June 2018
Master's Degree Thesis

Reliable Consensus Voting in FPGAs

Graduate School of Chosun University

Department of Computer Engineering

Umar Afzaal

Reliable Consensus Voting in FPGAs

고신뢰 FPGA 를 위한 동의 보팅 기법

June, 2018

Graduate School of Chosun University

Department of Computer Engineering

Umar Afzaal

Reliable Consensus Voting in FPGAs

Advisor: Prof. Lee, Jeong-A

A thesis submitted in partial fulfillment of the requirements for a Master’s degree

June 2018

Graduate School of Chosun University

Department of Computer Engineering

Umar Afzaal

아프잘 우마 석사학위논문을 인준함

위원장	조선대학교 교수	**신석주**	(인)
위　원	조선대학교 교수	**강문수**	(인)
위　원	조선대학교 교수	**이정아**	(인)

2018년 05월

조선대학교 대학원

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

MUX	Multiplexer
FPGA	Field-programmable Gate Array
PE	Priority Encoder
TMR	Triple Modular Redundancy
ASIC	Application Specific Integrated Circuit
PR	Partial Reconfiguration
SRAM	Static Random Access Memory
CRAM	Configuration RAM
PO	Primary Output
PI	Primary Input
EXOR	Exclusive OR
LUT	Look-up Table
SEE	Single Event Effect
SEU	Single Event Upset
SET	Single Event Transient
NFTVC	Novel Fault-Tolerant Voter Circuit
SFTVC	Simple Fault-Tolerant Voter Circuit
IFTVC	Improved Fault-Tolerant Voter Circuit
MCU	Multiple Cell Upsets
CLB	Configurable Logic Block
DSP	Digital Signal Processing
BRAM	Block RAM

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Reliable Consensus Voting in FPGAs

Umar Afzaal
Advisor: Prof. Lee, Jeong-A, Ph.D.
Department of Computer Engineering
Graduate School of Chosun University

Nanoscale FPGA circuits are more prone to radiation-induced effects in harsh environments because of their memory-based reconfigurable logic fabric. Consequently, for mission- or safety-critical applications, appropriate fault-tolerance techniques are widely employed. The most commonly applied technique for hardening FPGAs against radiation-induced upsets is triple modular redundancy (TMR). Voting circuits used in TMR implementations are decentralized and consensus is calculated from the redundant outputs off-chip. However if there is an insufficient number of pins available on the chip carrier, the TMR system must be reduced to an on-chip unprotected simplex system, meaning voters used at those locations become single points of failure. In this work, we propose a self-checking voting circuit for increased reliability consensus voting on FPGAs. Through fault injection and reliability analyses, we demonstrate that the proposed voter, which utilizes redundant voting copies, is approximately 26% more reliable than an unprotected simplex voter when reliability values of voters over normalized time are averaged.

한 글 요 약

고신뢰 FPGA를 위한 동의 보팅 기법

애프잘 우마
지도 교수: 이정아
컴퓨터공학과
대학원, 조선대학교

나노 스케일 FPGA 회로는 메모리 기반의 재구성 가능한 로직으로 구현되기 때문에 우주와 같은 극한환경에서 방사선에 의한 영향을 받기 쉽다. 임무필수 또는 안전필수가 요구되는 응용 프로그램의 경우는 적합한 결함감내를 위한 결함허용 기술이 일반적으로 사용된다. 방사선으로 인한 결함으로부터 FPGA를 보호하기 위해 가장 일반적으로 적용되는 기술이 회로의 삼중 중복화(TMR)이다. 삼중화된 회로의 출력을 결정하는 보팅 회로는 분산되어 있으며, 동의 결과는 오프 - 칩 (off-chip)의 중복 출력을 이용하여 계산된다. 그러나 칩에서 입출력으로 사용할 수 있는 핀 수가 충분하지 않은 경우, TMR 시스템은 칩 내부에서 결함허용이 고려되지 않은 단일 출력 시스템으로 변환되어야 하고, 이 때 사용되는 보팅 회로는 단일 실패 지점, 즉 단일 장애점(Single Point of Failure)이 될 수 있다. 본 논문에서는 FPGA로 구현된 동의 보팅 회로의 신뢰도를 높이기 위하여 보팅 회로의 복사본을 사용하는 자가검증 보팅 회로를 제안한다. 결함 주입 및 신뢰도 분석을 한 결과, 제안된 자가검증 동의 보팅 회로는 결함허용이 고려되지 않는 단일 보팅 회로에 비하여 약 26 % 더 신뢰할 수 있다.

I. INTRODUCTION

A. Motivation

1. Radiation Effects in FPGAs

Field-programmable gate arrays (FPGAs) are very popular in a variety of applications today because of their flexible general purpose nature. FPGAs are reconfigurable devices with a logic fabric available for configuration through a stream of digital bits. These devices were actually intended for facilitation of design prototyping, but as chips have grown dense in logic over the years, FPGAs are also being deployed in the field and given their field-programmability, they sometimes also prove an efficient alternative to application-specific integrated circuits (ASICs) when considering factors like *time-to-market* and lower *non-recurring engineering costs* since a designer is easily able to modify and debug the hardware design within a short amount of time. Thus they can combine the flexibility of being reconfigurable with the fast computation of the intended function by a dedicated circuit resulting in high performance.

The most relevant aspect of the FPGA architecture is the configuration memory which defines the functionality of all the logic elements and their interconnections present on the device. The configuration memory can be implemented with different technologies, such as flash, static RAM (SRAM) or antifuse. SRAM-based configuration memories support a feature called *partial reconfiguration*. It enables partial modification of the configuration bitstream at runtime and thus adds to the flexibility already offered by FPGAs.

The field programmability feature of the FPGAs enables the programming of the device remotely. The device can be reprogrammed after the system has been deployed for adding new functionalities to the existing design or for even uploading a completely different new design. This remote programmability combined with a flexible architecture makes FPGAs an attractive candidate for space-borne applications since such systems are very costly to transport them to the space. Popular examples of FPGAs being deployed in space-based systems include the Mars Rover [2] and NASA's James Webb Space Telescope (JWST) [3].

However, the re-programmability and flexibility offered by the FPGAs comes at a cost as FPGAs face unique dependability threats not known to traditional ASICs. The state of the configuration memory bits can be altered (flipped) if hit by an energetic particle (heavy ions and protons) which are abundant in harsh environments (e.g., in space) [4, 5]. While space conditions are certainly hostile, the extreme miniaturization of electronic circuits to achieve higher operating frequencies with lower power consumption [6] is also becoming a major concern for applications operating on ground as well [7]. These upsets can modify the user-defined functionality and cause erroneous computations. If the affected device is being used in a system classified as *critical*, the erroneous behavior of the device may result in loss of human lives, environmental damage or economic loss. Examples of such systems include switchgears for railways, biomedical applications, and control systems of an oil-field or the database of a bank.

Any upset in the configuration RAM (CRAM) induced due to radiation effects in taxonomically called a *single event effect* (SEE). When the upset affects the state of any configuration bit, it is known as a *single event upset* (SEU). If the upset causes a transient pulse in the combinatorial logic, such an upset event is called *single event transient* (SET). SETs are able to propagate through the circuit and if they get registered in a sequential logic element like the flip-flop, a soft-error is said to have occurred. Soft-errors are synonymous with functional errors and must be avoided for safety- or mission-critical applications. Therefore, such applications require effective *fault-tolerance* techniques to mitigate hardware faults in order to achieve the desired level of dependability.

