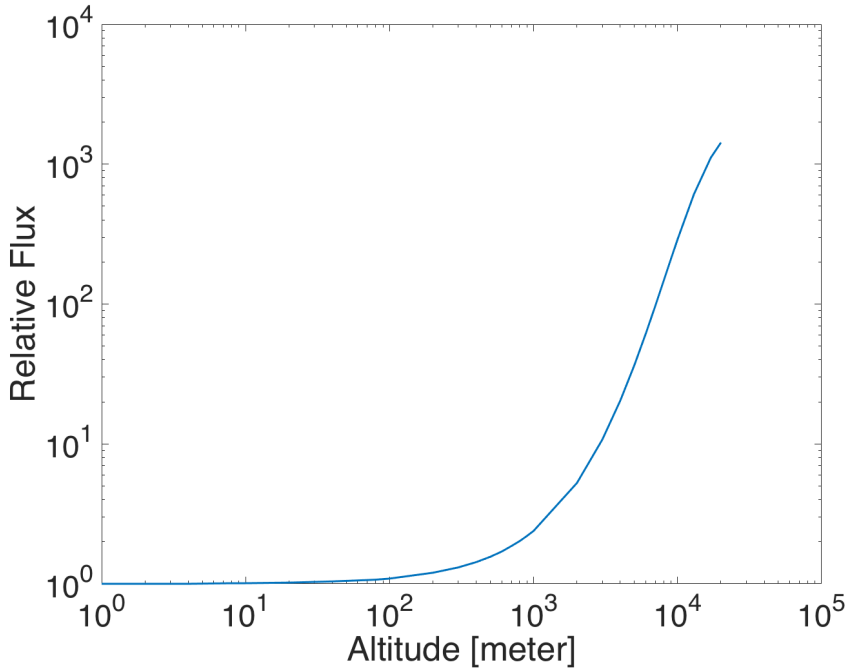


Figure 2.2: Relative neutron flux depending on altitude. [12]

and each frame is protected by ECC. However, the entire array of frames in the configuration memory is not used for a design loaded into a Xilinx FPGA, meaning that only a fraction of the bits in the memory is essential. These bits, known as *essential bits*, are associated with the circuitry of the design. Yet, only a fraction of the essential bits are considered to be *critical bits*, meaning that the design will develop a functional failure if any of these bits are altered [14].

2.2.1 SRAM-based FPGA

The technology that is most commonly used for storing configuration bitstreams in commercial FPGAs is a type of semiconductor memory known as SRAM. The SRAM architecture is organized as an array of memory cells consisting of flip-flop latches built with field-effect transistors (FETs) as shown in Fig. 2.4.

SRAMs are considered to be volatile memories since they require a power source in order to preserve data, which means that once the memory cells are powered off, the configuration bitstream will be lost. Therefore, SRAM-based FPGAs have to be programmed after every start-up. Another downside with this technology is that even though there are some radiation hardened SRAM-based FPGAs available, they are considerably more sensitive to radiation induced faults than other radiation hardened technologies [15]. However, the performance, flexibility and cost of SRAM-based FPGAs are much better than for other technologies. Characteristics such as high performance and high flexibility combined with low overall costs are considered to be of great benefit for airborne and space applications.

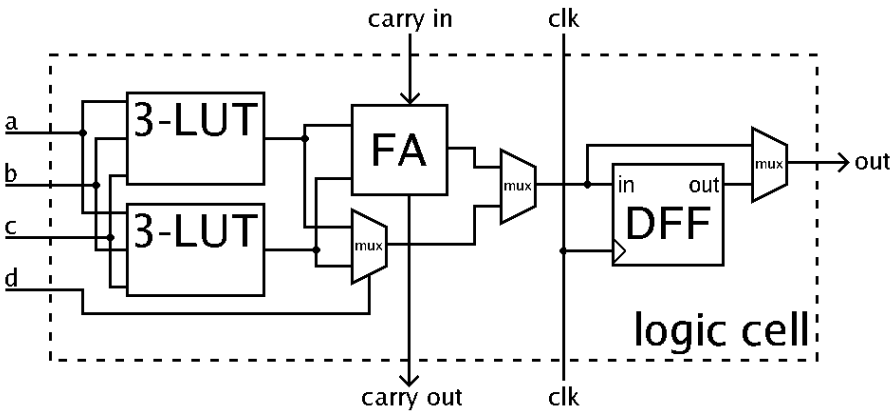


Figure 2.3: Example of a CLB cell

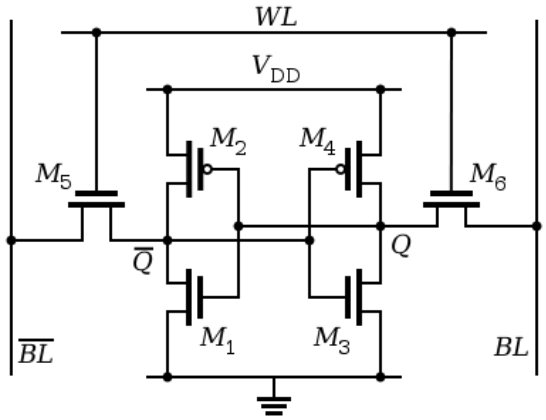


Figure 2.4: SRAM memory cell

2.2.2 Flash-based FPGA

In contrast to SRAM-based FPGAs, flash-based FPGAs are built of an array of non-volatile memory cells, which means that the configuration bitstream is not lost after a power-off. Flash-based FPGAs can be reprogrammed but there is no need to do so at every power-on.

The advantage of flash-based FPGAs for airborne and space applications is that their power consumption is relatively low and the radiation tolerance is higher than that of SRAM-based FPGAs. However, even though the permanent radiation induced faults are reduced in flash-based solutions, transient faults are still a concern [16].

The major drawbacks are that the performance of flash memory is lower than for SRAM-based solutions and that the production cost is much higher.

2.2.3 Antifuse-based FPGA

Antifuse-based FPGAs are different than previously mentioned technologies in the way that they can be programmed only once and cannot be reprogrammed afterwards. This type of technology is beneficial in airborne and space applications due to the low power consumption, high reliability and high design security.

Antifuse-based FPGAs are still vulnerable to faults when running in high radiation environments. However, there are ways to increase the availability of antifuse-based FPGAs for permanent faults. Furthermore, transient faults have also a significant impact on the reliability of antifused-based FPGAs in high radiation environments [17] and new methods for mitigating transient faults have to be studied.

As well as for flash-based solutions, antifuse-based FPGAs have lower performance and higher production costs than do commercial SRAM-based FPGAs.

2.3 Single Event Effects

SEEs can be divided into destructive and non-destructive effects caused by high levels of radiation. Destructive effects are considered to be faults experienced due to permanent damage to transistors or circuits. Non-destructive effects, also known as soft errors, are considered to be faults caused by internal state changes or temporal voltage pulses. Some examples of destructive effects are SELs and SEBs. However, the effects considered in this thesis are non-destructive effects such as SEUs and SETs.

2.3.1 Single Event Upset

If an energized particle or wave hits a flip-flop, memory cell or register cell, it can cause the value to flip from logical zero to a one or vice versa. This error is categorized as an SEU and will cause that value to be permanently switched, which results in an error that can cause a failure [18]. A memory cell can also be part of a LUT, controlling how the logic is routed on the FPGA. If a value is changed inside a LUT, this means that the logic has changed and will not work as designed [19].

2.3.2 Multi-Cell and Multi-Bit Upset

In addition to SEUs, there is a risk of obtaining MCUs and multi-bit upsets (MBUs) due to radiation. MCUs are upsets where a particle affects multiple cells at the same time. The affected cells can either be bits of the same word or bits in different words. The probability of an MCU ranges from approximately 1 % when the Effective linear energy transfer (LET) is $1\,\mathrm{MeV} - \mathrm{cm}^2/\mathrm{mg}$ to approximately 35 % when the Effective LET is $50\,\mathrm{MeV} - \mathrm{cm}^2/\mathrm{mg}$. The majority of these upsets, up to 22.5 % in the Effective LET range mentioned above, are affecting two cells. Upsets affecting three cells or four cells are less probable but each can still happen up to 7.5 % of the time [20].

MBUs are MCUs where the affected cells are part of the same word, meaning that MBUs are a subset of all MCUs. The probability of an MBU occuring is a lite lower than for MCU and goes from approximately 1 % when the Effective LET is $1.2\,\mathrm{MeV} - \mathrm{cm}^2/\mathrm{mg}$ and goes to approximately 28 % for an Effective LET of $50\,\mathrm{MeV} - \mathrm{cm}^2/\mathrm{mg}$ [20].

These MCU and MBU probability estimates vary with FPGA technology. The estimates presented in this section are based on the 28-nm technology used in a Kintex 7 FPGA.

2.3.3 Single Event Transient

An SET is when a energized particle or wave hits a logical path, causing a voltage spike on the location and sometimes the area around it. The voltage spike can cause a logical zero to be interpreted as a logical one and propagate to a flip-flop. If the flip-flop is active when the wrong value reaches it and the wrong value becomes stored in the flip-flop, this fault is then classified as an SEU [18, 2].

Technology scaling is about making transistors smaller and that has made ICs more susceptible to SETs. The decreased capacitance has allowed a particle, or wave, with lower energy to increase the voltage enough for it to cause an error. Lower supply voltage has made the margins, for a logical zero or one, smaller and therefore caused the same problem as the decreased capacitance [21].

2.3.4 Single Event Burnout

If a high-energy neutron hits a reverse-biased power diode or a transistor, the transferred energy can cause electron-hole pairs to be created in an avalanche fashion. This effect can e.g. cause the transistor to be short-circuited and a large current will permanent destroy the transistor [22].

2.3.5 Single Event Latchup

A latchup is when a power-surge causes a low-impedance path between the supply voltage and ground inside the complementary metal oxide semiconductor (CMOS) circuit. When a latchup is caused by a single event it is classified as an SEL. The problem with a latchup is that the low impedance path will remain even after the power-surge has dissipated. An excessive amount of current can go through the path and cause destructive failure of the IC. The only way of clearing a latchup after it has occurred is by removing the supply power from the circuit [23, 24].

3
Evaluation Context

This chapter presents the hardware and mitigation techniques adopted for this project together with a discussion of why these design decisions were made.

3.1 FPGA

Xilinx produces several different FPGAs developed specifically for aerospace and defense applications with built-in functionality for handling SEEs [25]. They also support separate IP cores to add more functionality handling SEEs. These FPGAs are radiation-tested using neutron beam and all results are published on their website [26].

The hardware chosen by SAAB for this project was a Kintex-7 KC705 evaluation board from Xilinx that uses the XC7K325T FPGA. The XC7K325T is a commercially available FPGA that is built on a common 28-nm architecture used in the Kintex-7 family.

The XC7K325T has a low power consumption as well as a medium density for both logic and memory. The specifications can be seen Table I. This FPGA model also has a built-in configuration scrubbing technique with support for Configuration Readback and Self-Repair.

Table I: Product specification for the XC7K325T FPGA

Specification	**XC7K325T**
Slices	50,950
Logic Cells	326,080
CLB Flip-Flops	407,600
Maximum Distributed RAM (Kbits)	4,000
Block RAM (36Kbits each)	445
Total Block RAM (Kbits)	16,020
DSP48E1 Slices	840
Configuration AES / HMAC Blocks	1
Configuration Memory (Mbits)	87,3

The programmable elements in an FPGA are known as CLBs. In the Kintex-7 KC705 evaluation board the Kintex-7 KC705 FPGA, each CLB consists of several

slices and each slice consists of different logic cells. In the Kintex-7 KC705 each slice consists of four LUTs and eight flip-flops. For digital signal processing (DSP) functions, there are also specific DSP slices in the Kintex-7 KC705 FPGA that consists of a pre-adder, a 25 x 18 multiplier, an adder, and an accumulator.

The LUTs in an FPGA are usually used to perform logic functions but they can also be used as storage elements and that is known as the distributed random access memory (RAM) or look-Up table RAM (LUTRAM). On the other hand, the common blocks used as storage elements in an FPGA are the block random access memorys (BRAMs).

3.2 Selection of mitigation techniques

This section will discuss why and how the different mitigation techniques were selected for testing. The selection method is divided between general techniques, which are the techniques selected from the literature review, and the improved techniques, which are general techniques selected together with SAAB for further improvement.

3.2.1 General techniques

The techniques were selected from the literature review as the most used techniques and some of those supplied by the vendor. The different techniques have their advantages and disadvantages, e.g. some techniques will only protect from a certain kind of error while others give a higher protection. The techniques that were chosen are the following:

- Triple modular redundancy
- Error correcting code
  - Hamming code
- Active partial reconfiguration
- Time redundancy
- Combinations of these techniques

A short summary for why each technique was chosen can be seen below.

Triple modular redundancy

This technique was chosen because it can handle faults in the configuration memory, block memory as well as SETs. It is widely used for these reasons and if it is combined with scrubbing it is possible to correct errors without the need for pausing the design. The advantage with this method is that it increases the reliability during a finite time-period. However, the disadvantage is that 200 % extra resources are required.

Error correcting code

Using an ECC to protect data from errors goes outside FPGAs and simple variants will give protection for most faults without using large amount of resources.

Hamming Code: This ECC is a simple method that is able to correct single-bit errors caused by SEUs in memories such as BRAM and distributed memory. However, the disadvantage is that it is not able to correct multi-bit errors caused by MBUs.

Active Partial Reconfiguration

This technique is focused on protecting the configuration memory from SEUs and is a good way of correcting errors in the configuration memory. However, the disadvantage is that both the error detection and error correction have a substantial delay which can affect the functionality of an application.

Time redundancy

This technique is focused on protecting the design from SETs without the overhead TMR uses. This technique is able to mask SETs by using redundant registers instead of using e.g. TMR on the whole design. However, the disadvantage is that a delay is required between the clocks used by the redundant registers. In some applications this delay may be difficult and even impossible to implement without lowering the clock frequency of the design.

3.3 Multi-word and Multi-bit Upsets

In the case of an SEU there is a possibility that the radiation will affect more than one bit and also more than one word in the memory. This probability differs from architecture to architecture and also depending on the radiation levels. Multi-word upsets are harder to simulate and will increase the amount of logic needed for simulating upsets. For the chosen mitigation techniques it will not impact the functionality of the techniques and the comparison between them. Multi-word upsets will for these reasons not be simulated.

Multi-bit upsets have a bigger impact on the functionality of the mitigation techniques where the chosen ECC can repair a single bit error but not a multi-bit error. The probability for a multi-bit upset differs, as mentioned before, and for the FPGA used, the probability differs from approximately 1.5 %-28 %, as mentioned before. A value of 10 % will occur when the radiation is around $20\,\mathrm{MeV} - \mathrm{cm}^2/\mathrm{mg}$-$40\,\mathrm{MeV} - \mathrm{cm}^2/\mathrm{mg}$ [20] and is the value chosen for the simulations in BRAM and distributed memory. Because the ECC used can only repair single-bit errors, only single-bit and double-bit errors are simulated.

4

Mitigation Techniques for FPGAs

FPGAs have become a strong candidate on the market for electronics in both avionics and aerospace for its flexibility and short TTM. The problem is that FPGA includes more electronics susceptible for SEUs and SETs. Many studies have therefore been done on how to mitigate the chances of such errors. Some of these techniques will be presented below.

The circuit in Fig. 4.1 will be used as a reference circuit to describe the functionality behind the mitigation techniques that can protect against SETs. The signals A and B goes into an AND gate and if both A and B are '1' the output, Y, will be '1' , otherwise it will be '0'. Signal Y will then go into a register, saving the value and outputting it every time Clk goes to '1'.

