RISC-V BASED TRIPLE MODULAR REDUNDANT CPU DESIGN FOR SPACE APPLICATIONS

by
EMİR CAN YAMAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of Master of Science

Sabancı University
December 2022

ABSTRACT

RISC-V BASED TRIPLE MODULAR REDUNDANT CPU DESIGN FOR SPACE APPLICATIONS

EMİR CAN YAMAN

Electronics Engineering, M.S. Thesis, December 2022

Thesis Supervisor: Prof. Yusuf Leblebici

Keywords: TMR, RISC-V, Spacecraft, SEU, Processor, Word voter

Fault-tolerant processors are essential for spacecraft because of the high-radiation environment of space. While the required reliability can be achieved by using specialized radiation-hardened processors, the cost is prohibitive. With the advent of CubeSats, low-cost spacecraft started to increasingly rely on non-radiation-hardened commercial-off-the-shelf components. Triple-modular redundancy can be applied to existing processor designs to increase reliability without using costly radiation-hardened ASIC processes or FPGAs. In this thesis, we demonstrate a conceptual fault-tolerant RISC-V processor by applying coarse-grain TMR to the open-source PicoRV32 core. To implement coarse-grain TMR, we attached word voters to the memory bus without modifying the internal structure of the processors. Using word voters increases error detection capability by revealing multi-module errors which can be masked by conventional bit-by-bit voters. Moreover, we propose a TMR controller module that can relay fault conditions to software and reset the CPUs. The module can help software to facilitate fault recovery procedures. Finally, we compare the synthesis results of our demonstration system with similar applications that use finer-grain TMR implementation. Our preliminary experiments show that the proposed coarse-grain TMR architecture can be used to protect ready-made IP cores with less development effort, which can be useful for low-cost space applications.

ÖZET

UZAY UYGULAMALARI İÇIN RISC-V TABANLI ÜÇLÜ MODÜLER YEDEKLİ CPU TASARIMI

EMİR CAN YAMAN

Elektronik Mühendisliği Yüksek Lisans Tezi, Aralık 2022

Tez Danışmanı: Prof. Dr. Yusuf Leblebici

Anahtar Kelimeler: TMR, RISC-V, Uzay aracı, SEU, İşlemci, Word oylayıcı

Uzaydaki yüksek radyasyon seviyesi nedeniyle arızaya dayanıklı işlemciler uzay araçları için elzemdir. Gerekli güvenilirlik seviyesi radyasyona dayanıklı özel işlemciler tarafından sağlanabilir, ancak bu işlemcilerin maliyeti yüksektir. Küp uyduların yaygınlaşması ile birlikte, düşük maliyetli uzay araçlarında ticari kullanıma hazır komponentlerin kullanımı artmıştır. Hazır işlemci tasarımları, üçlü modüler yedekleme sayesinde radyasyona dayanıklı ASIC prosesleri ve FPGA'ler gibi maliyetli unsurlar kullanılmadan daha güvenilir hale getirilebilir. Bu tezde, açık kaynaklı PicoRV32 çekirdeğine kaba taneli TMR uygulayarak hataya dayanıklı konsept bir RISC-V işlemci tasarımı sunulmaktadır. Kaba taneli TMR uygulamasını gerçekleştirmek için, word oylayıcılar işlemcinin iç yapısında değişiklik yapılmadan bellek anayoluna bağlanmıştır. Geleneksel bit-by-bit oylayıcılar yernine word oylayıcıların kullanılması, birden çok modülde aynı anda gerçekleşebilecek hataları ortaya çıkartarak hata tanılama kapasitesini arttırmaktadır. Bunların yanında, yazılımın oluşan hatalardan haberdar olmasını sağlayan ve işlemcileri yeniden başlatabilen bir TMR kontrol modülü tasarlanmıştır. Bu modül yazılımın hata kurtarma prosedürlerini başlatmasına yardımcı olabilir. Son olarak sistemin sentez çıktıları, ince taneli TMR kullanan benzer uygulamalarla karşılaştırılmıştır. Ön sonuçlar, tasarlanan kaba taneli TMR mimarisinin, hazır IP bloklarının daha düşük geliştirme eforu sarf edilerek korunmasında kullanılabileceğini göstermektedir. Bu tasarımlar düşük bütçeli uzay uygulamaları için faydalı olabilir.

ACKNOWLEDGEMENTS

I want to thank my advisors Dr. Yusuf Leblebici and Dr. Erdinç Öztürk for their academic guidance, and my family and friends for supporting me throughout my work.

I also want to especially thank Gizem Kerem who has always been with me and shared all my feelings.