2. Radiation Hardening by Design

These problems have led to the development of several radiation hardening techniques for FPGAs [8]. The most commonly used technique for mitigating the effects of radiation-induced upsets on FPGAs is *triple modular redundancy* (TMR) [9]. In a TMR, all three copies of the same circuit operate concurrently for achieving redundancy. It is highly recommended to couple TMR with periodic scrubbing of CRAM contents to create an effective error-tolerant design [10]. TMR can be applied to a design in different ways. Some applications may only require a partial application of TMR to critical portions of the design. Here, critical portions are those whose failure may lead to a system-wide failure. Others may require a full module TMR implementation to meet reliability requirements. Consensus or centralized voting is required when reducing a TMR system to a simplex (unprotected) system. Such voters create a single point of failure, unlike those used in decentralized voting.

Voting circuits are classified based on their insertion sites within a design [11]. For example, in a TMR system, errors can enter state machines via feedback paths. Therefore, to protect the system and synchronize all three modules, voters must cross all feedback paths. This is achieved by triplicating voters on all feedback paths [12]. Single partition TMR systems, such as that shown in Fig. 1, add voters at the primary outputs (POs) of the modules. They are good at masking errors as long as they can guarantee that only one copy will be affected at any given time before it is corrected through a recovery mechanism, such as scrubbing through a partial reconfiguration.

The voter in a TMR system calculates the consensus before the outputs can be used. This centralization of the system in the case of FPGAs is typically accomplished off-chip using components with higher reliabilities. However, in some cases, it may not be possible to reduce all or some of the POs off-chip because of an insufficient number of pins available on the chip carrier. In such a case, the designer may allow only those POs that are critical to the operation of the system to be made redundant. In another case, a designer may wish to apply a partial TMR to only the portions of the circuit that are critical to the operation of the system and reduce the triplex system to a simplex system. Such cases also require consensus voting on-chip. The primary focus of this thesis is the implementation of a dependable voting circuit for such cases that improves the overall reliability of a TMR system in a resource-efficient manner by leveraging the architecture of state-of-the-art FPGAs.

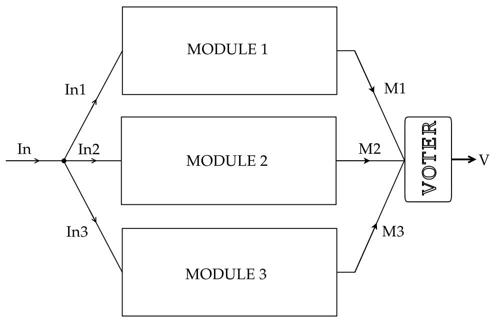

Figure 1: Single partition TMR with centralized voters only.

3. The Need for a Reliable Voter Circuit

We will now analyze the reliability of different system configurations with respect to the reliability of a single module. We will use the model shown in Fig. 1 for this reliability analysis. It is the simplest model of TMR with three modules and a single voter used for system reliability analysis. If we consider the three modules to be independent and identical, R_M to be the reliability of a single module, R_{TMR_V} to be the reliability of the TMR system (TMR_V), and R_V to be the reliability of the voter, we can calculate the overall TMR system reliability as [13]:

$$R_{TMR_V} = (3R_M^2 - 2R_M^3)R_V \tag{1}$$

If we assume a constant-failure rate λ during the useful life of the system, we can rewrite (1) as:

$$R_{TMR_V} = (3e^{-2\lambda t} - 2e^{-3\lambda t})R_V \tag{2}$$

Assuming that the voting circuit is made out of components identical to those in the redundant modules and that it has the same size as a module, we can assume that it also has the same constant-failure rate λ. (2) can then be rewritten as:

$$R_{TMR_V} = (3e^{-2\lambda t} - 2e^{-3\lambda t})e^{-\lambda t} \tag{3}$$

Let $R_{TMR_{ideal}}$ be the reliability of a TMR system (TMR_{ideal}) with an ideal voter. For an ideal voter ($R_V = 1$), we can rewrite (2) for TMR_{ideal} as:

$$R_{TMR_{ideal}} = (3e^{-2\lambda t} - 2e^{-3\lambda t}) \tag{4}$$

Figure 2 presents the reliability plots of the simplex, TMR_{ideal}, and TMR_V systems. Some interesting observations can be based on these plots. The reliability of TMR_{ideal} is equal to the reliability of the simplex system when $(R_{module} = 0.5)$. Typically, the individual module reliabilities are much higher than 0.5. Therefore, $R_{TMR_{ideal}}$ is typically greater than $R_{simplex}$. Additionally, it is clear from these plots that when the reliability of a non-ideal voter is quantified, R_{TMR_V} is always smaller than $R_{simplex}$ for all values of module reliability. Thus, our goal is to increase R_{TMR_V} such that it becomes greater than $R_{simplex}$ for some value of module reliability R_{module} because, otherwise, there is no benefit in applying a TMR.

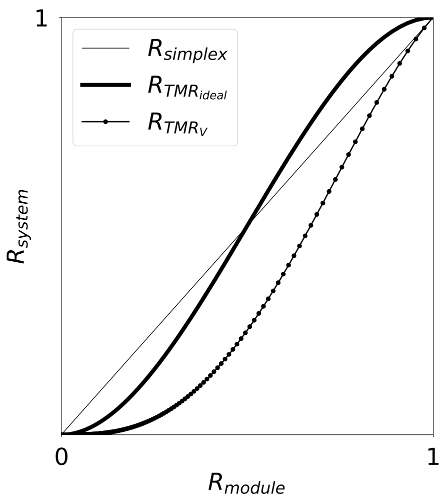

Figure 2: Reliability comparison of simplex, TMR_{ideal} and TMR_V systems.

B. Contributions

In this thesis, we present the first self-checking voting circuit for reliable on-chip consensus voting that improves the overall reliability of a TMR system in a resource-efficient manner by leveraging the architecture of state-of-the-art

FPGAs.

- We demonstrate how the proposed voter design achieves fault-tolerance by means of switching between redundant voter copies attached to the same output line.
- We show how the logic design of the voting circuit can be efficiently mapped onto the architecture of a modern FPGA.
- We show that our voter improves the TMR system reliability with minimum hardware overheads.

C. Thesis Layout

The thesis is organized as follows. In Chapter II, we present an overview of the general FPGA architecture. Then in chapter III, we describe the different types of voter circuit toplogies used in TMR systems. Next in Chapter IV, we discuss previous work related to the topic of fault-tolerant increased reliability voting in digital circuits. We then propose our voter design in Chapter V. We analyze it analytically, through fault injection simulations and reliability analyses. And finally, we conclude the thesis in Chapter VI.

II. FPGA ARCHITECTURE OVERVIEW

A. Configuration Memory

The configuration memory or CRAM is the most relevant aspect of the FPGA architecture. The highly flexibile nature of FPGAs is due to this programmable CRAM. The smallest unit of CRAM for addressing its contents is called a *frame*. The contents of CRAM define the behaviour of the logic elements and their interconnections. As mentioned previously in section I.A, CRAM can be implemented using different technologies such as flash, antifuse or SRAM. Flash-based FPGA devices consume comparatively lesser power and are also more tolerant against radiation effects. Antifuse-based FPGAs different from all other CRAM technologies can only be programmed once however. On the other hand, SRAM-based have the highest logic densities compared to any other FPGA technology. The configuration bitstream in the case of SRAM-based FPGAs is stored in a volatile memory and thus they need to be programmabed upon each start.

B. Look-up Table

Look-up table or LUT is the most basic logic element in FPGAs. A k-input LUT is actually implemented as a multiplexer with k select lines which select among 2^k bits. A k-input LUT can thus implement any k-input boolean function as shown in Fig. 3.