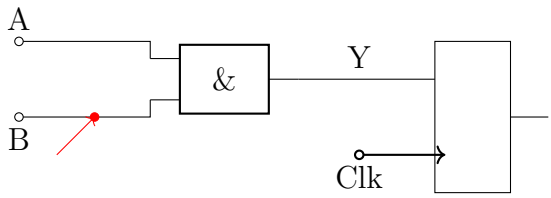


Figure 4.1: Refence circuit with an AND gate and a register where a wire is getting hit by a particle causing an SET

Now assume that the circuit in Fig. 4.1 is hit by a particle with enough energy to flip the value of B, see Fig. 4.2. This fault may happen just before the rising edge of Clk which would result in the wrong value becoming saved in the register, see Fig. 4.2.

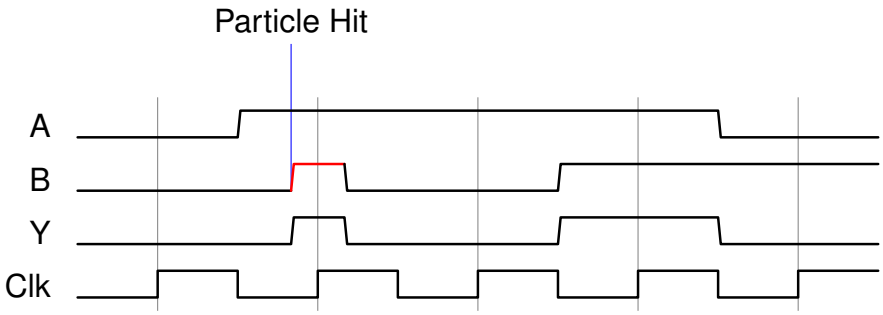


Figure 4.2: Timing-table for circuit in Fig. 4.1

An SET is a temporary error while an SEU, in for example a LUT, is a permanent error and will change the functionality of the design. In Table IIa the LUT

for the AND gate in Fig. 4.3 is shown. If the particle hits the LUT instead of a wire, it might result in an erroneous LUT as shown in Table IIb. This effect would change the behavior and result in the timing-table seen in Fig. 4.4.

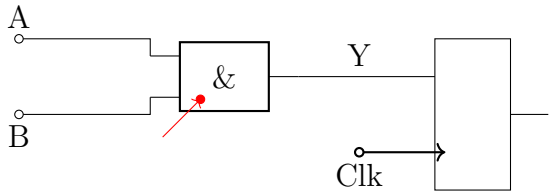


Figure 4.3: Reference circuit with an AND gate and a register where the LUT is getting hit by a particle causing an SEU

Table II: Possible effects on a LUT when hit by a particle.

(a) LUT for AND gate in Fig. 4.3 before particle hit

A	B	Y
0	0	0
0	1	0
1	1	1
1	0	0

(b) LUT for AND gate in Fig. 4.3 after particle hit

A	B	Y
0	0	0
0	1	0
1	1	1
1	0	1

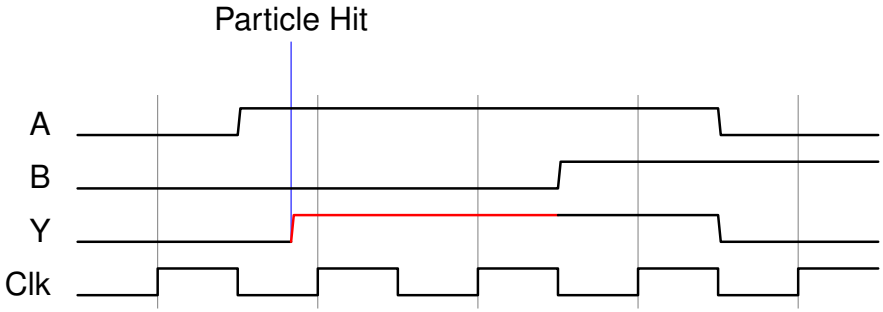


Figure 4.4: Timing-table for circuit in Fig. 4.3

The results from the simulations will be compared to a basic fault tolerant model where the reliability for the reference design that doesn't use any mitigation technique is calculated using:

$$R_{reference}(t) = e^{-\lambda \cdot t}$$
$$\lambda = Failure\ rate$$
$$t = time$$

The configuration memory, the BRAM and the distributed memory have different failure rates and are presented as λ_{config}, λ_{BRAM} and λ_{LUTRAM} respectively

when they need to be distinguished from each other. The results will also include the mean time to failure (MTTF) which can be calculated from the model using:

$$MTTF = \int_0^\infty R(t)\ dt$$

which for the reference-design becomes:

$$MTTF_{reference} = \int_0^\infty e^{-\lambda \cdot t}\ dt = \left[\frac{1}{-\lambda} e^{-\lambda \cdot t}\right]_0^\infty = 0 - \frac{1}{-\lambda} = \frac{1}{\lambda}$$

4.1 Triple modular redundancy

The most used and studied technique is TMR where the logic is implemented in three copies, executed concurrently, with a majority-voter at the output. This technique can cover both SEUs and SETs occurring in one of the copies at any moment or a permanent error in the logic of one copy [27]. If a permanent fault occurs in one of the copies, the design is not be able to support a temporary SET or an SEU in any of the copies still functioning.

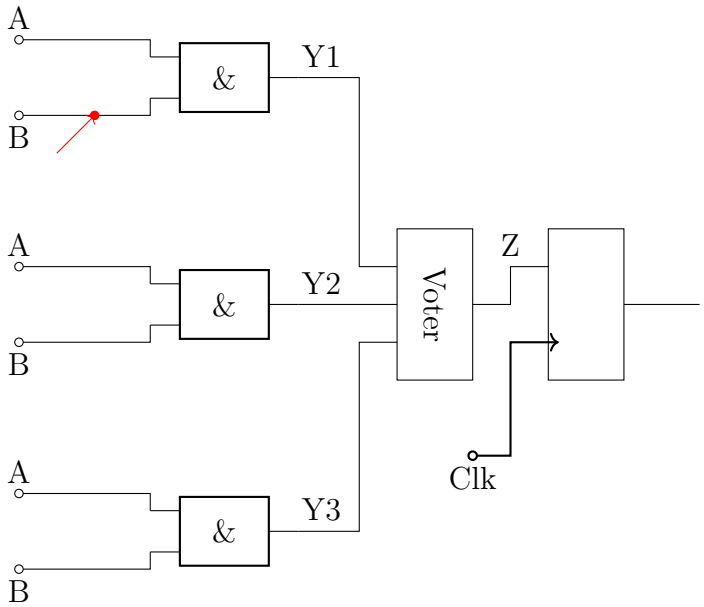


Figure 4.5: Circuit with an AND gate and a register using TMR where a wire is getting hit by a particle, causing an SET

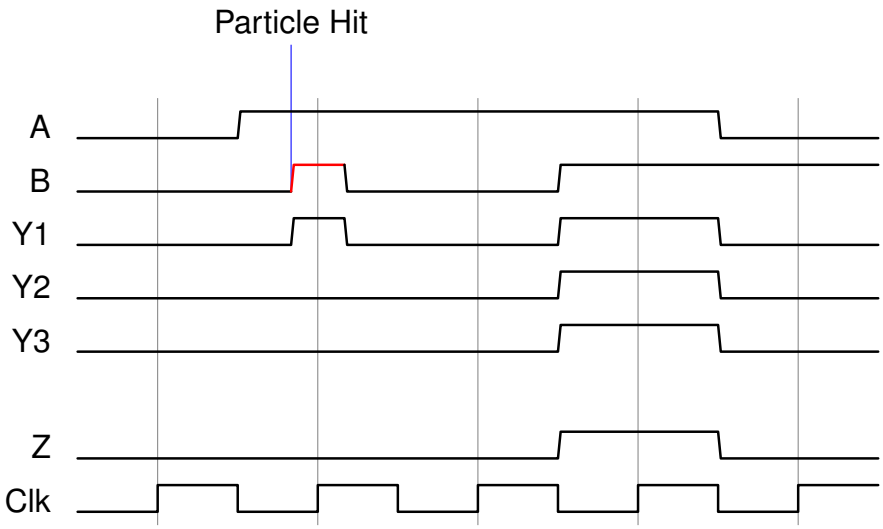


Figure 4.6: Timing-table for circuit in Fig. 4.5

Fig. 4.5 shows a design using TMR getting hit by a particle in one of its wires causing an SET, resulting in the behavior seen in Fig. 4.6. In Fig. 4.7 the same design is shown but where the particle hits the LUT for the AND gate instead of the wire, causing the error seen in Table II. The resulting behavior can be seen in Fig. 4.8.

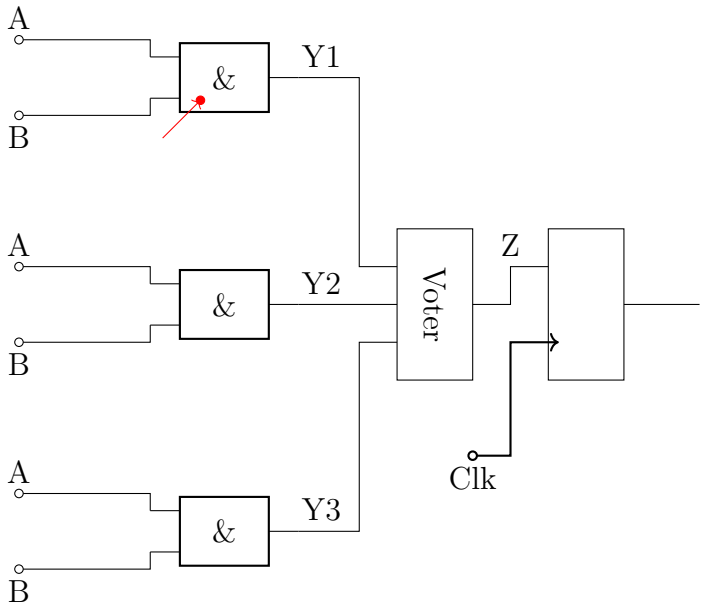


Figure 4.7: Circuit with an AND gate and a register using TMR where a LUT is getting hit by a particle, causing an SEU

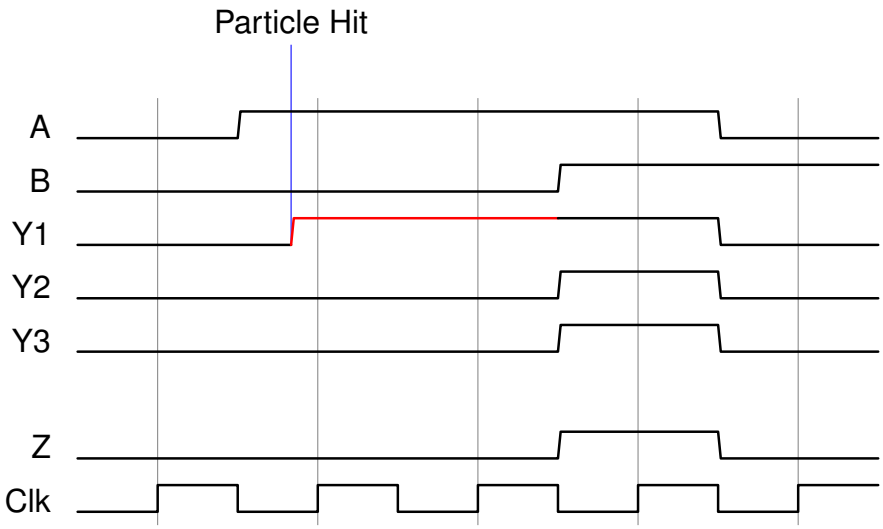


Figure 4.8: Timing-table for circuit in Fig. 4.7 with particle hitting the LUT for the AND gate

Several different versions of TMR are used for mitigation errors where the simplest one takes the whole logic and implements three copies of it together with a single voter at the output [2], as for example the design in Fig. 4.5. More advanced versions use several voters at the end to ensure that a faulty voter can not affect the result. Both a single voter and multiple voters use more than 200 % extra area compared to the original design and are not always possible. When using partial TMR, the whole logic is not triplicated, instead only critical parts of the design are implemented with redundancy. However, this simplification has a lower overhead area at the cost of decreased reliability and relies on a design analysis to find the critical parts of the design.

The reliability-model for TMR is calculated below and doesn't take into account the failure rate for the voter.

$P_n(t)$ = Probability of system being in state n at time t
$\mathbf{Q}$ = Transition matrix

The TMR technique has three states, P_2 where all three copies work, P_1 where two copies work and the voter can mask the behavior from the third copy and P_0 where two or more copies are not working resulting in system failure. To find the reliability $R(t)$ the equation below should be solved for P_2 and P_1 which added together gives $R(t)$.

$$\mathbf{P}'(\mathbf{t}) = \mathbf{P}(\mathbf{t}) \cdot \mathbf{Q}$$
$$\mathbf{P}'(\mathbf{t}) = [P_2'(t)\ P_1'(t)\ P_0'(t)]$$
$$\mathbf{P}(\mathbf{t}) = [P_2(t)\ P_1(t)\ P_0(t)]$$

$$\mathbf{Q} = \begin{bmatrix} -3\lambda & 3\lambda & 0 \\ 0 & -2\lambda & 2\lambda \\ 0 & 0 & 0 \end{bmatrix}$$

$$P_2'(t) = -3\lambda \cdot P_2(t)$$
$$P_1'(t) = 3\lambda \cdot P_2(t) - 2\lambda \cdot P_1(t)$$
$$P_0'(t) = 2\lambda \cdot P_1(t)$$

$$Laplace\ transform$$
$$f'(t) \propto s\tilde{f}(s) - f(0) \rightarrow s \cdot \tilde{\mathbf{P}}(\mathbf{s}) - \mathbf{P}(\mathbf{0}) = \tilde{\mathbf{P}}(\mathbf{s}) \cdot \mathbf{Q}$$
$$where$$
$$\mathbf{P}(\mathbf{0}) = [P_2(0)\ P_1(0)\ P_0(0)] = [1\ 0\ 0]$$

$$s \cdot \tilde{P}_2(s) - 1 = -3\lambda \cdot \tilde{P}_2(s)$$
$$s \cdot \tilde{P}_1(s) - 0 = 2\lambda \cdot \tilde{P}_2(s) - 2\lambda \cdot \tilde{P}_1(s)$$
$$s \cdot \tilde{P}_0(s) - 0 = 2\lambda \cdot \tilde{P}_1(s)$$

$$\tilde{P}_2(s) = \frac{1}{s+3\lambda} \rightarrow P_2(t) = e^{-3\lambda \cdot t}$$
$$\tilde{P}_1(s) = \frac{3\lambda \cdot \tilde{P}_2(s)}{s+2\lambda} = \frac{3\lambda}{(s+2\lambda)(s+3\lambda)} = \frac{3}{s+2\lambda} - \frac{3}{s+3\lambda}$$
$$\rightarrow P_1(t) = 3e^{-2\lambda \cdot t} - 3e^{-3\lambda \cdot t}$$

$$R_{TMR}(t) = P_2(t) + P_1(t) = 3e^{-2\lambda \cdot t} - 2e^{-3\lambda \cdot t}$$

In Fig. 4.9 the reliability is plotted over time and shows that TMR-based design should have better reliability than the reference design during a finite time-period. After this finite time-period, the reliability is much lower for the TMR-based design than for the reference design.

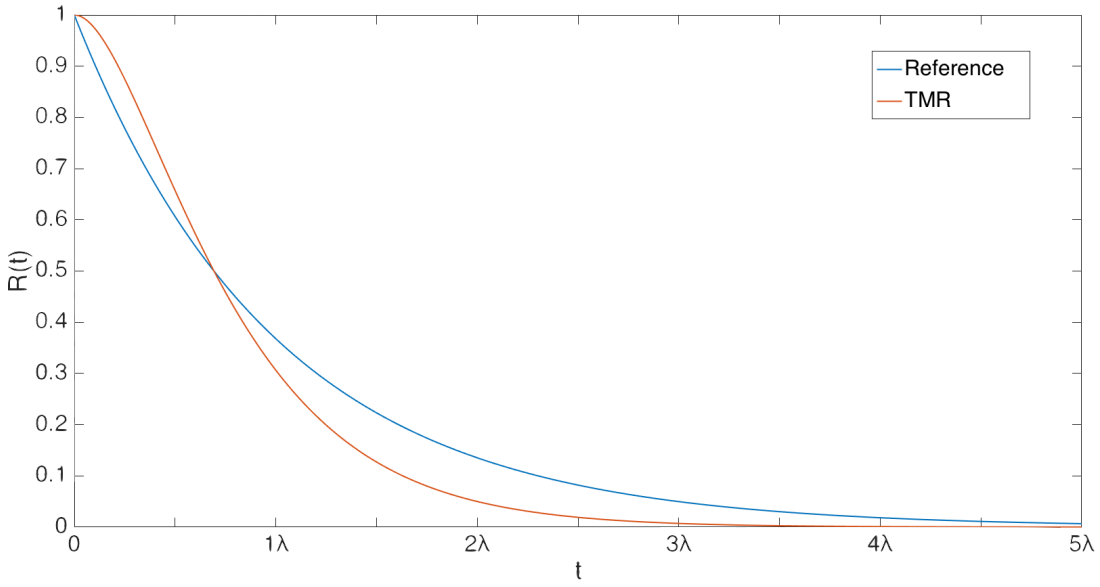


Figure 4.9: The reliability of reference design and TMR design without voter

The MTTF for TMR is calculated as follows:

$$MTTF_{TMR} = \int_0^\infty 3e^{-2\lambda \cdot t} - 2e^{-3\lambda \cdot t} = \left[\frac{3}{-2\lambda}e^{-2\lambda \cdot t} - \frac{2}{-3\lambda}e^{-3\lambda \cdot t}\right]_0^\infty$$
$$= 0 - \left(\frac{3}{-2\lambda} - \frac{2}{-3\lambda}\right) = \frac{3}{2\lambda} - \frac{2}{3\lambda} = \frac{5}{6\lambda}$$

These results show that $MTTF_{TMR} \approx 0.83 \cdot MTTF_{reference}$.

4.2 Error-correcting code

ECC is a method for protecting data by adding redundant data or parity bits. Different methods have different overhead and can handle different amounts of faults. An example of parity bits is adding one more bit to the word, being '1' if the amount of ones in the word is even, otherwise '0'. This method makes it possible to detect if one of the bits has flipped value. The limitation of this method is the fact that an even number of bit-flips cannot be able to be detected. The method will not be able to correct the bit either because there is no way of knowing which bit has the wrong value.

This project has used one specific kind of ECC which is called Hamming code and is described below.

4.2.1 Hamming code

Hamming code is a form of error-correction code capable of detecting 2-bit errors and correct a single-bit error. It uses parity bits and for n parity bits it will be able to protect the data

$$Data\ bits:\ d = 2^n - n - 1 \tag{4.1}$$

giving a total block length of

$$Block\ length:\ b = 2^n - 1 \tag{4.2}$$

To be able to detect 2-bit errors it will need an additional parity bit and the technique for that will not be described in this project. The structure of a 7-bit block with three parity bits together with the coverage of each parity bit can be seen in Table III[2, 28].

Table III: Structure and parity coverage for a 7-bit hamming-code with three parity bits

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Block data	d_4	d_3	d_2	p_3	d_1	p_2	p_1
p_1	X		X		X		X
p_2	X	X			X	X	
p_3	X	X	X	X			

From Table III it can be seen which data-bit is covered by which parity bit and the data-coverage will look like in Table IV

Table IV: Data coverage for each parity bit

	d_4	d_3	d_2	d_1
p_1	X		X	X
p_2	X	X		X
p_3	X	X	X	

If the data $x =$ "1011" should be converted it would get the form $y =$ "101$p_3$1p_2p_1" and the parity bits are calculated using Table IV. Each parity bit takes the data-bits it covers and counts how many of them are '1'. If the amount is even the parity becomes a '0' and if it is odd the parity becomes a '1'.

$$p_1 : d_1 =' 1',\ d_2 =' 1' \ and\ d_4 =' 1' \rightarrow 3 \rightarrow p_1 =' 1'$$
$$p_2 : d_1 =' 1',\ d_3 =' 0' \ and\ d_4 =' 1' \rightarrow 2 \rightarrow p_2 =' 0'$$
$$p_3 : d_2 =' 1',\ d_3 =' 0' \ and\ d_4 =' 1' \rightarrow 2 \rightarrow p_3 =' 0'$$

The resulting block becomes $y =$ "1010101" and when it is decoded the placements for all '1' is taken and then counted and converted as when encoded.

$$1 : 0001$$
$$3 : 0011$$
$$5 : 0101$$
$$7 : 0111$$
$$Sum : 0224$$
$$Result : 0000$$

If there is an error the result will point to the position of the error. If for example "1010101" gets an error and becomes "1000101" the result will be

$$1 : 0001$$
$$3 : 0011$$
$$7 : 0111$$
$$Sum : 0123$$
$$Result : 0101$$

pointing at position 5 where the error occurred. This means that this bit should be flipped in order to obtain the original value [29].

This process can also be done using matrix-calculation and creating the code generator matrix G and the parity-check matrix H can be done from the process

above. Checking if the amount is even and odd can be done using mod-function.

$$\left(\begin{pmatrix} x_1 & x_2 & \cdots & x_d \end{pmatrix} \begin{pmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,b} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,b} \\ \vdots & \vdots & \ddots & \vdots \\ g_{d,1} & g_{d,2} & \cdots & g_{d,b} \end{pmatrix}\right) mod\ 2 = \begin{pmatrix} y_1 & y_2 & \cdots & y_b \end{pmatrix}$$

$$\left(\begin{pmatrix} y_1 & y_2 & \cdots & y_b \end{pmatrix} \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{b,1} & h_{b,2} & \cdots & h_{b,n} \end{pmatrix}\right) mod\ 2 = \begin{pmatrix} r_1 & r_2 & \cdots & r_n \end{pmatrix}$$

Start with the generator matrix for the same size as the example above

$$\left(\begin{pmatrix} d_4 & d_3 & d_2 & d_1 \end{pmatrix} \begin{pmatrix} g_{1,1} & g_{1,2} & g_{1,3} & g_{1,4} & g_{1,5} & g_{1,6} & g_{1,7} \\ g_{2,1} & g_{2,2} & g_{2,3} & g_{2,4} & g_{2,5} & g_{2,6} & g_{2,7} \\ g_{3,1} & g_{3,2} & g_{3,3} & g_{3,4} & g_{3,5} & g_{3,6} & g_{3,7} \\ g_{4,1} & g_{4,2} & g_{4,3} & g_{4,4} & g_{4,5} & g_{4,6} & g_{4,7} \end{pmatrix}\right) mod\ 2$$
$$= \begin{pmatrix} d_4 & d_3 & d_2 & p_3 & d_1 & p_2 & p_1 \end{pmatrix}$$

The first column in the G matrix should only let d_4 through, second should let d_3 through, third should let d_2 through and fifth should let d_1 through.

$$\left(\begin{pmatrix} d_4 & d_3 & d_2 & d_1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & g_{1,4} & 0 & g_{1,6} & g_{1,7} \\ 0 & 1 & 0 & g_{2,4} & 0 & g_{2,6} & g_{2,7} \\ 0 & 0 & 1 & g_{3,4} & 0 & g_{3,6} & g_{3,7} \\ 0 & 0 & 0 & g_{4,4} & 1 & g_{4,6} & g_{4,7} \end{pmatrix}\right) mod\ 2$$
$$= \begin{pmatrix} d_4 & d_3 & d_2 & p_3 & d_1 & p_2 & p_1 \end{pmatrix}$$

The columns that are left can be found from Table IV where p_1 uses d_1, d_2 and d_4, p_2 uses d_1, d_3 and d_4 and p_3 uses d_2, d_3 and d_4.

$$\left(\begin{pmatrix} d_4 & d_3 & d_2 & d_1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}\right) mod\ 2$$
$$= \begin{pmatrix} d_4 & d_3 & d_2 & p_3 & d_1 & p_2 & p_1 \end{pmatrix}$$

The parity-check matrix is actually Table III rotated 90°

$$\left(\begin{pmatrix} d_4 & d_3 & d_2 & p_3 & d_1 & p_2 & p_1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\right) mod\ 2 = \begin{pmatrix} r_1 & r_2 & r_3 \end{pmatrix}$$

The Hamming code can correct single-bit errors and will therefore be able to correct 90 % of all injected errors in BRAM and distributed memory. This is true as long as a second error is not injected to the same address before the first one has been repaired. This possibility is minor and if ignored the reliability-model for the BRAM will become:

$$R_{ECC-BRAM}(t) = e^{-0.1\lambda_{BRAM}t}$$

with a MTTF of

$$\begin{aligned} MTTF_{ECC-BRAM} &= \int_0^\infty e^{-0.1\lambda_{BRAM}\cdot t}\, dt = \left[\frac{1}{-0.1\lambda_{BRAM}} e^{-0.1\lambda_{BRAM}\cdot t}\right]_0^\infty \\ &= 0 - \frac{1}{-0.1\lambda_{BRAM}} = \frac{10}{\lambda_{BRAM}} \end{aligned}$$

The same model can be applied to the distributed memory by changing λ_{BRAM} to λ_{LUTRAM}.

4.3 Active partial reconfiguration

Some Xilinx devices support perform partial reconfiguration, which is the reprogramming of a portion of the configuration memory in an FPGA without changing the whole configuration memory. However, the possibility of performing a full reconfiguration of the configuration memory is also available.

Usually, when the configuration memory of an FPGA is configured, the device is held in reset while a design is loaded by an external controller. However, when performing active partial reconfiguration or APR, the FPGA doesn't have to enter a reset mode since only a portion of the configuration memory is reconfigured without affecting the rest of the FPGA. This means that part of the FPGA can still be up and running while the selected portion of the configuration memory is reprogrammed.

The APR technique can be used to remove errors with the soft error mitigation (SEM) intellectual property (IP) core provided by Xilinx. The configuration memory uses a CRC-based error detection method and if e.g. a 1-bit error is detected, the error can be repaired based on an ECC algorithm using APR. However, a delay may be experienced when the error detection as well as the repair processes are used. This means that the configured application may result in disruption or degradation of service until the FPGA configuration memory is repaired.

The Xilinx SEM IP core also provides the ability to detect and correct multiple-bit errors by using the CRC-based enhanced repair method together with APR.

In Fig. 4.10 it can be seen how the LUT from Table IIb has been repaired after a certain delay.

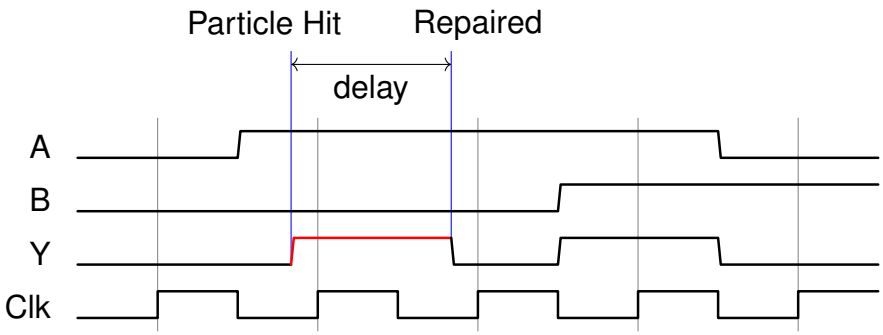


Figure 4.10: Timing-table for circuit in Fig. 4.3 when APR is active

4.4 Scrubbing

Scrubbing is an error-mitigation technique that runs error detection on the memory either continuously or periodically. When an error is detected, the data is restored to a previously known error-free condition by the use of redundancy. This means that the redundant data has to be stored in an external memory. This approach is useful in FPGAs for detecting and correcting SEUs in the configuration memory [30].

Modern FPGAs perform scrubbing in the configuration memory by using in-circuit configuration interfaces such as the internal configuration access port (ICAP) interface used in Xilinx devices. When a designer chooses to use the Xilinx scrubbing technique for the configuration memory of an FPGA, the repair process can be achieved without interrupting the functionality of the design [31]. However, the time it will take for an error to be repaired depends on the number of sensitive configuration frames which is proportional to the total percentage of essential bits.

4.5 Time-redundancy

Time-redundancy uses the transient properties of an SET to mitigate possible errors. A voltage-peak caused by an energized particle or wave will dissipate and by using several registers, triggering at different moments in the clock-cycle, an error can be masked. The difference between the clocks must be longer than the SET, otherwise it will affect more than one register and the voter will give the wrong value [2].

If the circuit in Fig. 4.1 uses timing-redundancy the circuit can look as Fig. 4.11 instead.

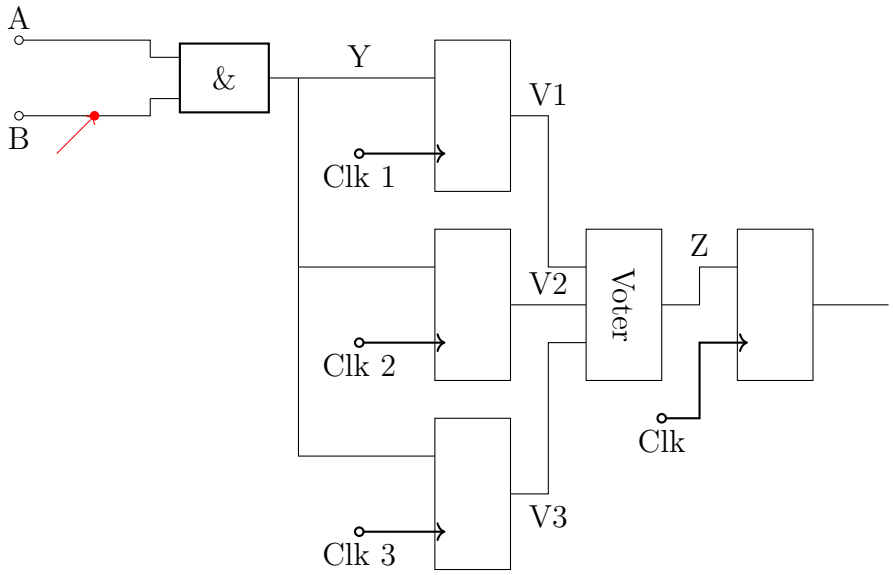
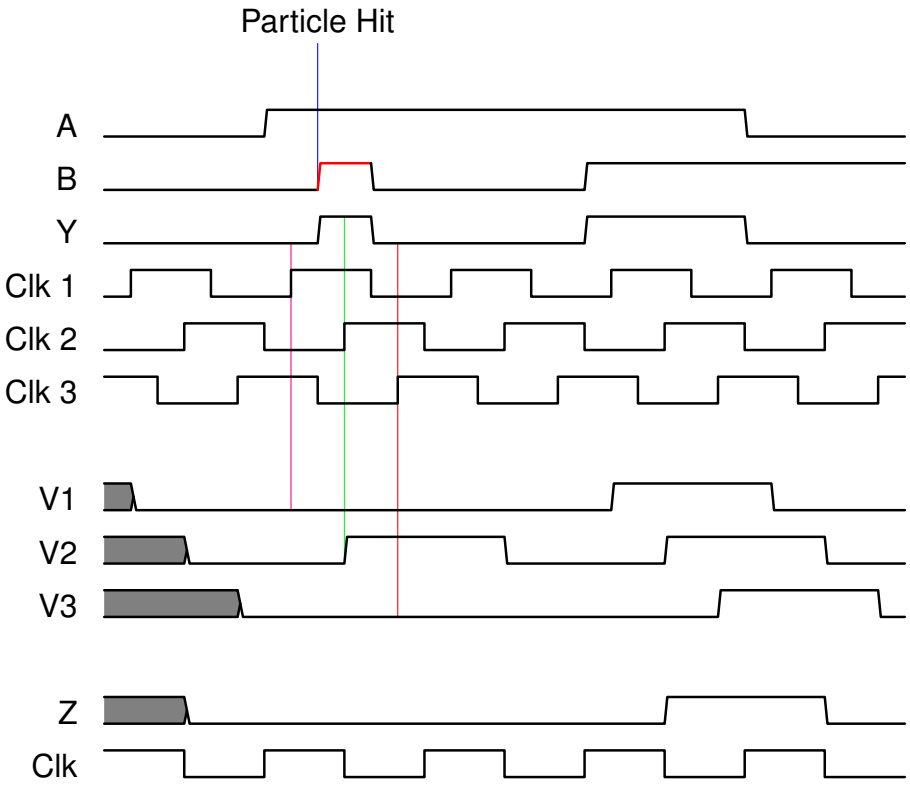


Figure 4.11: Circuit with an AND gate and three registers implementing time redundency

The timing-table will then look like in Table V where it can be seen that only register 2 will receive the wrong value. This error can then by masked by the voter using majority decision.