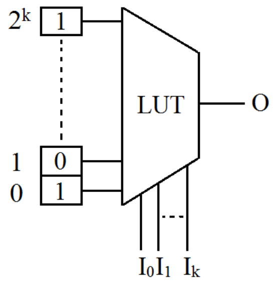

Figure 3: A *k*-input LUT.

C. Configurable Logic Block

Configurable Logic Blocks (CLBs) combine LUTs and other basic logic components like flip-flops in a small module which connects to a single switch matrix (Fig. 4). CLBs contain two slices and each slice further contains four 6-input LUTs and eight complementary flip-flops. Additionally, CLBs also contain some hardwired circuitry called carry-chains built from XOR gates and MUXes for fast implementation of adders and subtracters. The carry-chain structures extend from one CLB to another in the same column.

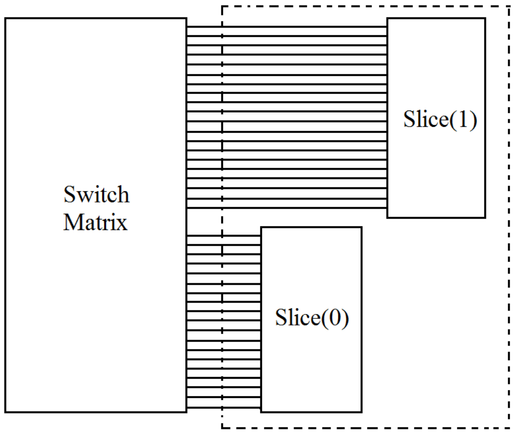

Figure 4: A CLB divides into two slices [1].

D. Digital Signal Processing

FPGAs contain dedicated computation blocks for digital signal processing (DSP) applications that would otherwise consume a large number of LUTs for simple operations. These DSP blocks are hardwired and each one of them can perform a fixed point multiplication. State-of-the-art FPGAs also contain some other dedicated units like digital clock managers for more specific functionality.

E. Block RAM

Registers such as counters can be efficiently implemented using the flip-flops contained in the CLBs. A LUT can also be configured as distributed RAM with read and write accesses. However, when a larger RAM is required as buffers or as instruction memories, the use of LUTs for implementing these very specific functionalities may well be considered a wastage of general purpose logic resources. Therefore, FPGAs include block RAMs (BRAMs) implemented with SRAM cells and addressable through a dedicated read/write circuitry to be specifically as general purpose memory.

III. VOTER TOPOLOGIES

Voting circuits in TMR systems can be classified based on their insertion points in the design. Some main types of voters topologies are described in this section.

A. Feedback Path Voters

In a full TMR system, triplicated voters cross all feedback paths to prevent any error from entering the state machines and to keep all three modules synchronized. This is illustrated in Fig. 5 with a simple 1-bit counter. On every rising clock edge, the counter loads the value opposite to that in its previous state, alternating its state between 1 and 0. In the absence of feedback path voters as shown in Fig. 5(a), an SEU affecting one of the counter registers can cause it to go out of sequence. The erroneous register remains out of sequence until the system is reset. With one failed register the system does not not fail but if another register is affected by an SEU, the system becomes permanently erroneous.

This problem can be solved by adding triplicated voters at the feedback paths as depicted in Fig. 5(b). These voters can mask a single error and keep all three registers synchronized. This is why triplicated voters at feedback paths are also called *synchronization* voters [11].

B. Partition Voters

Single partition TMR adds voters at the POs as shown in Fig. 1. It is able to mask errors as long as only one copy is affected at a given time before it is corrected by scrubbing. But if multiple copies are affected then the mask breaks and erroneous values are able to reach POs. By dividing the circuit into smaller partitions with each partition separated by decentralized triplicated voters as depicted in Fig. 6,

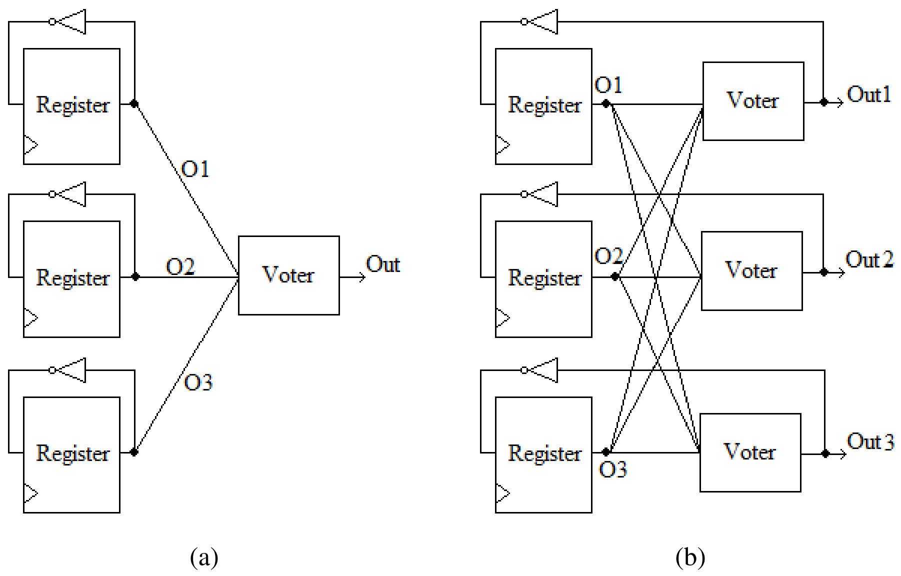

Figure 5: (a) Without voters at feedback paths. (b) Triplicated voters at feedback paths.

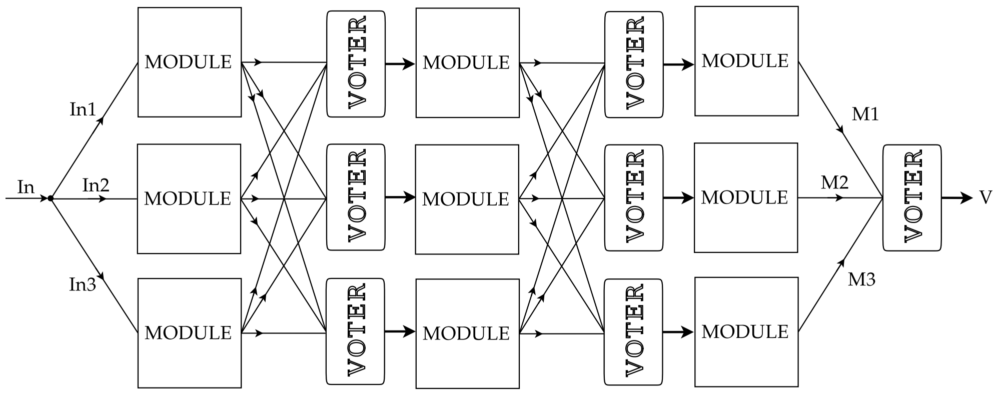

Figure 6: Decentralized voters in a multiple partition TMR.

the system can handle multiple concurrent non-overlapping errors.

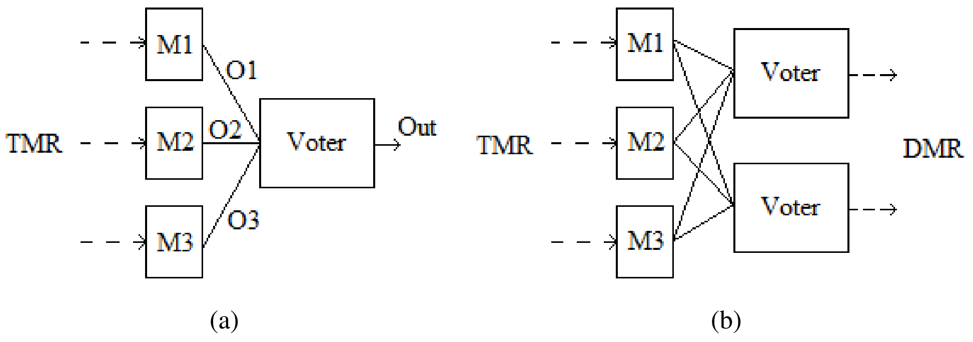

Figure 7: (a) TMR system reduction to a simplex system. (b) TMR system reduction to a Duplex system.

C. Reduction Voters

Reduction voters are present at the border between and a TMR and a non-TMR system. In a full TMR, primary inputs (PIs) and POs are triplicated and outputs are reduced to a single set via voting external to the FPGA. However, when there are an insufficient number of pins available on the package, it is desirable to reduce the circuit outputs to a single set on the FPGA. Voters that do this job are called reduction voters as shown in Fig. 7(a). There are also some cases where a TMR is partially applied to only critical parts or it may be that after applying TMR and covering all the feedback paths, the reliability requirements for the given application are met by reducing to a simplex system.

Reduction voters are also used to reduce a TMR system to a duplex system or dual modular redundancy (DMR) as shown in Fig. 7(b). A system is deployed in a DMR configuration for concurrent error detection which works by comparing the computations between the duplicate modules.

IV. RELIABLE VOTING IN DIGITAL CIRCUITS

A basic-logic gate representation of the typical majority bit-voting circuit used in TMR systems is shown in Fig. 8.

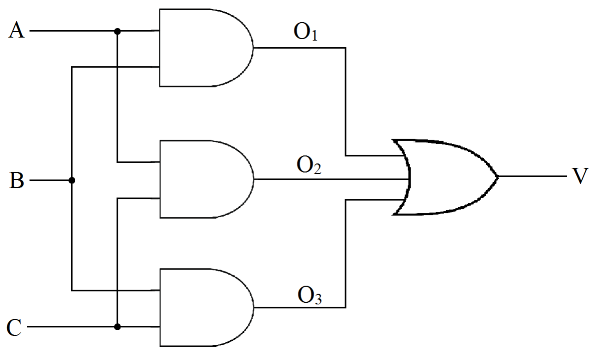

Figure 8: Conventional majority voting circuit used in TMR.

This voting circuit is vulnerable as a single fault in this circuit can travel as an error to the output wire causing an incorrect output value. For example, suppose the current inputs to the voter are (‘A,B,C’ = ‘0,0,0’). If there is a stuck-at-1 fault at either O_1, O_2 or O_3, the output at V will be ‘1’ instead of ‘0’. As shown in section I.A, the use of such an unreliable circuit actually degrades the overall reliability of a TMR system achieving even lower reliability than that of a single module. Therefore, in order to usefully apply a TMR, reliable fault-tolerant voter circuits are required. To this end, several researchers have proposed fault-tolerant voter designs for dependable consensus voting in digital circuits. In the following, we briefly review the most relevant fault-tolerant voting systems from the literature.

A. A Novel Fault-Tolerant Voter Circuit (NFTVC)

In [13], a fault-tolerant bit-voting circuit has been proposed for TMR implementations and it is shown to be six times more reliable and fault-tolerant compared to the conventional TMR voter circuit with lower area, power, and delay requirements. This voter circuit is shown in Fig. 9.

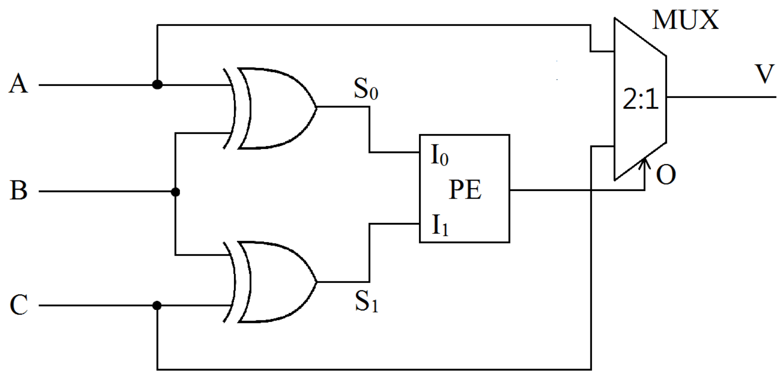

Figure 9: A novel fault-tolerant voter circuit.

If all the inputs to this circuit are correct, then irrespective of the faults in the voter circuit, the output will follow the input. Suppose the input vector to the voter circuit is (‘A,B,C’ = ‘0,0,0’). The correct output at V in this condition is ‘0’. If S_0, S_1 or O is stuck-at-1, the circuit will still produce the correct output. The priority encoder (PE) gives the highest priority among its inputs to the one which is at ‘0’. If I_0 is at ‘0’, then a ‘0’ is produced at O, otherwise if I_1 is at ‘0’, O is driven by a ‘1’. In this way, if it is assumed that S_0 is stuck-at-1, PE will produce ‘1’ at O due to prioritizing I_1. This will cause the multiplexer (MUX) to select the input C and the correct value ‘0’ will be obtained at the output line V. Other faults in this circuit can also be tolerated in the same way. The voter circuit only fails if any one of the four transistors corresponding to the output MUX is affected.

B. A Simple Fault-Tolerant Voter Circuit (SFTVC)

The reliability of NFTVC was surpassed in [14] by a simpler bit-voting circuit shown in Fig. 10 that has even lower area and power demands, as well as a smaller propagation delay.

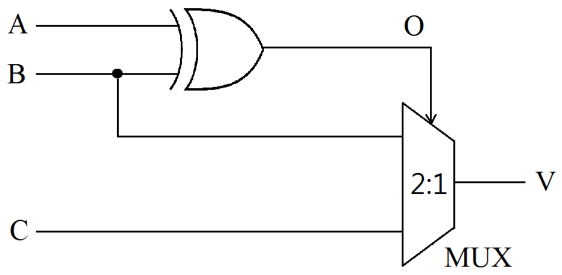

Figure 10: A simple fault-tolerant voter circuit.

It is obvious that the hardware overheads of SFTVC are lower than NFTVC due to a very simple logic design. The circuit consists of only two logic blocks, a single exclusive OR (EXOR) gate and a single MUX. There is also a single node S internal to the voter. SFTVC just like NFTVC can tolerate a single internal or external fault. Let us now look at the fault-tolerant operation of SFTVC under different fault conditions.

- S *is faulty*: Because of the single-fault model, V is correct since all the inputs to the circuit are correct (i.e., A = B = C).
- A *or* B *is faulty*: Because A and B are different, S is at logic-1 which selects the error-free input C at the voter output V.

- C *is faulty*: Because A and B are in agreement, S is at logic-0 which selects the error-free input B at the voter output V.

C. An Improved Fault-Tolerant Voter Circuit (IFTVC)

[15] proposed an improved fault-tolerant majority bit-voter (Fig. 11) that performed better than previous designs for a new figure of merit called FT-FOM, which was produced through a combination of evaluation of the circuit's fault-tolerance and design parameters.

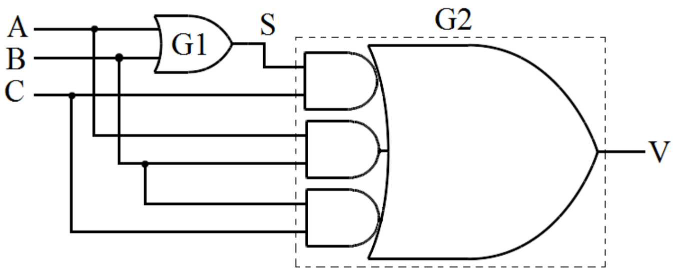

Figure 11: An improved fault-tolerant voter circuit.