Table V: Timing-table for circuit in Fig. 4.11



5
Evaluation Method

This chapter describes how the project was performed. First, the test designs are described as well as the different methods used for error injection. Furthermore, the testing-protocol is described and the process of method selection is mentioned.

5.1 Error injection

SEUs can occur in all forms of memory, including BRAM, distributed memory and configuration memory. The main forms of memories that were injected with errors are BRAM, distributed memory and configuration memory. The errors in the configuration memory were injected using Xilinx SEM IP core which includes an error injection function. On the other hand, BRAM and distributed memory were injected using specific test-platforms developed during the project.

5.1.1 BRAM

Error were injected in BRAM by creating a component that wraps around the BRAM memory. The component needs a BRAM memory in form of a simple dual-port memory and will inject an error when its injecterror pin goes high. It will then inject an error at the address specified on the "read address" port by flipping one or several bits in the word and then writing the word back to the memory. A random generator decides which bit or bits in the word are flipped.

5.1.2 Distributed memory

To inject errors in the distributed memory, a specific error injection block was designed. This block works as an interface between the application and the distributed memory and is able to inject errors simultaneously with the regular read and write performed by the application. This error injection block is generic and not design specific, which means that it can also be used for injecting errors in BRAM.

The reason for using a different type of error injection block for the distributed memory was to test the functionality of injecting errors simultaneously with regular read and write, which enables the possibility of injecting errors in randomly selected addresses.

5.1.3 Configuration memory

To inject errors in the configuration memory, an SEM IP core from Xilinx was used. It used a 40-bit address together with a strobe to change states or inject errors. This SEM IP core also offers the possibility of using APR to repair errors in the configuration memory.

The SEM IP core was controlled by a state machine, with the purpose of injecting errors at random addresses in the design and monitoring the status-signals from the SEM IP core. The state-machine kept the SEM IP core in observation state at all times except for when an error was injected. Depending on the settings selected for the SEM IP core, the status-signals differed and if no error-classification is activated all errors are classified as uncorrectable and essential, causing the SEM IP core to stop. When this happens, the SEM IP core is restarted and the error injection continues until the tested design signals an error in the system.

5.1.4 Single Event Transient

To inject pulses in the designs simulating SETs a block that creates pulses from the clock was implemented. The block then uses a random-generator to decide which port out from the block that should receive the pulse. The different ports are then wired to different parts of the design where an OR gate is implemented, simulating the result from a SET.

5.2 Test designs

The designs that were used to test the different mitigation techniques were similar in nature, but helped verify that the results were accurate. In general, each design included three simulated sensors, one with saved values and two using arithmetic functions, and each sensor was connected to a block with a PID controller and a transfer function. The whole design with all three sensors, three PID controllers and three transfer functions was then duplicated with comparators connected to the outputs. The designs use some form of memory to store values for the input of the sensors. This was developed in order to test the vulnerability of the different types of memories used in FPGAs.

For both test design 1 and test design 2, linear-feedback shift register (LFSR) modules were developed to randomize the error injection address and bit in the memories. This includes both configuration memory as well as application memory such as BRAM and distributed memory.

5.2.1 Test design 1

This design uses BRAM to store all values for one of the sensors, 16384 36-bit words. Each of these words is used to create a sine wave and a single error will result in a sample with wrong value is getting sent to the output within one period of the sine wave. A single error will result in a complete system failure because the two parts of the design will output different values. These errors are monitored by 6

voters where each transfer function output, and its duplicated version, is compared by 2 voters to decrease the possibility of a voter-error causing a system failure not to show.

The error injection function works, as mentioned above, by implementing a simple dual-port memory and injecting an error at the word being read at the moment the injection-pin goes high.

5.2.2 Test design 2

Test design 2 was developed in order to test the vulnerability of the distributed memory available in FPGAs. The values stored in the distributed memory are critical for the proper function of the application. In case an error occurs in a critical address of the memory such as an SEU, the whole system will result in a failure.

The distributed memory was divided in 4 KB blocks where each block consists of 2^{10} addresses with 32 bit data. The reference design uses 2 memory blocks, where only $\approx 10\%$ of the addresses in each block are critical for the design.

Another important aspect of test design 2 is that the error injection in the memory is performed simultaneously with the regular read and write performed by the application. Compared to test design 1, this method enables the opportunity of injecting errors in randomly selected addresses instead of design specific addresses. However, in order to achieve this functionality, an extra clock is needed. In this case, the extra clock should be at least 4 times faster than the clock used for regular read and write. For this specific design, the extra clock used for error injection in the distributed memory was set to 100 MHz. The regular clock used by the application was set to 17 MHz.

A specific module was designed for the error injection functionality in the distributed memory. The module is easy to implement and works as an interface between application and the memory. This error injection module is not design specific and can also be used for injecting errors in BRAM.

5.3 Testing

The different mitigation techniques were tested by error injection performed in BRAM, distributed memory and configuration memory. Time redundancy (TR) was also tested using SET error injection but because of lack of information for pulse length and probability this form of error injection was not used on the other mitigation techniques.

5.3.1 BRAM, distributed memory and configuration memory

Firstly, the two designs were tested without any mitigation technique to obtain reference data and then each of the designs went through three different tests. Error injection was performed in BRAM, distributed memory and configuration memory

separately. Each of the tests for BRAM and distributed memory was made by injecting 5000 errors. The tests for configuration memory was done in the same fashion except that 10 000 errors was injected. The amount of observed failures as well as the MTTF were logged for each test.

After obtaining the reference results, the different mitigation techniques were tested separately as well as combined with other mitigation techniques. This was done because a technique that only protects e.g. the BRAM is not able to perform better than the failure rate for the configuration memory in a real-life setting. All of the injected errors were done using random-generators, with a uniform distribution, to achieve as realistic results as possible. Unfortunately, there are certain factors that limit the obtained results in contrast to realistic results for the hardware used. For example, it is hard to simulate the probability that a single event affects several bits in close proximity. The statistics for this probability varies with FPGA architecture techniques, radiation characteristics and how close the memory bits are placed next to each other.

5.3.2 Single event transient pulses

Some studies have looked at the length of SET pulses but these results are based on other architectures and cannot be used as a reference for the Kintex 7 FPGA. These studies show test-results with pulse lengths in the span of 100 ps-1400 ps [21, 32]. The logic needed to create the pulse did however not allow for shorter pulse lengths than 3 ns. The tests where therefore instead made to check the behavior of TR when the pulse lengths are around the same size as the delay between the register-clocks. The tests are made under the premise that the design runs in a clock-frequency lower that what is possible and therefore the delay between the register-clocks can be changed without affecting the system-clock. The pulses from the earlier described injection-block was wired to 32 different points in the design while trying to vary the distance to the next register. The different pulse lengths used in the tests are integer multiplication from the shortest pulse length possible, resulting in 3 ns, 6 ns and 9 ns. It is most interesting to see the behavior when the pulse length is just longer than the delay why the clock-delays were chosen to 2 ns, 5 ns and 8 ns.

6
Results

First, this chapter presents the results of each mitigation technique that can protect against SEUs as well as combinations of these techniques. The results include failure rates for error injection in BRAM, distributed memory and configuration memory. The resource utilization and maximum clock-frequency for each technique and test design are presented. Furthermore, a comparison among all SEU mitigation techniques is presented in terms of reliability, performance, overhead area, power consumption and overall system cost. Finally, the results from the SET simulations are presented for both the reference design and TR-based design with different delays between the clocks.

The MTTF is presented for all techniques. The mean number of SEUs per failure was obtained by the simulations and the MTTF was then calculated in years based on the data obtained from the Xilinx failures in time (FIT) rate calculator. The data from the FIT rate calculator was based on a neutron flux of 4361 $\mathrm{N/cm^2/h}$ which is the nominal value for latitude and longitude coordinates of Chalmers University of Technology at an altitude of 34 000 feet with moderate solar activity.

6.1 Reference Design

Reference tests were performed on both test designs by injecting errors into the configuration memory for design 1 and 2, into the BRAM for design 1 and into the distributed memory for design 2. The results from these tests were used as reference for the comparison of the different mitigation techniques. Table VI shows the maximum clock frequency and the total on-chip power for the two test designs.

Table VI: Maximum clock frequency and total on-chip power of reference test designs 1 and 2

Test Design	**Clock Frequency** [MHz]	**Total on-chip Power** [W]
Design 1	76.823	0.596
Design 2	17	0.821

In addition to the maximum clock frequencies presented in Table VI, design 1 uses an extra clock of 100 MHz for injecting errors in the configuration memory. Design 2 also uses an extra clock of 100 MHz for injecting errors into the configuration memory as well as the distributed RAM.

Table VII shows the utilization of the FPGA resources available for design 1 and 2. The major difference between the two test designs can be observed in the utilization percentage of BRAM and LUTRAM. Design 1 uses the BRAM resources to store values that are critical for the proper function of the application. This is reflected in the utilization percentage of BRAM for design 1. On the other hand, design 2 uses the distributed memory resources to store values that are critical for the proper function of the application. This is reflected in the utilization percentage of LUTRAM for design 2.

Table VII: Utilization percent of reference test designs 1 and 2

Resource	**Design 1 utilization** [%]	**Design 2 utilization** [%]
LUT	2.77	5.16
LUTRAM	0.00	2.20
FF	0.75	0.46
BRAM	8.09	0.00
DSP48	17.14	7.62
I/O	1.60	2.80
BUFG	6.25	15.62
MMCM	10.00	10.00

Table VIII shows the results of the error injections. A test of 10 000 errors injections was performed in the configuration memory for each test design. Furthermore, a test of 5000 error injections was performed in the BRAM for test design 1 and in the distributed memory for test design 2. The results are presented as the total number of observed failures and the calculated MTTF.

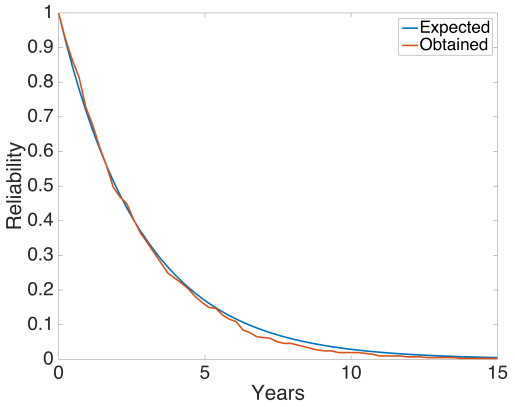
The observed failures of test design 1 are 413 failures per 5000 injected errors for the BRAM test and 68 failures per 10 000 injected errors for the configuration memory test. The mean number of SEUs was $\approx$ 12 SEUs for the BRAM test and $\approx$ 147 SEUs for the configuration memory test which was calculated to 2.818 and 8.578 years respectively.

The observed failures of test design 2 are 154 failures per 5000 injected errors for the distributed memory test and 74 failures per 10 000 injected errors for the configuration memory test. The mean number of SEUs was $\approx$ 32 SEUs for the distributed memory test and $\approx$ 135 SEUs for the configuration memory test, which were calculated to 1.883 and 7.866 years respectively.

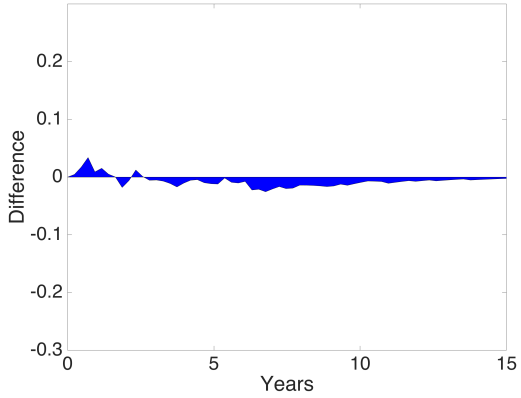
Table VIII: Total number of injections, observed failures and MTTF of reference test designs 1 and 2

	Design 1	**Design 2**
Total BRAM injections	5000	0
Observed failures in BRAM	413	N/A
BRAM MTTF [Years]	2.818	N/A
Total distributed memory injections	0	5000
Observed failures in distributed memory	N/A	154
Distributed memory MTTF [Years]	N/A	1.883
Total configuration memory injections	10 000	10 000
Observed failures in configuration memory	68	74
Configuration memory MTTF [Years]	8.578	7.866

Figs. 6.1 and 6.2 show the plots for the expected and obtained reliability over time for the BRAM and configuration memory of test design 1. The expected reliability was calculated based on $R(t) = e^{-\lambda t}$ where λ is the failure rate obtained from $\lambda = 1/MTTF$.

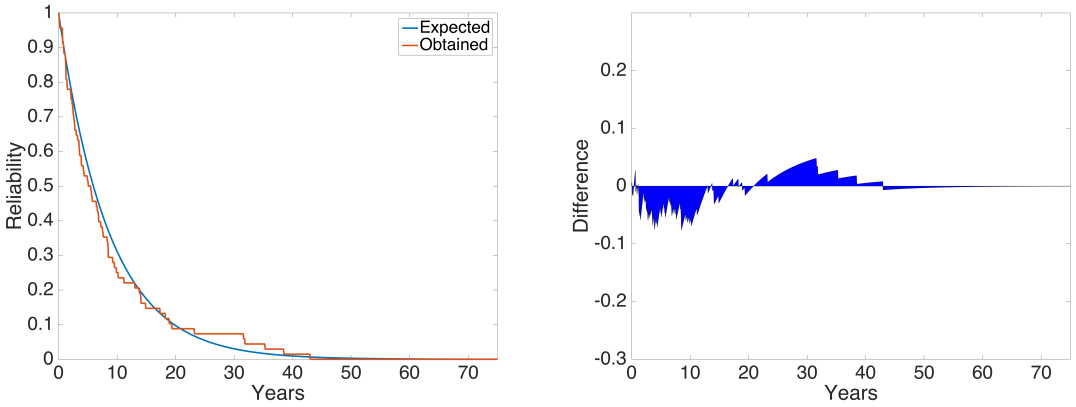


(a) Reliability obtained from test plotted together with expected value from model



(b) The difference between results observed at test and expected value from model

Figure 6.1: Reliability results for the BRAM test of reference design 1



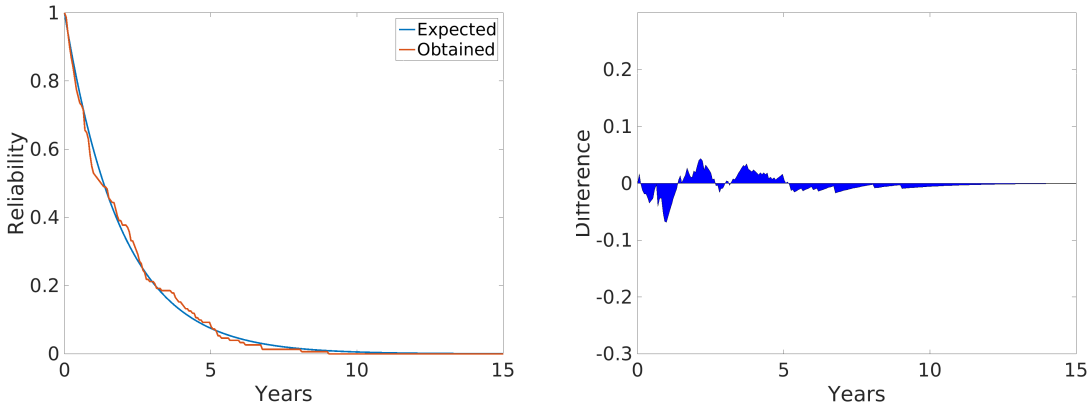
(a) Reliability obtained from test plotted together with expected value from model

(b) The difference between results observed at test and expected value from model

Figure 6.2: Reliability results for the configuration memory test of reference design 1

Figs. 6.3 and 6.4 show the plots for the expected and obtained reliability over time for the distributed memory and configuration memory of test design 2.

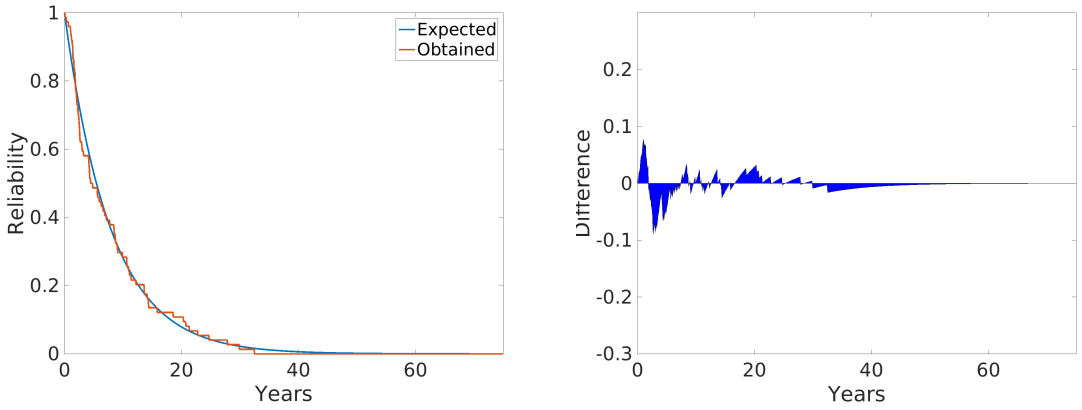
As shown in the plots, the obtained reliability results for both reference designs are within the expected range. However, small deviations from the expected values can be observed.



(a) Reliability obtained from test plotted together with expected value from model

(b) The difference between results observed at test and expected value from model

Figure 6.3: Reliability results for the distributed memory test of reference design 2



(a) Reliability obtained from test plotted together with expected value from model

(b) The difference between results observed at test and expected value from model

Figure 6.4: Reliability results for the configuration memory test of reference design 2

6.2 Triple modular redundancy

TMR was implemented on both test designs and tests were performed in the same manner as for the reference designs. The results from these tests were used to compare this specific mitigation technique with theoretical calculations as well as to compare it with other mitigation techniques. Table IX shows the maximum clock frequency and the total on-chip power for both test designs.

Table IX: Maximum clock frequency and total on-chip power of TMR-based test designs 1 and 2

Test Design	**Clock Frequency** [MHz]	**Total on-chip Power** [W]
Design 1	76.823	1.231
Design 2	17	1.912

The maximum clock frequencies are shown in Table IX. However, as presented in the reference design, both TMR test designs also use an extra clock of 100 MHz for error injection.

Table X shows the utilization of the FPGA resources available for both test designs when implementing TMR. When comparing to the reference design, some resources such as LUTs, LUTRAMs and flip-flops (FFs) are, as expected, 3 times larger when implementing TMR.

Table X: Utilization percent of TMR-based test designs 1 and 2

Resource	**Design 1 utilization** [%]	**Design 2 utilization** [%]
LUT	8.29	15.36
LUTRAM	0.00	6.60
FF	2.24	1.31
BRAM	24.27	0.00
DSP48	51.43	23.86
I/O	1.60	2.80
BUFG	6.25	28.12
MMCM	10.00	10.00

Table XI shows the results of the error injections. As for the reference, a test of 10 000 error injections was performed in the configuration memory for each test design. Furthermore, a test of 5000 error injections was performed in the BRAM for test design 1 and in the distributed memory for test design 2. The results are presented as the total number of observed failures and the calculated MTTF.

The observed failures of test design 1 are 484 failures per 5000 injected errors for the distributed memory test and 78 failures per 10 000 injected errors for the configuration memory test. The mean number of SEUs was $\approx$ 10 SEUs for the BRAM test and $\approx$ 128 SEUs for the configuration memory test which was calculated to 2.411 and 7.479 years respectively.

Table XI: Total number of injections, observed failures and MTTF of TMR-based test designs 1 and 2

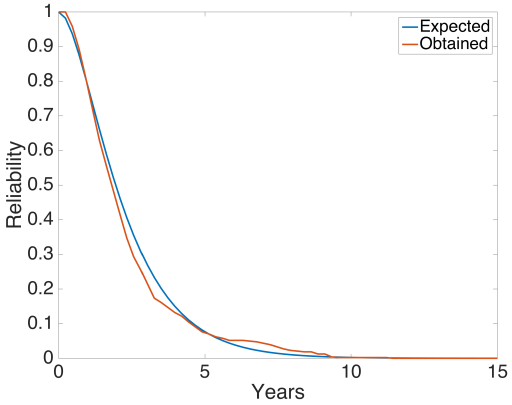
	Design 1	**Design 2**
Total BRAM injections	5000	0
Observed failures in BRAM	484	N/A
BRAM MTTF [Years]	2.411	N/A
Total distributed memory injections	0	5000
Observed failures in distributed memory	N/A	184
Distributed memory MTTF [Years]	N/A	1.581
Total configuration memory injections	10 000	10 000
Observed failures in configuration memory	78	89
Configuration memory MTTF [Years]	7.479	6.529

The observed failures of test design 2 are 185 failures per 5000 injected errors for the distributed memory test and 90 failures per 10 000 injected errors for the configuration memory test. The mean number of SEUs was $\approx$ 27 SEUs for the

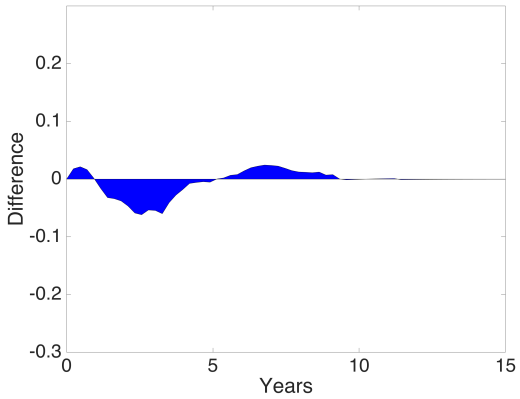
distributed memory test and ≈ 112 SEUs for the configuration memory test which was calculated to 1.581 and 6.529 years respectively.

As shown in Table XI, the number of observed failures is higher than for the reference design. This result is obtained because the TMR design has a higher reliability than the reference during a specific amount of time and after that the reliability is much lower compared to the reference.

Figs. 6.5 and 6.6 show the plots for the expected and obtained reliability over time for the BRAM and configuration memory of test design 1. The expected reliability was calculated based on $R(t) = 3e^{-2\lambda t} - 2e^{-3\lambda t}$ for a TMR design where λ is the failure rate obtained from the reference simulations.

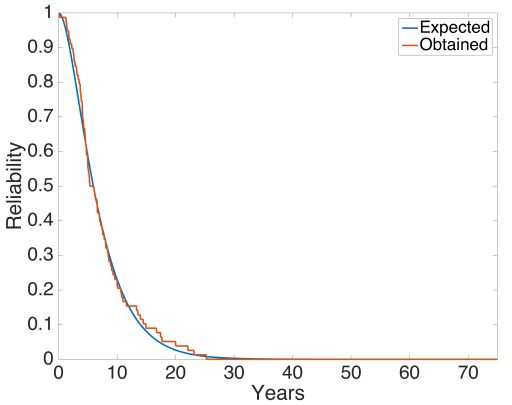


(a) Reliability obtained from test plotted together with expected value from model

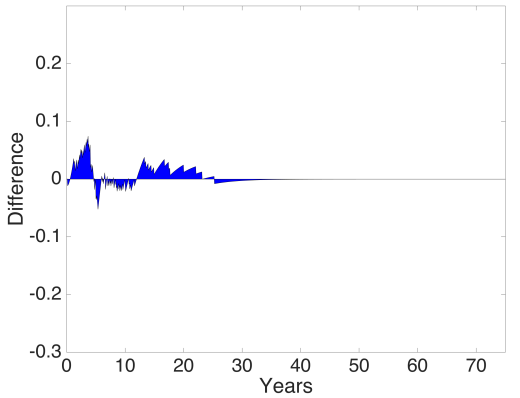


(b) The difference between results observed at test and expected value from model

Figure 6.5: Reliability results for the BRAM memory test of TMR design 1



(a) Reliability obtained from test plotted together with expected value from model

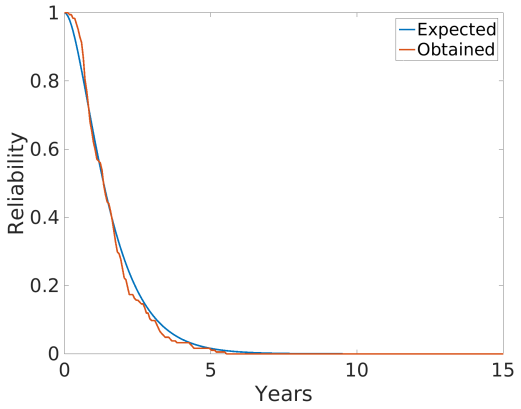


(b) The difference between results observed at test and expected value from model

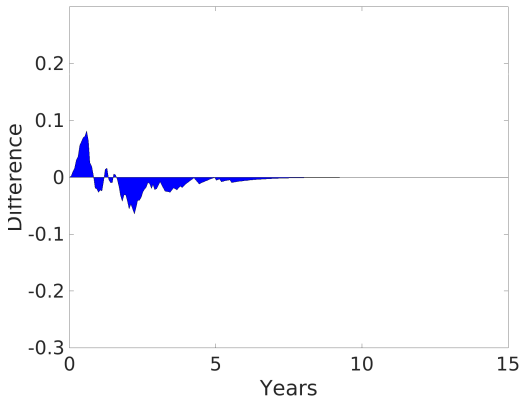
Figure 6.6: Reliability results for the configuration memory test of TMR design 1

Figs. 6.7 and 6.8 show the plots for the expected and obtained reliability over time for the distributed memory and configuration memory of test design 2.

The obtained reliability results for both TMR designs are within the expected range. However, small deviations from the expected values can be observed because of a low accuracy that depends on the small number of injected errors.

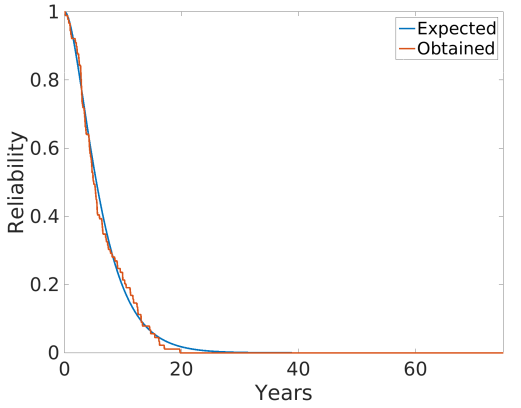


(a) Reliability obtained from test plotted together with expected value from model

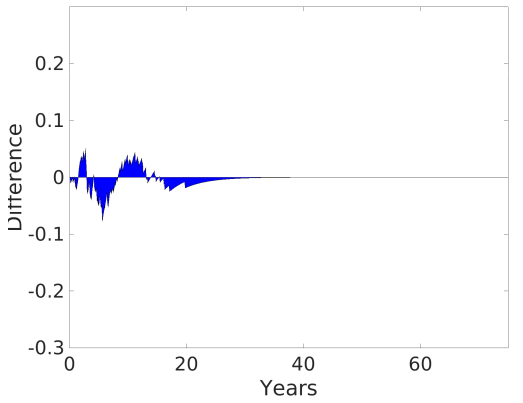


(b) The difference between results observed at test and expected value from model

Figure 6.7: Reliability results for the distributed memory test of TMR design 2



(a) Reliability obtained from test plotted together with expected value from model



(b) The difference between results observed at test and expected value from model

Figure 6.8: Reliability results for the configuration memory test of TMR design 2

6.3 Error-Correcting Code

A Hamming-based ECC was implemented on both test designs and tests were performed in the same manner as for the reference designs. The results from these tests were used to compare this specific mitigation technique with theoretical calculations as well as to compare it with other mitigation techniques. Table XII shows the maximum clock frequency and the total on-chip power for both test designs.

Table XII: Maximum clock frequency and total on-chip power of ECC-based test designs 1 and 2

Test Design	**Clock Frequency** [MHz]	**Total on-chip Power** [W]
Design 1	76.823	0.658
Design 2	17	0.824

The maximum clock frequencies are shown in Table XII. However, as presented in the reference and TMR-based designs, both ECC-based test designs also use an extra clock of 100 MHz for error injection.

Table XIII shows the utilization of the FPGA resources available for both test designs when implementing an ECC. When comparing to the reference design, some resources such as LUTs, LUTRAMs and FFs are larger due to the extra area needed for implementing the Hamming-based ECC.

Table XIII: Utilization percent of ECC-based test designs 1 and 2

Resource	**Design 1 utilization** [%]	**Design 2 utilization** [%]
LUT	2.91	5.40
LUTRAM	0.00	2.60
FF	0.81	0.48
BRAM	9.44	0.00
DSP48	17.14	7.62
I/O	1.60	2.80
BUFG	6.25	15.62
MMCM	10.00	10.00

Table XIV shows the results of the error injections. As for the reference, a test of 10 000 error injections was performed in the configuration memory for each test design. Furthermore, a test of 5000 error injections was performed in the BRAM for test design 1 and in the distributed memory for test design 2. The results are presented as the total number of observed failures and the calculated MTTF.

The observed failures of test design 1 are 40 failures per 5000 injected errors for the distributed memory test and 60 failures per 10 000 injected errors for the configuration memory test. The mean number of SEUs was ≈ 125 SEUs for the BRAM test and ≈167 SEUs for the configuration memory test which was calculated to 29.167 and 9.722 years respectively.