This circuit also consists of two logic blocks similar to SFTVC. G1 is an OR gate while G2 is a complex gate and it implements the boolean function V = SZ + XY + YZ, where S = X + Y is the output at G1. It is shown through truth-cum-fault tables in [15] that the proposed IFTVC voter is more fault resilient to internal and external faults than the classical, NFTVC or SFTVC voter circuits. Specifically, there are only two cases where IFTVC is not able to mask errors. For the input vectors ('X,Y,Z' = '0,0,1') and ('X,Y,Z' = '1,0,1') with a simultaneous internal fault, the voter output at V will be incorrect. For all other fault scenarios, IFTVC is able to mask the corresponding errors.

D. Word Voters

Different from bit-voters, a word-voter design was proposed in [16], which was also shown to increase the data integrity of TMR designs with unique advantages over bit-by-bit voting schemes. Bit-voters are simple and fast while word-voters are more expensive to implement, but consider all bits in parallel to determine a final output, which increases the data integrity. In [17], the authors explored the design space to propose an adaptable bit-width voter that combined the advantages of bit-voters and word-voters.

V. PROPOSED SELF-CHECKING VOTER

The voter proposed in this thesis is a fault-tolerant bit-voter that specifically targets FPGA circuits and leverages the FPGA architecture for compact implementation. The design is a self-checking circuit running a feedback combinatorial loop that cannot be described in a hardware description language for inference by synthesis tools. Instead, it requires an instantiation of FPGA primitives for synthesis. The major difference between our voter and others is that it achieves fault tolerance by means of switching between redundant voter copies. Because of the way it is implemented, it is able to maintain its fault-tolerance (i.e., the ability to give an error free output in the presence of a fault) on FPGAs. Previous voters were proposed for digital circuits in general, meaning they are not able to maintain their intended fault-tolerance on FPGA architecture. A detailed explanation is provided in the following sections.

A. Logic Design

Figure 12 illustrates the logic design of the proposed self-checking voter. The voter achieves fault-tolerance by means of shifting between redundant voter copies. There are four copies of a majority voting block that are grouped into two sets: set-A and set-B. During operation, only one of the two sets feeds lines V1 and V2. Voter output is taken from line V1 in Fig. 12, but we can select either of the lines V1 or V2 as the output as both of them have identical signal assertions. Each copy from set-A is matched against a copy from set-B using two flags called *conflict flags*: CF1 and CF2. An additional logic unit called the *arbiter* monitors V1 and V2 to control the selection between the two sets of voters by driving CF1 and CF2 accordingly. If only, a single flag was used, then the voter would only be able to shift once to the redundant voter set. However, the use of two flags by the arbiter allows flip-flop shifting between the two voter sets.

In order to understand the operation of the circuit, let us assume that set-A is currently active and sourcing the lines V1 and V2 (i.e., the value of CF1 is set to '0' at this point to select set-A). Additionally, let us assume that the current inputs to the voter are ('M1,M2,M3' = '0,0,0') such that the voter produces logic-'0' on lines V1 and V2. Here M1, M2, and M3 are the outputs of the three TMR modules. When both V1 and V2 are '0' and CF1 is also '0' , mux-x outputs '1', which selects CF1 at mux-z to set a '0' value for CF2. Thus, the signal states at this point are ('V1,V2,CF1,CF2' = '0,0,0,0'). This condition is represented by state-1 in Fig. 14. Now, assume that an upset introduces a fault in one of the two copies belonging to set-A and that this fault presents as an error in the input vector ('M1,M2,M3' = '0,0,0'). If this error presents, the arbiter observes a difference between V1 and V2 such that ('V1,V2,CF1,CF2' = '1,0,0,0'). In this case, the arbiter complements CF1 to switch to set-B ('V1,V2,CF1,CF2' = '1,0,1,0'). This is the transition to state-2 in Fig. 14. Assuming that set-B is fault-free, lines V1 and V2 will agree with each other again, which is then observed by the arbiter as ('V1,V2,CF1,CF2' = '0,0,1,0'). The arbiter then changes the value of CF2 to match that of CF1, which is represented by the transition back to state-1 in Fig. 14 ('V1,V2,CF1,CF2' = '0,0,1,1'). At this point, the voter can still tolerate another fault that does not overlap with the existing fault (i.e for some other combination of 'M1,M2,M3'). Given a repair mechanism such as CRAM scrubbing is active to correct the existing fault, it is also possible to tolerate overlapping non-concurrent faults. Additional details on fault-tolerance will be provided in the fault injection analysis section.

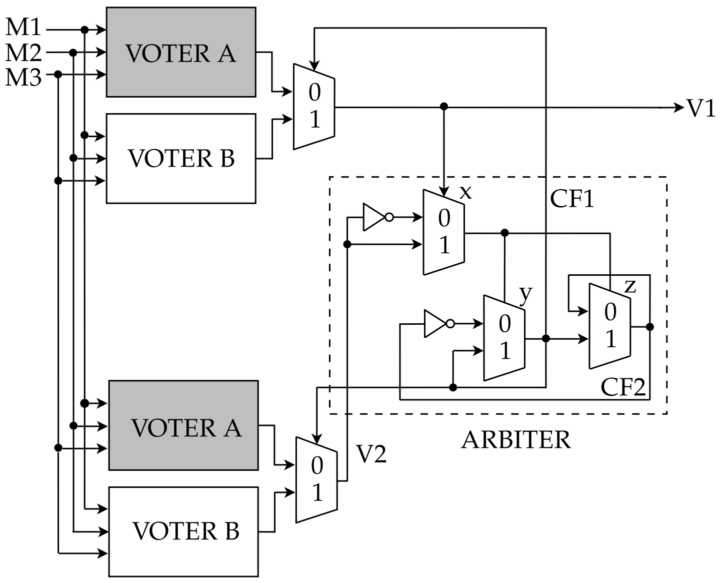

Figure 12: Logic design of the proposed voter.

Figure 14 shows that the design is running a combinatorial loop with no determined entry point into the loop. CF1 depends on the signal states of V1 and V2 for a decision, but the signal state of CF1 is required to determine which of the two voter sets will feed V1 and V2. The signal state of CF1 is also required for determining the signal state of CF2. This condition is known as a *deadlock*. However, when implemented on hardware, the voting circuit can escape this condition. This stems from the fact that these wires remain in metastable signal states for an undetermined length of time until random thermal or induced noise breaks the tie and makes the circuit converge to its operational state. This time duration is very small, but non-deterministic. However, once the circuit enters an operational state, it never exits the combinatorial loop shown in Fig. 14. The initial state in which the voter starts is also non-deterministic, but it is clear from Fig. 14 that as long as $(V1 = V2)$, the voter will eventually land on state-

1. Thereafter, in the event of an error, it passes through the state transitions explained above.

B. Hardware Implementation

Modern FPGAs support dual output look-up tables (LUTs). A 6-input LUT (6-LUT) can be configured to act as two 5-LUTs sharing the same inputs, but with two separate output lines, as shown in Fig. 13(a). O6 is the only output line used in the case when the LUT is configured to be used as a 6-LUT while O5 is used as the second output line when dual outputs are configured. To use a 6-LUT as a dual output 5-LUT, the sixth input line I5 must be set as '1'.