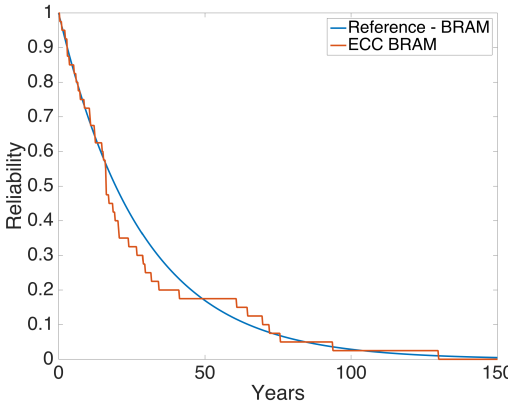
Table XIV: Total number of injections, observed failures and MTTF of ECC-based test designs 1 and 2

	Design 1	**Design 2**
Total BRAM injections	5000	0
Observed failures in BRAM	40	N/A
BRAM MTTF [Years]	29.167	N/A
Total distributed memory injections	0	5000
Observed failures in distributed memory	N/A	15
Distributed memory MTTF [Years]	N/A	18.764
Total configuration memory injections	10 000	10 000
Observed failures in configuration memory	60	76
Configuration memory MTTF [Years]	9.722	7.641

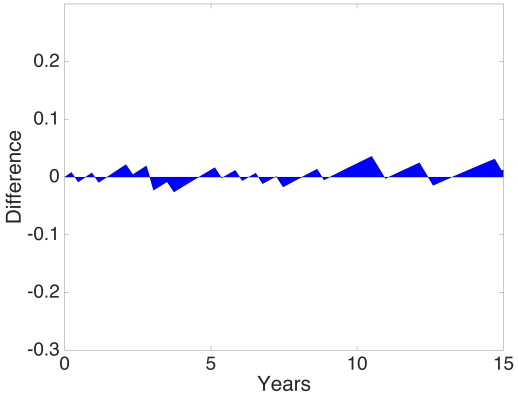
The observed failures of test design 2 are 15 failures per 5000 injected errors for the distributed memory test and 76 failures per 10 000 injected errors for the configuration memory test. The mean number of SEUs was ≈322 SEUs for the distributed memory test and ≈131 SEUs for the configuration memory test which was calculated to 18.764 and 7.641 years respectively.

As shown in Table XIV, the number of observed failures is much lower for the BRAM and the distributed memory when using the Hamming-based ECC than for the reference design. The only observed failures in the BRAM and the distributed memory are due to multi-bit errors which constitute about 10 % of all injected errors. This is because a Hamming-based ECC is able to detect, but not correct, multi-bit errors.

Figs. 6.9 and 6.10 show the plots for the obtained reliability over time for the distributed memory and configuration memory of test design 1.

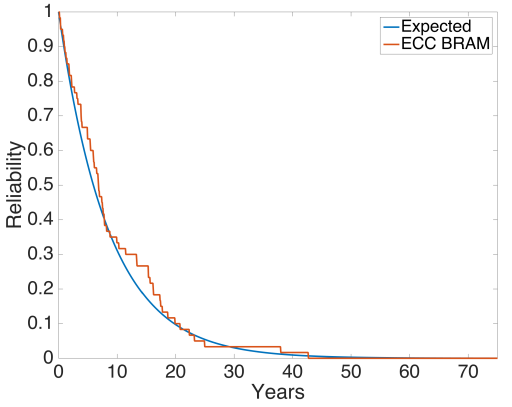


(a) The Reliability from test plotted together with expected value from model

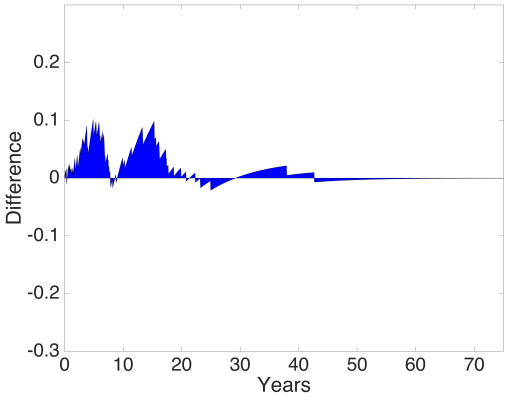


(b) The difference between results observed at test and expected value from model

Figure 6.9: Reliability results for the BRAM memory test of design 1 with ECC in BRAM



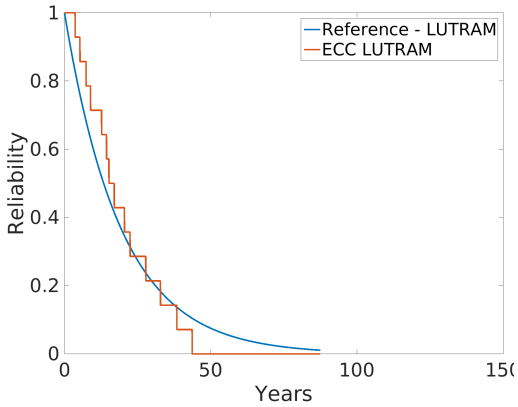
(a) Reliability obtained from test plotted together with expected value from model



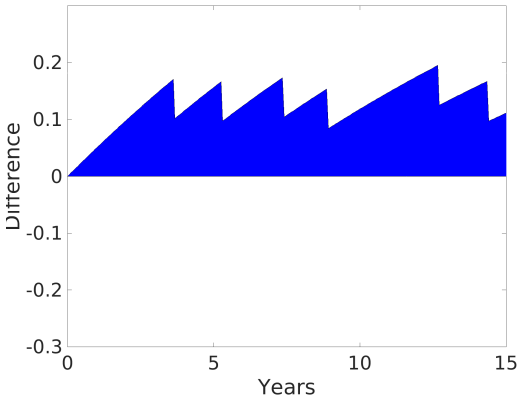
(b) The difference between results observed at test and expected value from model

Figure 6.10: Reliability results for the configuration memory test of design 1 with ECC in BRAM

Figs. 6.11 and 6.12 show the plots for the obtained reliability over time for the distributed memory and configuration memory of test design 2. As shown in Fig. 6.7, the reliability is much higher over time for the ECC-based test design 2, whereas the reliability over time for the configuration memory should be close to the obtained reference values. As previously mentioned, the low accuracy of the reliability plots depend on the small number of injected errors.

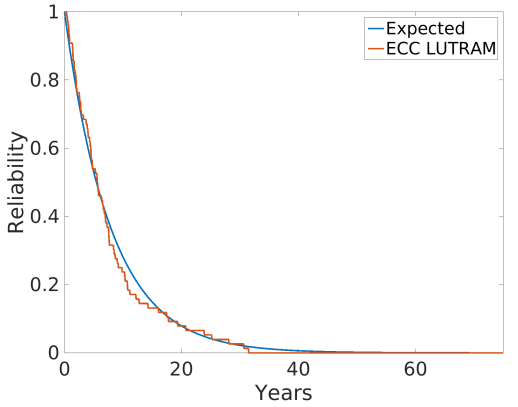


(a) The Reliability from test plotted together with expected value from model

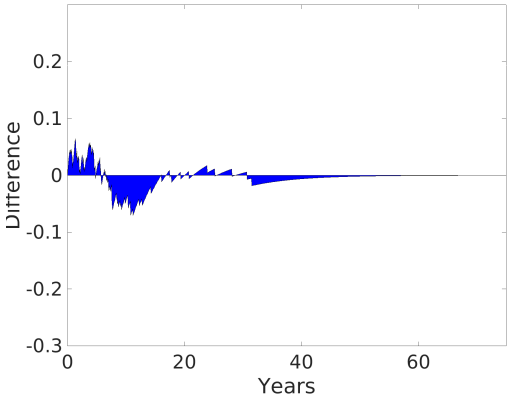


(b) The difference between results observed at test and expected value from model

Figure 6.11: Reliability results for the distributed memory test of design 2 with ECC in the distributed memory



(a) Reliability obtained from test plotted together with expected value from model



(b) The difference between results observed at test and expected value from model

Figure 6.12: Reliability results for the configuration memory test of design 2 with ECC in the distributed memory

6.4 Active partial reconfiguration

The use of Xilinx's APR technique for the configuration memory was implemented on both test designs and tests were performed in the same manner as for the reference designs. Table XV shows the maximum clock frequency and the total on-chip power of the two test designs with APR.

Table XV: Maximum clock frequency and total on-chip power of APR-based test designs 1 and 2

Test Design	**Clock Frequency** [MHz]	**Total on-chip Power** [W]
Design 1	76.823	0.611
Design 2	17	0.828

However, both APR-based test designs also use an extra clock of 100 MHz for error injection and error detection and correction of the configuration memory.

Table XVI shows the utilization of the FPGA resources available for both test designs when implementing APR. When comparing to the reference design, some resources such as LUTs, LUTRAMs, FFs and BRAMs are larger due to the extra area needed for implementing the Xilinx SEM IP core with the APR function enabled.

Table XVI: Utilization percent of APR-based test designs 1 and 2

Resource	**Design 1 utilization** [%]	**Design 2 utilization** [%]
LUT	2.99	5.54
LUTRAM	0.07	2.25
FF	0.86	0.58
BRAM	8.43	0.34
DSP48	17.14	7.62
I/O	1.60	2.80
BUFG	9.38	15.62
MMCM	10.00	10.00

Table XVII shows the results of the error injections. As for the reference, a test of 10 000 error injections was performed in the configuration memory for each test design. No tests were performed for the BRAM and distributed memory in this case since the results are considered to be the same as the ones obtained for the reference test design. The results are presented as the total number of observed failures and the calculated MTTF.

The observed failures of test design 1 are 66 failures per 10 000 injected errors for the configuration memory test. The mean number of SEUs was ≈152 SEUs which was calculated to 8.838 years based on the neutron flux data.

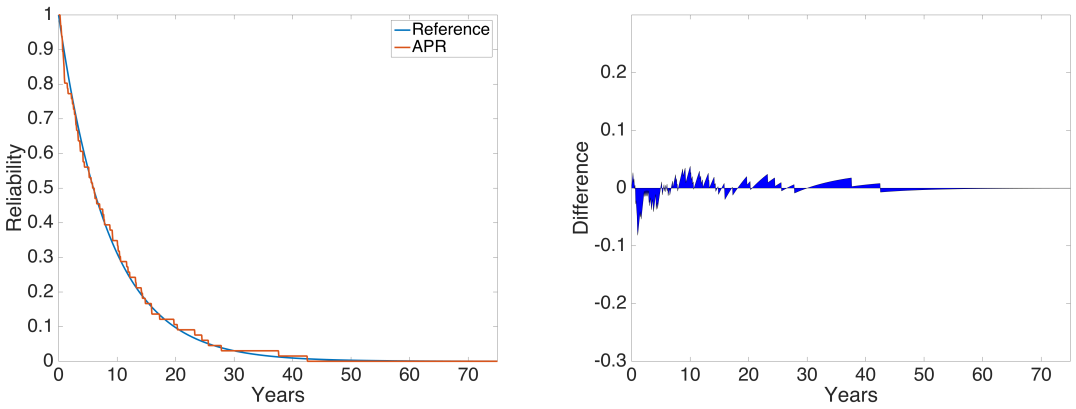
Table XVII: Total number of injections, observed failures and MTTF of APR-based test designs 1 and 2

	Design 1	**Design 2**
Total configuration memory injections	10 000	10 000
Observed failures in configuration memory	66	70
Configuration memory MTTF [Years]	8.838	8.283

The observed failures of test design 2 are 70 failures per 10 000 injected errors for the configuration memory test. The mean number of SEUs was ≈142 SEUs which was calculated to 8.283 years based on the neutron flux data.

As shown in Table XVII, the number of observed failures is lower than for the reference design. However, no major change in the reliability was observed since the values are quite close to the ones obtained for the reference test designs.

Fig. 6.13 shows the plot for the obtained reliability over time for the configuration memory of test design 1. The obtained reliability result for test design 1 with APR is very close to the reference design. However, small deviations from the expected values can be observed because of a low accuracy that depends on the small number of injected errors.

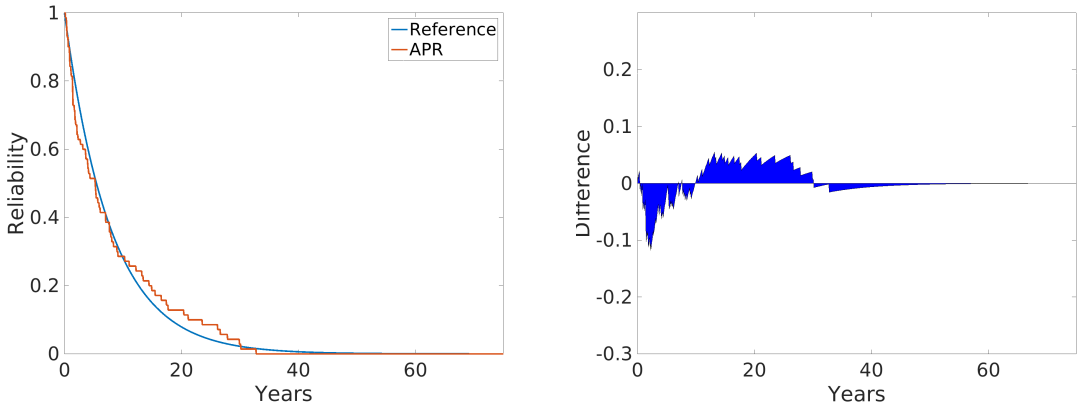


(a) Reliability obtained from test plotted together with expected value from reference

(b) The difference between results observed at test and expected value from reference

Figure 6.13: Reliability results for the configuration memory test when using APR in design 1

Fig. 6.14 shows the plot for the obtained reliability over time for the configuration memory of test design 2. As for test design 1, the obtained reliability result for test design 2 with APR is very close to the reference design.



(a) Reliability obtained from test plotted together with expected value from reference

(b) The difference between results observed at test and expected value from reference

Figure 6.14: Reliability results for the configuration memory test when using APR in design 2

6.5 TMR combined with APR

TMR was combined with the use of Xilinx APR technique for the configuration memory on both test designs. All tests were performed in the same manner as for the reference designs. Table XVIII shows the maximum clock frequency and the total on-chip power respectively for the two TMR test designs with APR.

Table XVIII: Maximum clock frequency and total on-chip power of TMR-based test designs 1 and 2 with APR

Test Design	**Clock Frequency** [MHz]	**Total on-chip Power** [W]
Design 1	76.823	1.247
Design 2	17	1.947

However, both TMR test designs with APR also use an extra clock of 100 MHz for error injection and error detection and correction of the configuration memory. This clock is directly connected to the Xilinx SEM IP core.

Table XIX shows the utilization of the FPGA resources available for both test designs when implementing TMR with APR. When comparing to the reference design, some resources such as LUTs, LUTRAMs, FFs and BRAMs are larger due to the extra area needed for implementing the Xilinx SEM IP core with the APR function enabled.

Table XIX: Utilization percent of TMR-based test designs 1 and 2 with APR

Resource	**Design 1 utilization** [%]	**Design 2 utilization** [%]
LUT	8.50	15.74
LUTRAM	0.06	6.65
FF	2.34	1.43
BRAM	24.61	0.34
DSP48	51.43	22.86
I/O	1.60	2.80
BUFG	12.50	28.12
MMCM	10.00	10.00

Table XX shows the results of the error injections. As for the reference, a test of 10 000 error injections was performed in the configuration memory for each test design. No tests were performed for the BRAM and distributed memory in this case since the results are considered to be the same as the ones obtained for the test design with TMR alone. The results are presented as the total number of observed failures and the calculated MTTF.

The observed failures of test design 1 are 77 failures per 10 000 injected errors for the configuration memory test. The mean number of SEUs was ≈ 130 SEUs which was calculated to 7.5758 years based on the neutron flux data.