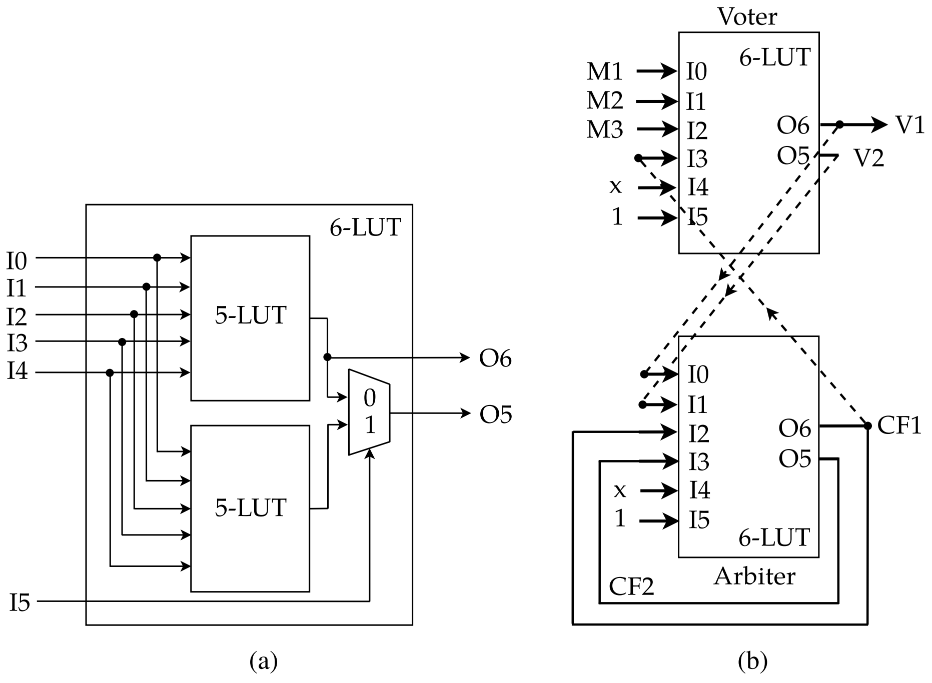

Figure 13: (a) A 6-LUT configured as a dual output 5-LUT. (b) Implementation of the proposed voter with two dual output 5-LUTs.

The proposed voter can be efficiently implemented with just two LUTs using the dual output feature of LUTs on the Virtex FPGAs from Xilinx [18], as shown in Fig. 13(b). One of the LUTs houses all of the voter redundancies while the arbiter logic is implemented by another LUT. Both LUTs are configured as dual output LUTs. In the voter-LUT, both 5-LUTs share the same inputs, but one of them implements the group (VoterA,VoterB) from Fig. 12 while the other implements the other group (VoterA,VoterB). Additionally, in the arbiter-LUT, one 5-LUT implements the logic that drives CF1 while the other 5-LUT implements the logic that drives CF2. This design cannot be inferred by synthesis tools and requires the instantiation of LUT primitives with the INIT attribute (**64'h**e8e8e8e8e8e8e8e8) for the voter-LUT and (**64'h**90f690f6f690f690) for the arbiter-LUT. INIT is the Xilinx terminology that refers to the contents of an LUT. The value of this attribute defines the functionality of an LUT primitive.

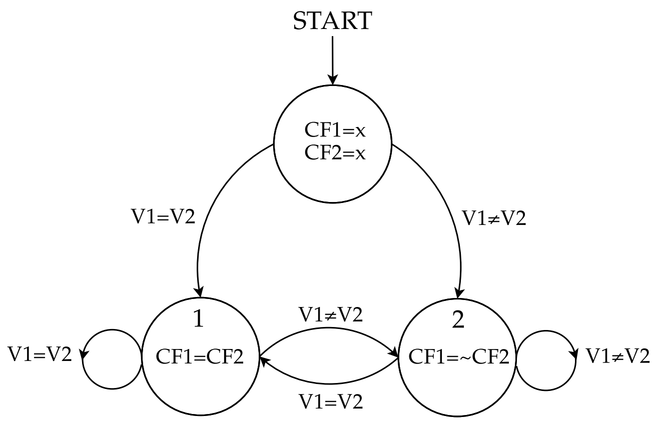

Figure 14: State diagram for operation of the voting circuit.

C. Fault Injection Analysis

The proposed design can tolerate any single fault in the CRAM bits related to the voter and arbiter LUTs. It can also handle multiple faults that lead to concurrent non-overlapping errors. For example, it can tolerate a fault in one of the copies of set-A that presents as an error for (‘M1,M2,M3’ = ‘0,0,0’) and another fault in set-B that presents as an error for (‘M1,M2,M3’ = ‘1,1,1’). Any number of faults in the arbiter-LUT will not affect the voter output because the voter-LUT that drives the V1 output line will be unaffected. The only effect a fault in the arbiter-LUT can have on the voter-LUT is to switch the sets unnecessarily. The voter was simulated for all such faults using the Xilinx ISIM tool. Faults were injected by modifying the INIT attribute of the LUTs. In order to allow the circuit to enter the combinatorial loop described in section V-A, the effect of random noise to break out of the deadlock condition in hardware was simulated by forcing the line CF1 to be ‘0’ initially, and then releasing it to allow normal voter operation.

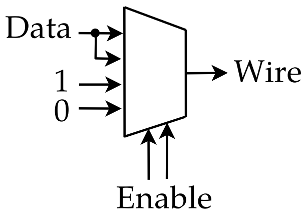

Figure 15: Fault injection mechanism for injecting faults in routing nets.

Faults affecting the internal nets were also analyzed. Upsets in the configuration bits that control the FPGA’s routing resources are responsible for these faults. Table 1 contains the results for faults injected into the routing nets for the input vector (‘M1,M2,M3’ = ‘0,0,0’). It can be seen that even when multiple faults affect the internal nets concurrently, the voter output remains unaffected.

Table 1: Truth-cum-fault enumeration table of the proposed voting circuit for faults affecting the internal nets.

Voter PIs			**Internal Nets**						**Voter PO**
M0	*M1*	*M2*	*CF1* *(O6→I3)*	*CF1* *(O6→I2)*	*CF2* *(O5→I3)*	*V1* *(O6→I0)*	*V2* *(O5→I1)*	V1(O6)	
0	0	0	x	x	x	x	x	0	Correct
0	0	0	x	x	x	x	x	0→1	Error

Again, the only effect that these faults can have is that voter copies may be unnecessarily switched or the circuit's ability to switch the voter copies may be lost. An error in the primary voter output can only be produced by a fault in net V1(O6), which directly feeds the output line and this fault can be either an SEU affecting the bits corresponding to its switch box or a single event transient (SET). The same is true for any combination of (M1,M2,M3) which are the PIs to the voter. Any SET that can affect the combinatorial logic paths corresponding to the current input vector would be again limited to either switching the voter copies or travelling to the output as a glitch. Faults in the routing nets were injected using the setup shown in Fig. 15. To inject a fault, the net is interrupted by the multiplexer (or mux) such that the net drives the *Data* inputs while the *Wire* output of the mux drives the signal that was meant to be driven by the net. The selection of the type of fault to be injected was performed using the *Enable* inputs to the mux.

D. Simulation

We simulated the post-place & route (PAR) implementation of the voting circuit to demonstrate its recovery from a fault. The timing simulation waveforms are shown in Fig. 16 and were produced using the Xilinx ISIM tool. A fault was injected into the copy belonging to set-A for the vector ('M1,M2,M3' = '0,0,0') by modifying the INIT attribute of the voter-LUT. The events in the waveform diagrams in Fig. 16 are explained below:

1. V1 and V2 disagree with each other when the fault in set-A is expressed as an error for ('M1,M2,M3' = '0,0,0').
2. The arbiter complements the value of CF1.
3. Correct value of V1 is restored after shifting to the other voter set and V1 and V2 are now in agreement with each other again.
4. The arbiter alters the value of CF2 to achieve agreement between the two flags and the system successfully switches to the other voter set.

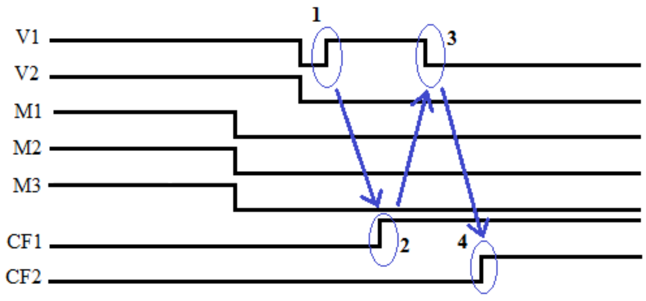

Figure 16: Timing simulation waveforms of the proposed voter.

E. Reliability Calculations

The use of redundancies improves the reliability of the proposed voter compared to a simplex voter. However, the switching functionality of the arbiter can be affected by an upset. For example, if an upset affects net CF1(O6→I3), the arbiter loses its ability to shift the voter set. If another upset affects the voter-LUT before the previous upset is fixed, the system malfunctions.

We will now analyze the reliability of the voting circuit when a repair process (e.g., scrubbing) is absent. Suppose the probability that the arbiter successfully operates to switch the voter redundancies is p. Then, the probability that the arbiter fails to operate successfully is $(1-p)$. It follows that for a failure rate λ_1, the probability that the voter is affected and the switching process works properly is $\lambda_1 p$ and the probability that the voter is affected but the switching process does not work properly is $(1-p)\lambda_1$. It is important to note that although it is possible that a successful switch may not occur, the probability of this event actually occurring is very small. The reason for this is that, as the Rosetta experiment [19] demonstrates, a single upset event in most cases only changes (flips) the state of a single CRAM bit. Multiple cell upsets (MCUs) where a single charged particle affects more than one physically neighbouring sensitive nodes of the SRAM cells are also possible but less probable. Multiple bit upsets on the other hand where multiple upsets occur in a single word are almost never caused by a single charged particle. Due to bit interleaving and address scrambling, MCUs usually appear in different words of the memory and thus, error correction codes can correct this condition most of the times [20, 21]. Additionally, the number of CRAM bits related to the voting circuit compared to the total number of CRAM bits in the FPGA are negligible. Therefore, the probability of a second upset affecting the voting logic or arbiter logic after one of them is already affected is very low. Furthermore, if scrubbing is active, then the time required to complete a successful repair activity is on the order of milliseconds and can even be reduced to microseconds [22], which is very small compared to the mean time between two upsets for FPGA devices [23].

We can derive the reliability of the voting circuit using the state transition diagram presented in Fig. 17. State-1 and state-2 are the operational states of the voter and state-3 represents system failure. State-2 represents the state where the voter is already upset and a successful switch has taken place. The transition rate from state-2 to state-3 is represented by λ_2. The reliability of the voter in the state transition diagram is then computed assuming $(\lambda_1 = \lambda_2)$ and is given by (5).

$$R_V(t) = (1 + p\lambda t)e^{-\lambda t} \tag{5}$$

The reliability curves for different values of p against normalized time (λt) are presented in Fig. 18(a). As p decreases, the reliability of the voter decreases. When p reaches zero, the reliability of the proposed voter reduces to the reliability of a simplex voter $(e^{-\lambda t})$. By averaging the reliability values in Fig. 18(a) for p=1 (perfect switching case) vs p=0 (unprotected simplex voter) we roughly obtain a 26% improvement in reliability. Now, R_V can be substituted from (5) into (3) to get:

$$R_{TMR_V} = (3e^{-2\lambda t} - 2e^{-3\lambda t})(1 + p\lambda t)e^{-\lambda t} \tag{6}$$

The plot of R_{TMR_V} from Fig. 2 can now be modified according to (6). The result is shown in Fig. 18(b) for $(p = 1)$. It can be seen in Fig. 18(b) that the gap between $R_{TMR_{ideal}}$ and R_{TMR_V} can be minimized by increasing the reliability of the voter R_V. As p decreases, the curve becomes closer to the one shown in Fig. 2 and exactly matches it at $(p = 0)$. Furthermore, the reliability R_{TMR_V} equals R_{module} at 0.62 and R_{TMR_V} becomes greater than R_{module} for values of R_{module} beyond 0.62. Therefore, we have successfully met the goal of a TMR system because with the proposed voter, a TMR system becomes more reliable to use than a simplex system if we operate in the region of the curve beyond ($R_{module} = 0.62$), provided we have a guaranteed switch between the voter sets.

$$\begin{tikzcd}
1 \arrow[r, "p\lambda_1"] \arrow[rr, bend left, "(1-p)\lambda_1"] & 2 \arrow[r, "\lambda_2"] & 3
\end{tikzcd}$$

Figure 17: State transition diagram for reliability analysis of the proposed voter.

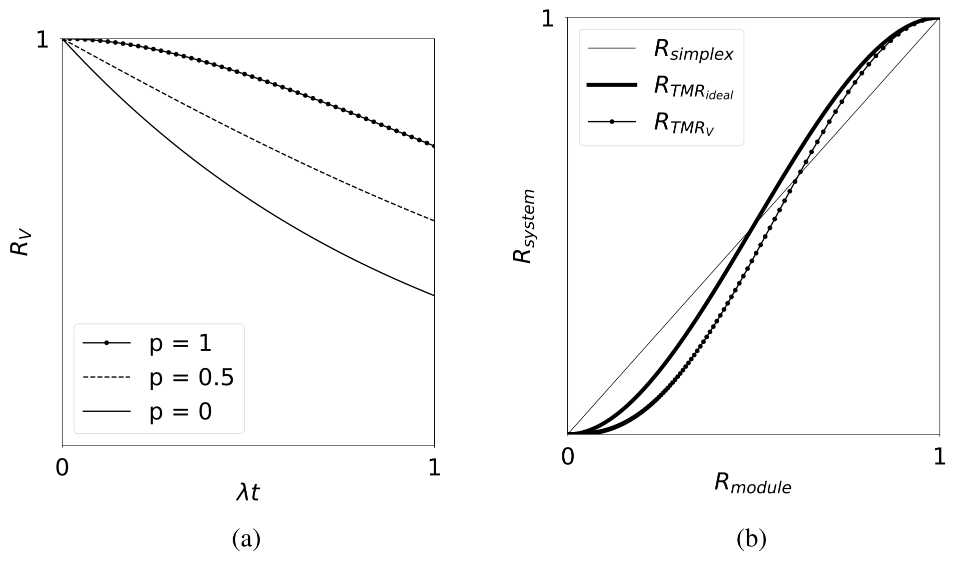

Figure 18: (a) Reliability curves of the voter circuit for different values of *p*. (b) Reliability comparison of simplex, TMR_{ideal} and TMR_V (proposed voter) systems.

F. Design Entry: Full Adder

Three different implementations of a full adder circuit were compared in terms of area, power and delay on a XC5VLX110T Virtex-5 FPGA: a simple full adder circuit, using TMR with a simple voting circuit, and using TMR with the proposed voting circuit. The circuits were synthesized using the Xilinx Synthesizer Tool (XST). All results were obtained for Post-PAR implementations. XPower Analyzer was used to measure power and Timing Analyzer was used to measure circuit delays.

Table 2: Comparison of the three techniques in terms of area, power and delay for full adder circuit.