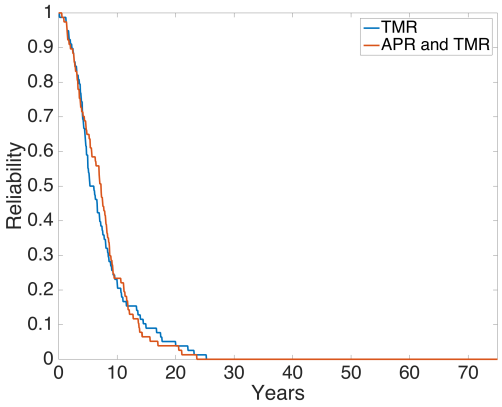
Table XX: Total number of injections, observed failures and MTTF of reference test designs 1 and 2

	Design 1	**Design 2**
Total configuration memory injections	10 000	10 000
Observed failures in configuration memory	77	84
Configuration memory MTTF [Years]	7.576	6.940

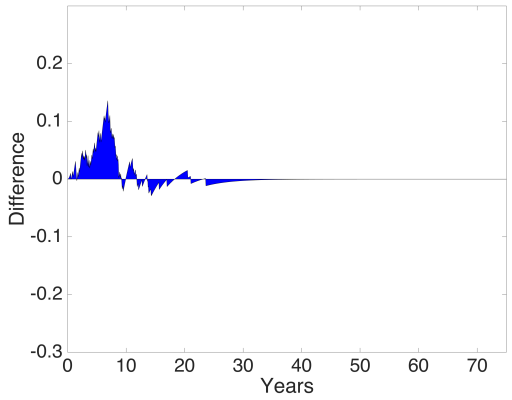
The observed failures of test design 2 are 84 failures per 10 000 injected errors for the configuration memory test. The mean number of SEUs was ≈119 SEUs which was calculated to 6.940 years based on the neutron flux data.

As shown in Table XX, the number of observed failures is lower than for the test designs with TMR alone. As shown for the previous designs with APR, no major change in the reliability was observed since the values are quite close to the ones obtained for the test designs with TMR alone.

Fig. 6.15 shows the plot for the obtained reliability over time for the configuration memory of test design 2.



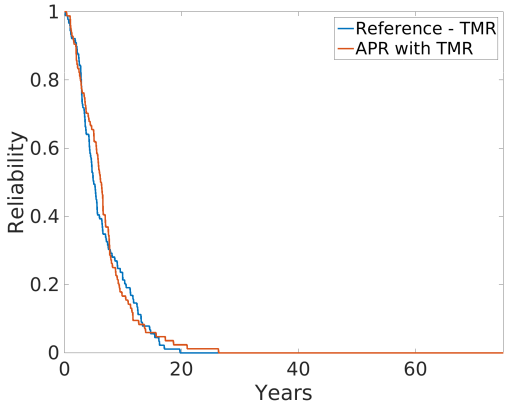
(a) Reliability obtained from test plotted together with expected value from reference



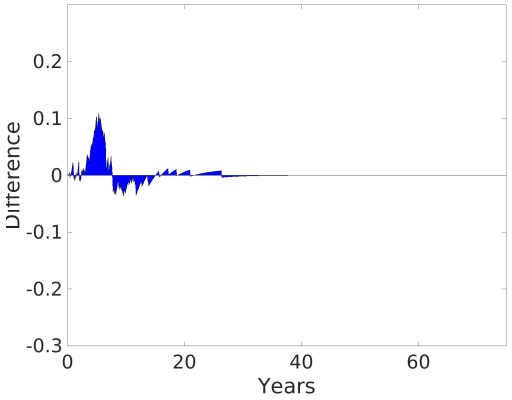
(b) The difference between results observed at test and expected value from reference

Figure 6.15: Reliability results for the configuration memory when using TMR combined with APR in design 1

Fig. 6.16 shows the plot for the obtained reliability over time for the configuration memory of test design 2.



(a) Reliability obtained from test plotted together with expected value from reference



(b) The difference between results observed at test and expected value from reference

Figure 6.16: Reliability results for the configuration memory when using TMR combined with APR in design 2

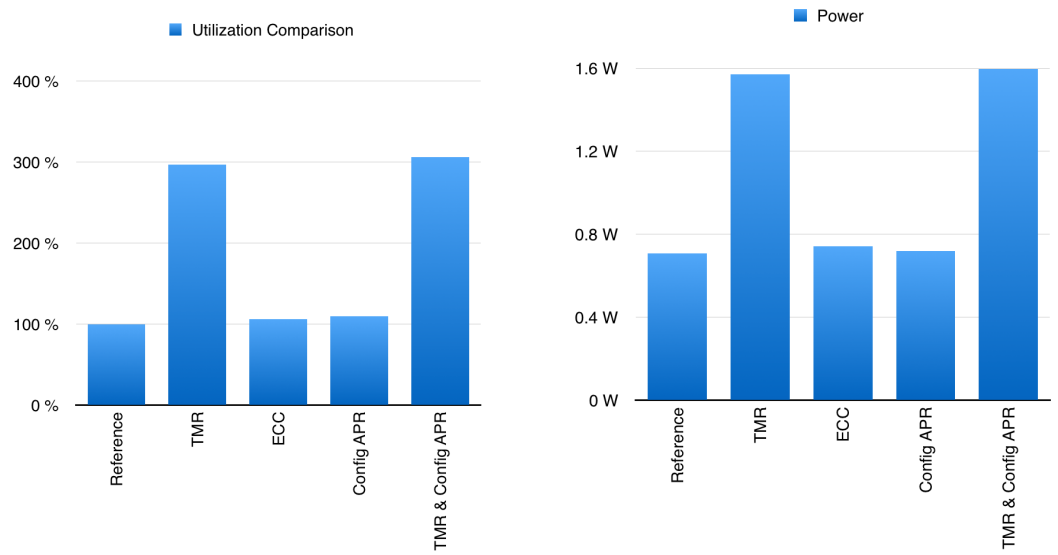
6.6 Comparison

In this section, a comparison between the different techniques is presented. All values are an average of the results from the two test designs. These graphs visually show how the techniques differ in reliability and effectiveness.

6.6.1 Performance, utilization and power

As show in previous sections, the performance of the designs didn't change while performing the different tests. The clock frequencies for both design were constant for all tests. However, when using the APR functionality from the SEM IP core, an extra clock of 100 MHz was required.

As shown in Fig. 6.17 the total average utilization when implementing TMR is three times higher than for the reference design. The utilization is also higher when implementing ECC in the BRAM and distributed memory compared to the reference designs. However the difference in utilization for the ECC in BRAM and distributed memory compared to the reference designs is very limited since just a small percentage is added to the whole design. This extra utilization percentage is obtained when the Hamming code is implemented. Finally, the utilization is also higher when using the APR, both with and without a combination of TMR. The extra utilization percentage obtained when using APR is due to the implementation of the SEM IP core.



(a) Total average utilization for different techniques

(b) Total average power-usage for different techniques

Figure 6.17: Negative effects for different techniques.

The power consumption varies among the different mitigation techniques where TMR is the technique consuming most power due to the use of three redundant systems.

6.6.2 Reliability

The investigated mitigation techniques have different levels of reliability with the ECC in BRAM and distributed memory being the most reliable mitigation technique in terms of injections per failure. Fig. 6.18 shows the failure rate of the different mitigation techniques where application memory is the average result obtained from

BRAM and distributed memory. The average value of BRAM and distributed memory is presented since both memories are used for the same purpose in the design. However, it is important to notice that even though TMR shows a low injection per failure rate, as seen in Fig. 6.18, it does not necessarily mean that the system is less reliable when using TMR.

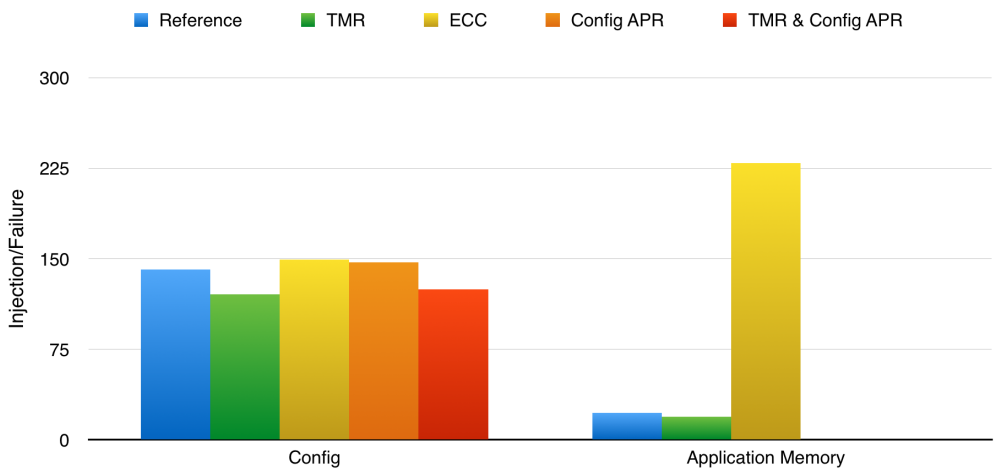


Figure 6.18: Average failure rate comparison among all tested techniques

The TMR technique is still the most reliable method during a finite time-period as shown in Fig. 6.19. Generally, it takes about 5.3 years for both TMR designs to reach the same probability of failure as for the reference designs. Until that point in time, the TMR designs have a remarkably higher reliability than does the reference.

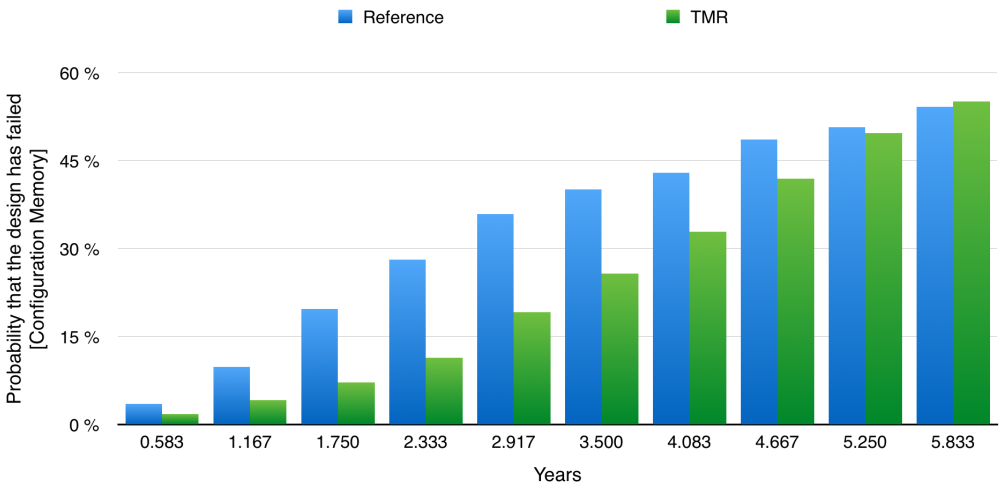


Figure 6.19: Average failure rate comparison between the reference and TMR

6.6.3 Overall system cost

Parameters such as performance, utilization and power will affect the overall system cost. The reference design is considered to be the system having the lowest cost

compared to the design implementing the different techniques used. When using TMR, due to the large utilization and power consumption, the system is considered to have a much higher cost than when using only ECC in BRAM and distributed memory. However, the use of APR is considered to be device specific since it only applies to some Xilinx FPGAs and a Xilinx license is required in order to use the SEM IP core which holds the APR functionality. Therefore, it is important to consider the Xilinx license cost when analyzing the different trade-offs.

6.7 Single Event Transient

The SET simulations were performed by injecting 500 pulses in each design. Three different pulse length were used and tested separately in order to study how different delays between clocks affected the sensitivity. Table XXI shows the number of failures that were observed when injecting the design with these 500 pulses.

Table XXI: Failures observed with 500 injected SETs pulses

	Pulse length		
Design	3 ns	6 ns	9 ns
Reference	26	52	73
TR with 2 ns delay between each clock	15	49	67
TR with 5 ns delay between each clock	0	10	73
TR with 8 ns delay between each clock	0	0	20

Three different pulse lengths of 3 ns, 6 ns and 9 ns were used to simulate SETs. First, the SET test was performed on a reference design. Afterwards, the SET test was performed on three designs using the TR technique on several critical parts of the application. The three TR-based designs had delays of 3 ns, 6 ns and 9 ns between the clocks of the redundant registers.

As seen in Table XXI, the application tested is more sensitive to SETs when the clock frequency is increased and the delay between the redundant registers is decreased.

7
Discussion

This chapter provides a discussion of the different test designs and the results obtained. Firstly, the advantages and drawbacks of the test designs are discussed. Furthermore, the results are analyzed, discussed and compared to the expected values. Finally, the performance of the project is discussed and some recommendations on future work are presented.

7.1 Limitations of test designs and error injection methods

Both test designs developed have the same functionality that is based on PID controllers. The reason for basing the test designs on PID controllers was because they can be used for many flight control applications. For example, PID controllers can be used for stabilizing the flight altitude and speed in unmanned aerial vehicles (UAVs). If a failure occurs due to, e.g., SEUs or SETs in the functionality of the PID controller that is used for stabilizing the altitude or speed of a UAV, then the UAV will lose its stability and that can have serious consequences.

The major difference between the two test designs developed during this project is that they use different types of memories to store values that are critical for the proper function of the application. However, the functionality of these memories is the same. Test design 1 uses BRAMs to store data and test design 2 uses the distributed memory available in the FPGA. The use of distributed memory has the advantage that it has a better performance and it is faster than BRAM. However, distributed memory is only a good choice for a small amount of data. If an application needs to store a large amount of data in memory then BRAM is the best choice. The resources available for distributed memory in an FPGA are low compared to the 445 BRAMs available. Furthermore, based on Xilinx data the distributed memory has lower reliability than the BRAM since the occurrence of an SEU in the distributed memory is about four times higher than for BRAM. Apart from the fact that the two test designs use different types of memories to store values that are critical for the proper function of the application, the error injection methods are also different. Even though the results from injecting errors in different types of memories are going to be the same, the two different error injection methods were developed to cover a broad application spectrum since the methods can be applied for different application specific tests. Both error injection methods have some advantages and drawbacks. Therefore, in order to select the right error injection method when testing the memory, the trade-offs of each method should be carefully

considered.

The first error injection method developed in this project was used for testing test design 1. The advantage with this method is that it only uses one clock, which is the same clock used for the regular read and write. The major drawback with this method is that for some applications where the read and write is continuous, the error injection can only take place in application specific addresses. This means that if an application is designed in a way where a memory read or write occurs every clock cycle, then it is impossible to inject errors in a randomly selected address. The addresses used for the error injection will be the same as the one used by the application. Another drawback with this method is that the error injection can experience a delay since the errors are not injected in the memory simultaneously with the regular read and write. However, this delay may not necessarily affect the failure outcome.

The second error injection method was developed for testing test design 2. The advantage with this method is that the error injection is performed simultaneously with the regular read and write performed by the application. In contrast to the first injection method, the second method can be used for applications where the read and write is continuous. This facilitates the possibility of injecting errors in addresses that are not application specific, such as completely random addresses. The drawback with this method is that an extra clock is needed in order to inject errors into the memory. The frequency of this extra clock has to be at least three times higher than of the clock frequency used by the application for regular read and write. This method will not be feasible for some applications that operate on high frequencies since adding an extra clock that has a three times higher frequency may be impossible to implement.

LFSR modules are used in all error injection methods to decide the error injection address and bit for the configuration memory as well as the application memories such as BRAM and distributed memory. The LFSRs were used to simulate a more realistic situation where a particle can hit anywhere on the FPGA. The problem with LFSRs is that they have a sequence and will not give a fully random output. For a specific bit-width it will always follow that specific sequence and that can have an effect on the simulation results. A way of increasing sequences that follow a specific number is by increasing the number of bits the LFSR uses and then only output some of the bits. An example is a 3-bit LFSR where the three values after a 5 will always be [3, 7, 6]. If a 5-bit LFSR is used instead, where the three least significant bits are extracted, the three values that follow a 5 could either be [3, 6, 4], [3, 7, 6], [2, 5, 2] or [2, 4, 0]. This method gives a result that is more realistic having the value 0 as part of the sequence as well.