Fault-tolerance Technique	**Levels of Logic**	**Area (LUTs)**	**Power (mW)**	**Delay (ns)**
None	3	1	19	6.293
TMR (Simplex Voter)	4	4	20	7.232
TMR (Proposed Voter)	4	7	20	9.161

Table 2 contains the results for these implementations of a full adder circuit. There is a proportional increase in area for TMR (proposed voter) because our voter uses two LUTs instead of one. The power dissipation results show no increase in comparison to the TMR (simplex voter). In terms of performance, the delay increased by 1.929 *ns* for TMR (proposed voter) compared to the TMR (simplex voter). This increase in delay may seem large, but it is actually only large when the 2-bit full adder is considered because its circuit is very small in size. For example, when the priority encoder circuit *priority* which consumes more than 600 LUTs in a TMR configuration from the EPFL benchmarks suite is considered, the delay only increases from 33.692 *ns* (TMR with simplex voter) to 34.631 *ns* (TMR with proposed voter) which is now only a 2.7% increase in delay. Therefore as the size of the circuit grows, the effect of increase in delay is also reduced.

VI. CONCLUSION

In this thesis, a fault-tolerant voting circuit for reliable centralized voting in FPGA circuits was presented.

- It increases the reliability of a consensus voter in a TMR configuration.
- It has a unique design. It uses multiple copies of the majority voting circuit in two sets and shifts between the sets using additional logic such that flip-flop shifting is possible.
- The proposed design specifically targets FPGAs and leverages their architecture for compact implementation and the retention of fault-tolerance.
- Improvements in fault-tolerance were demonstrated through fault injection analysis and improvements in reliability were formally proved to be guaranteed.

PUBLICATIONS

A. Journals

1. U. Afzaal and J.A. Lee, "A Self-Checking TMR Voter for Increased Reliability Consensus Voting in FPGAs," *IEEE Transactions on Nuclear Science*, vol. 65, no. 5, pp. 1133-1139, May 2018.

B. Conferences

1. U. Afzaal and J.A. Lee, "FPGA-based Design of a Self-Checking TMR Voter," in *Field-Programmable Logic and Applications, 2017. FPL'17.* 27th *International Conference on.* IEEE, 2017.

2. U. Afzaal and J.A. Lee, "On-Demand Selective Region Scrubbing in FPGAs with an LUT-grain Fault Masking and Detection," in *European Network on High Performance and Embedded Architecture and Compilation, 2017. HiPEAC'17.* 12th *International Conference on.* 2017.

REFERENCES

[1] Xilinx, "Virtex-5 FPGA User Guide," *User guide, UG190, V5.4, March*, vol. 5, 2012.

[2] D. Ratter, "FPGAs on Mars," *Xcell J*, vol. 50, pp. 8–11, 2004.

[3] A. Johns, B. Seaton, J. Gal-Edd, R. Jones, C. Fatig, and F. Wasiak, "James Webb Space Telescope: L2 Communications for Science Data Processing," in *Observatory Operations: Strategies, Processes, and Systems II*, vol. 7016. International Society for Optics and Photonics, 2008, p. 70161D.

[4] H. Quinn, "Radiation Effects in Reconfigurable FPGAs," *Semiconductor Science and Technology*, vol. 32, no. 4, p. 044001, 2017. [Online]. Available: http://stacks.iop.org/0268-1242/32/i=4/a=044001

[5] J. Gaisler, "Suitability of Reprogrammable FPGAs in Space Aspplications," ESA Technical Report, 2002 http://www. estec. esa. nl/microelectronics/asic/fpga_002_01-0-4. pd f accessed on 5th March, Tech. Rep., 2004.

[6] J. Hussein, M. Klein, and M. Hart, "Lowering Power at 28 nm with Xilinx 7 Series Devices," *Xilinx, White paper, WP389 (v1. 2)*, 2013.

[7] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*, 1st ed. Springer US, Boston, MA, 2010.

[8] F. Siegle, "Fault Detection, Isolation and Recovery Schemes for Spaceborne Reconfigurable FPGA-Based Systems," Ph.D. dissertation, Department of Engineering, University of Leicester, 2015.

[9] J. F. T. Olano, “Exploring the Use of Multiple Modular Redundancies for Masking Accumulated Faults in SRAM-based FPGAs,” Ph.D. dissertation, Institute of Information, Universidade Federal do Rio Grande do Sul (UFRGS), 2014.

[10] C. Bolchini, A. Miele, and M. D. Santambrogio, “TMR and Partial Dynamic Reconfiguration to Mitigate SEU faults in FPGAs,” in *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT’07. 22nd IEEE International Symposium ons*. IEEE, 2007, pp. 87–95.

[11] J. M. Johnson and M. J. Wirthlin, “Voter Insertion Algorithms for FPGA Designs Using Triple Modular Redundancy,” in *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA ’10. New York, NY, USA: ACM, 2010, pp. 249–258. [Online]. Available: http://doi.acm.org/10.1145/1723112.1723154

[12] K. Chapman, “Triple Modular Redundancy Design Techniques for Virtex FPGAs,” *Xilinx, XAPP197, V1.0.1, July*, 2006.

[13] R. Kshirsagar and R. Patrikar, “Design of a novel fault-tolerant voter circuit for TMR implementation to improve reliability in digital circuits,” *Microelectronics Reliability*, vol. 49, pp. 1573–1577, 2009.

[14] T. Ban and L. A. de Barros Naviner, “A Simple Fault-tolerant Digital Voter Circuit in TMR Nanoarchitectures,” in *Proceedings of the 8th IEEE International NEWCAS Conference 2010*, June 2010, pp. 269–272.

[15] P. K. Balasubramanian P., “A Fault Tolerance Improved Majority Voter for TMR System Architectures,” *WSEAS Transactions on Circuits and Systems*, vol. 15, pp. 108–122, 2016.

[16] S. Mitra and E. J. McCluskey, “Word-voter: A New Voter Design for Triple Modular Redundant Systems,” in *Proceedings 18th IEEE VLSI Test Symposium*, 2000, pp. 465–470.

[17] T. B. Ló, F. L. Kastensmidt, and A. C. S. Beck, “Towards an Adaptable Bit-width NMR Voter for Multiple Error Masking,” in *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct 2014, pp. 258–263.

[18] Xilinx, “UltraScale Architecture Configurable Logic Block,” *User guide, UG574, V1.5, February*, 2017.

[19] A. Lesea, S. Drimer, J. J. Fabula, C. Carmichael, and P. Alfke, “The rosetta experiment: atmospheric soft error rate testing in differing technology fpgas,” *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 317–328, Sept 2005.

[20] M. Wirthlin, D. Lee, G. Swift, and H. Quinn, “A Method and Case Study on Identifying Physically Adjacent Multiple-Cell Upsets Using 28-nm, Interleaved and SECDED-Protected Arrays,” *IEEE Transactions on Nuclear Science*, vol. 61, no. 6, pp. 3080–3087, 2014.

[21] H. Quinn, P. Graham, J. Krone, M. Caffrey, and S. Rezgui, “Radiation-Induced Multi-Bit Upsets in SRAM-Based FPGAs,” *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2455–2461, 2005.

[22] G. L. Nazar, L. P. Santos, and L. Carro, “Scrubbing Unit Repositioning for Fast Error Repair in FPGAs,” in *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, ser. CASES ’13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 2:1–2:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2555729.2555731

[23] Xilinx, “Device Reliability Report,” *User guide, UG116, V10.6.1, July*, 2017.

ACKNOWLEDGEMENTS

All praise is due to the Lord of the Worlds alone. This thesis made possible, my heartfelt appreciation for the constant reassurance and prayer of my parents who words cannot describe the level of patience with which they persevere my absence. The exceptional guidance and support from Professor Lee, my supervisor during the course of this degree. In sincere acknowledgement, the outstanding companionship of the Muslim community, especially the delicious desi food. And in gratefulness, my labmates: Sami Hassan, Tooba Arifeen and Inayat Ullah. Thank you every one.