If the error injection is started simultaneously as the tested application, the major problem with using the LFSR is that the errors would always be injected at the same addresses and bits. This is an issue since an LFSR has the same sequence of values. To overcome this problem the error injection will not start until a button on the board is pressed. This solution makes the start value of the LFSR sequence to be randomly selected. However, the simulation results will be based on random values as long as the button is pushed with equal possibility anywhere in the sequence. However, when the sequence becomes large, the time it takes for

LFSR to do one loop also becomes long. For a 29 bit LFSR, the period is 536 870 911 which at 100 MHz will take over 5 s. It will then be hard for the tester to push the button at a random point in the sequence. Increasing the LFSR with one bit will approximately double this time span.

It is harder to simulate SETs than SEUs because a SET pulse should be able to hit anywhere at the board with equal probability. To achieve these more realistic results, the block creating the pulses and the paths used to inject SETs at all possible points in the design have to be created manually. Since this method is not feasible, the way the SETs were simulated in this project is by choosing 32 random points in the design where the pulse could hit. Due to logical paths, is a hard task to place these points with different distances from the registers as well as at points where the pulse would have different probabilities of reaching the registers. Therefore, the simulation results differ from realistic results even if the pulse lengths would have been more realistic.

7.2 Evaluation of test results

The discussion about the test results is divided in the same sections as the comparison in Section 6.6. First, a discussion about the negative effects is presented followed by a discussion regarding the reliability as well as a discussion about future work.

7.2.1 Performance, utilization and power

In terms of performance, each test design had a constant clock frequency for all tests. However, the difference in clock frequencies between the two designs are because the applications perform the same task but were developed in different ways. In the case with the techniques affecting the clock frequency it will depend on what is the critical path in the design. In the test designs, the critical path would be the path from after the sensor, through the PID controller and transfer function where it then would meet a register on the feedback path. This means that for example the ECC, which affects the speed of the application memory, could affect the maximum clock frequency if it is part of the critical path. There is also a risk that TMR will affect the maximum clock frequency since adding one or several voters at the output of the design will increasing the pathway. There is also a risk that because of the increased utilization, the tools would not be able to optimize the design in the same manner and in that way affect the maximum possible clock frequency.

The performance requirements for the test platform used in design 2 are higher than for the test platform used in design 1. Therefore, the test platform used for design 2 may be limited to some applications in order to meet all time requirements.

In terms of utilization, the design with the lowest utilization is the reference design. The utilization when implementing TMR was, as expected, three times higher than the reference. In theory it should be more than three times higher because the design is multiplied by three together with one or several voters at the end. The reason why a bigger difference is not obtained in the results is because that certain parts of the design are not multiplied, for example the clock generator. The

utilization of the ECC and APR-based designs is higher than that of the reference design but the difference is not significantly big. Therefore, the extra utilization needed for the implementation of the ECC functionality for BRAM and distributed memory is beneficial when looking at the obtained reliability boost.

The power consumption is a little more than twice for the TMR-based designs than that of the reference. The power consumption was expected to be three times higher, but the reason why this result was not obtained is probably due to a high default FPGA power usage. This extra power usage can be a problem where low power is of great importance, for example in battery powered systems. The power consumption for the ECC and APR-based designs is also higher due to the extra functionality needed but, as for the utilization, the difference is not significantly large.

7.2.2 Reliability

One important aspect when studying the reliability results is that the tests were based on a neutron flux obtained for an altitude of 34 000 feet with moderate solar activity and coordinates for Chalmers University of Technology. This means that these tests consider only the reliability for an application executed on this constant altitude and coordinates. These parameters were just used as an example and may be changed for future tests. For example, an application that is used in an airplane is not going to be constantly at the same coordinates and altitude. Therefore, the relevant average altitude, coordinates and flight hours for an application used in an airborne system should be selected in order to obtain a realistic MTTF.

The results of techniques such as TMR were as expected, following the theory model. However, there was still a significant difference between the obtained and expected values for some tests. The accuracy of the test can be increased by increasing the number of total injected errors. Furthermore, it is important to notice that even though the number of injected errors per failure is lower for the TMR-based designs comparing to the reference, the reliability is still higher for the TMR design than for the reference design the first 5.3 years.

The reliability of the designs was expected to be higher when the repair-based APR method was used. The reason for a high failure rate was that the test designs were built in such a way that a temporary error and disruption due to the repair process were automatically considered as failures even though the configuration memory was repaired. In cases where the design is not sensitive to temporary errors or disruptions, this technique would probably bring better results. For example, a signal processing unit using some form of feedback will give inaccurate results during some time and then continue with the expected functionality. For example, when a particle hits the digital circuit of a surveillance camera the error may result in blurry frames before a self-repair is conducted. A self-repair is much more effective than having a technician repairing the fault.

One of the biggest reliability difference obtained in relation to the reference was for the Hamming-based ECC design used for application specific memories such as BRAM and distributed memory. With this technique, the only uncorrectable errors experienced in the application memories were multi-bit errors, which were

considered to be about 10 % of all events based on Xilinx data. This is because a Hamming-based ECC is only able to correct single-bit errors.

The results from the SET testing is, as mentioned earlier, not based on realistic pulse lengths but could still hint of the behavior of TR. The simulations showed that if the pulse length is shorter than the delay, between the register, the SET would be completely masked. If the pulse length is a bit longer, percentage-wise, than the delay the probability of a SET getting latched was decreased. When this method is used in more real-life settings the delay would be decreased to much smaller value because of the real SET pulse lengths. Something to take in consideration then is that clock-skew as well as the hold times for the registers will have a larger impact on the behavior. This means that clock delays created in the design would have to be a bit larger than needed to counter these effects.

7.3 Future Work

There is still a lot of work to be done in order to have more accurate results when determining the reliability of a system. For example, the number of injected errors in this project was limited to 10000 error injections for the configuration memory and 5000 error injections for BRAM and distributed memory. These injection limits were set due to the limited time of the project. To obtain more accurate results, the number of injected errors should be radically increased in future projects.

Furthermore, the operating clock frequencies of the test designs can be lowered to the point where the chance of obtaining a failure in the configuration memory when using APR is low. Since the repair-based APR method has a detection and a correction delay, the maximum clock frequency of a design should be selected based on this delay.

There are also many other mitigation techniques, as well as combination of techniques, that can be tested in the future. The test platforms developed in this project should work with other mitigation techniques and help evaluating these techniques.

SETs are hard to simulate in an FPGA and the development of a better test platform for simulating the effects of SETs would be preferred. It is desirable that future test platforms studying SETs are able to create simulated pulses with more realistic pulse lengths as well as a simple way of routing these pulses to different points in the design.

It is always good to test the hardware in a real-life high-radiation environment in order to study the accuracy of the simulations. Therefore, it is really recommended to test the FPGA in a radiation chamber if the time, place and budget of the project permits it.

8

Conclusion

It is known that when ICs such as FPGAs are exposed to high levels of radiation, for example particle radiation and electromagnetic radiation, unexpected state changes and temporal voltage pulses can occur which can lead to errors and system failures. These changes, known as SEEs, can cause an FPGA-based application to malfunction and in some cases even pose a threat to human safety. Therefore, the main purpose of this thesis was to develop test platforms to evaluate different design methods in order counteract system failures caused by radiation induced faults.

There are several types of mitigation techniques used for increasing the reliability of FPGAs, such as TMR, ECC, APR, scrubbing, TR, etc. Some of these techniques are studied in this thesis by using the two different test platforms developed. The two test platforms use different types of error injection methods in memories such as BRAMs and distributed memory in order to cover a wide range of designs to test.

The simulation results showed great results for both designs implementing TMR as well as ECC. The circumstances for where these techniques would give the highest improvement do however differ. TMR showed a lower MTTF than the reference design but a higher reliability during the first part of its lifespan. ECC could however protect the application memory from 90 % of all upsets due to the ability of correcting single-bit upsets. The technique that did not show a substantial improvement for the test designs was APR. The reason for this small improvement was probably due to the delay of detection and repair which meant that the designs would be classified as failed before the upset was repaired. Further test of this technique, in applications where temporary incorrect result is allowed, is recommended because of its possibilities to increase the reliability and availability.

SET simulations showed that TR has the possibility of completely masking these erroneous pulses that propagate through the design. The simulations were, however, made using unrealistic long pulse lengths due to limitation in the pulse generation. Further tests with more accurate conditions are recommended to study how clock-skew and hold times for registers would affect the results.

Bibliography

[1] *Storms From the Sun*. Image. NASA. URL: https://www.flickr.com/photos/gsfc/6819077978 (visited on 2016-03-30).

[2] Ricardo Reis Fernanda Lima Kastensmidt Luigo Carro. *Fault-Tolerance Techniques for SRAM-based FPGAs*. Springer, 2006.

[3] Fredrik Brosser and Emil Milh. "SEU Mitigation Techniques for Advanced Reprogrammable FPGA in Space". Chalmers University of Technology, 2014.

[4] M. Zhu, N. Song, and X. Pan. "Mitigation and Experiment on Neutron Induced Single-Event Upsets in SRAM-Based FPGAs". In: *IEEE Transactions on Nuclear Science* 60.4 (Aug. 2013), pp. 3063–3073. ISSN: 0018-9499. DOI: 10.1109/TNS.2013.2270562.

[5] KeFei Xing et al. "Single event upset induced multi-block error and its mitigation strategy for SRAM-based FPGA". In: *Science China Technological Sciences* 54.10 (2011), pp. 2657–2664. ISSN: 1862-281X. DOI: 10.1007/s11431-011-4542-6.

[6] *Dictionary.com*. URL: http://dictionary.reference.com/browse/radiation.

[7] World Health Organization. *What is Ionizing Radiation?* URL: http://www.who.int/ionizing_radiation/about/what_is_ir/en/ (visited on 2016-03-30).

[8] World Nuclear Assosiation. *What is Radiation?* URL: http://www.world-nuclear.org/Nuclear-Basics/What-is-radiation-/ (visited on 2016-03-30).

[9] The Physics Classroom. *The Electromagnetic and Visible Spectra*. URL: http://www.physicsclassroom.com/class/light/Lesson-2/The-Electromagnetic-and-Visible-Spectra (visited on 2016-03-30).

[10] Warren K Sinclair. *Radiation Exposure and High-Altitude Flight: (Commentary No. 12)*. National Council on Radiation Protection and Measurements (NCRP), 1995. ISBN: 978-0-929600-44-4. URL: http://app.knovel.com/hotlink/toc/id:kpREHAFCN3/radiation-exposure-high/radiation-exposure-high (visited on 2016-03-30).

[11] Lembit Sihver et al. "Radiation Environment at Aviation Altitudes and in Space". In: *Radiation Protection Dosimetry* 164 (4 2015), pp. 477–483. ISSN: 0144-8420.

[12] *Flux Calculation*. [Settings: Latitude 40.7° N, Longitude 74° W, Solar modulation 50 %]. URL: http://seutest.com/cgi-bin/FluxCalculator.cgi (visited on 2016-03-29).

[13] I. Kuon, R. Tessier, and J. Rose. "FPGA Architecture: Survey and Challenges". In: *Foundations and Trends in Electronic Design Automation* 2.2 (2008), pp. 135–253. DOI: 10.1561/1000000005.

[14] Xilinx. *Soft Error Mitigation Controller v4.1*. Xilinx, Inc. Sept. 2015.

[15] Paul Graham et al. *Consequences and Categories of SRAM FPGA Configuration SEUs*. Tech. rep. Los Alamos National Laboratory and Xilinx Research Laboratories, Nov. 2003.

[16] N. Battezzati et al. "On the Evaluation of Radiation-Induced Transient Faults in Flash-Based FPGAs". In: *On-Line Testing Symposium, 2008. IOLTS '08. 14th IEEE International*. July 2008, pp. 135–140. DOI: 10.1109/IOLTS.2008.47.

[17] K. F. Zhang, S. F. Chen, and S. Z Xiao. "Anti-radiation design and irradiation test of antifuse FPGA". In: *Applied Mechanics and Materials*. Vol. 380-384. 2013, pp. 3249–3253.

[18] Dagan White. *Considerations Surrounding Single Event Effects in FPGAs, ASICs, and Processors*. Tech. rep. Xilinx, Mar. 2012.

[19] Luca Cassano. *Analysis and Test of the Effects of Single Event Upsets Affecting the Configuration Memory of SRAM-based FPGAs*. Tech. rep. Department of Information Engineering, University of Pisa.

[20] M. Wirthlin et al. "A Method and Case Study on Identifying Physically Adjacent Multiple-Cell Upsets Using 28-nm, Interleaved and SECDED-Protected Arrays". In: *IEEE Transactions on Nuclear Science* 61.6 (Dec. 2014), pp. 3080–3087. ISSN: 0018-9499. DOI: 10.1109/TNS.2014.2366913.

[21] B. Narasimham et al. "Characterization of Digital Single Event Transient Pulse-Widths in 130-nm and 90-nm CMOS Technologies". In: *IEEE Transactions on Nuclear Science* 54.6 (Dec. 2007), pp. 2506–2511. ISSN: 0018-9499. DOI: 10.1109/TNS.2007.910125.

[22] H. Asai et al. "Tolerance Against Terrestrial Neutron-Induced Single-Event Burnout in SiC MOSFETs". In: *IEEE Transactions on Nuclear Science* 61.6 (Dec. 2014), pp. 3109–3114. ISSN: 0018-9499. DOI: 10.1109/TNS.2014.2371892.

[23] J.R. Schwank et al. "Effects of particle energy on proton-induced single-event latchup". In: *Nuclear Science, IEEE Transactions on* 52.6 (Dec. 2005), pp. 2622–2629.

[24] Marty Johnson et al. *Latch-Up*. Tech. rep. Texas Instruments, Apr. 2015.

[25] *Aerospace and Defence*. Xilinx. URL: http://www.xilinx.com/applications/aerospace-and-defense.html (visited on 2016-04-11).

[26] Xilinx. *Device Reliability Report. Second Half 2015*. Version 10.4. Apr. 1, 2016. URL: http://www.xilinx.com/support/documentation/user_guides/ug116.pdf (visited on 2016-04-11).

[27] M. Berg et al. *Using Classical Reliability Models and Single Event Upset (SEU) Data to Determine Optimum Implementation Schemes for Triple Modular Redundancy (TMR) in SRAM-Based Field Programmable Gate Array (FPGA) Devices*. Tech. rep. NASA Goddard Space Flight Center, July 2015.

[28] Zekun Cao et al. "High capacity data hiding scheme based on (7, 4) Hamming code". In: *SpringerPlus* 5 (2016), pp. 1–13.

[29] Brajesh Kumar Gupta and Prof. Rajeshwar Lal Dua. "Article: 30 Bit Hamming Code for Error Detection and Correction with Even Parity and Odd Parity Check Method by using VHDL". In: *International Journal of Computer Applications* 35.13 (Dec. 2011), pp. 31–38.

[30] M. Berg et al. “Effectiveness of internal vs. external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis”. In: IEEE, 2007.

[31] B. Brendan, C. Carl, and T. Chen. *Single-Event Upset Mitigation Selection Guide*. Xilinx, Inc. Mar. 2008.

[32] J. R. Ahlbin et al. “Single-Event Transient Pulse Quenching in Advanced CMOS Logic Circuits”. In: *IEEE Transactions on Nuclear Science* 56.6 (Dec. 2009), pp. 3050–3056. ISSN: 0018-9499.