To my family & friends

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ALU:	Arithmetic Logic Unit
AMBA:	Advanced Microcontroller Bus Architecture
ASIC:	Application-Specific Integrated Circuit
AXI:	Advanced eXtensible Interface
CISC:	Complex Instruction Set Computer
COTS:	Commercial Off-The-Shelf
CPU:	Central Processing Unit
DRAM:	Dynamic Random Access Memory
DUT:	Device Under Test
ECC:	Error Correction Code
ESA:	European Space Agency
FPGA:	Field Programmable Gate Array
FRAM:	Ferroelectric Random Access Memory
GPL:	General Public License
HDL:	Hardware Description Language
IP:	Intellectual Property
ISA:	Instruction Set Architecture
LVDC:	Launch Vehicle Digital Computer
MIPS:	Million Instructions per Second
MRAM:	Magnetoresistive Random Access Memory
PCPI:	Pico Co-Processor Interface
PDK:	Process Design Kit
RAM:	Random Access Memory
RISC:	Reduced Instruction Set Computer
RTL:	Register-Transfer Level
SEE:	Single-Event Effect
SEL:	Single-Effect Latchup
SET:	Single-Event Transient
SEU:	Single-Event Upset
SOI:	Silicon-on-Insulator

SOS:	Silicon-on-Sapphire
SPARC:	Scalable Processor ARChitecture
SRAM:	Static Random Access Memory
TID:	Total Ionizing Dose
TMR:	Triple-Modular Redundancy

1. Introduction

Fault-tolerant systems are designed to continue operating correctly if some part of the system becomes faulty. Fault-tolerant systems are especially important for "mission-critical" and "life-critical" devices such as avionics and medical devices. They are also crucial for devices that are designed to operate in harsh environments where error detection and mitigation are imperative to keep the device in working condition during the mission lifetime. The space environment is one of the harsh environments where fault-tolerant electronics are a must because of the high doses of radiation and energetic particles.

Triple Modular Redundancy (TMR) is one of the commonly used methods to protect logic circuits from errors. In TMR, circuit blocks are triplicated and their results are compared against each other. In case of a difference, the outlier block can be ignored and the output remains error-free. This principle can be applied to logic systems at different granularity levels.

In this thesis, a concept study of implementing TMR to an open-source RISC-V CPU is showcased. Without modifying the internal design, TMR logic is implemented as a wrapper around 3 identical processor cores. The advantages and disadvantages of implementing TMR at this granularity level are investigated. Different concepts for error mitigation and recovery besides the standard TMR approach are also discussed. In addition, the feasibility of using the implemented system as a co-processor in spacecraft systems is investigated.

1.1 Reliable Computing in Space Applications

Beginning in the 1960s, early digital computers started to be used in the control and data handling systems of spacecraft. The first digital computer to be used

in space was the Gemini Guidance Computer (Tomayko, 1988). As reliability has the utmost importance in spacecraft systems, especially manned ones, spacecraft computers are started to evolve in that direction. Since the early digital computers are simple and less advanced, they are less affected by the radiation environment of space. Still, some of the critical computer systems such as the Saturn Launch Vehicle Digital Computer used triple modular redundancy for guaranteed reliability (Tomayko, 1988).

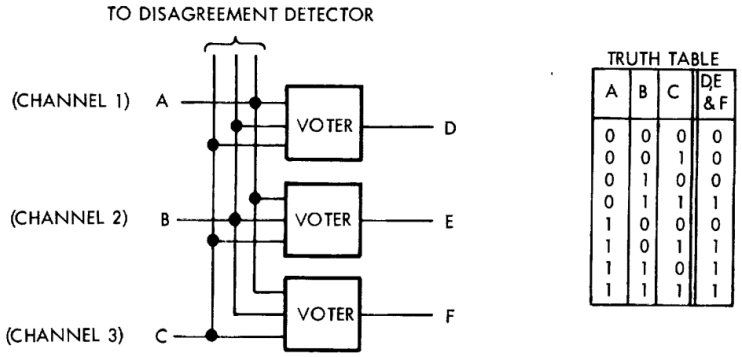


Figure 1.1 Saturn Launch Vehicle Digital Computer was one of the first digital spacecraft computers that use TMR. Voter diagram taken from the original Saturn LVDC manual (International Business Machines Corporation, 1964).

As spacecraft computers become more advanced and the technology nodes for integrated circuits started to get smaller, their radiation sensitivity increased. Many of the high-budget government missions started to use specialized radiation-hardened processors. These processors usually use special processes and packaging technologies such as silicon-on-sapphire (SOS). While radiation-hardened processors are certified and guaranteed to work reliably in high flux environments such as deep space, they are usually several generations behind their modern counterparts and they cost orders of magnitude higher than the equivalent non-hardened ones. For example, a radiation-hardened processor RAD750 from BAE Systems, first launched in 2005 can clock up to 200 MHz and performs around 400 MIPS (BAE Systems, 2008). A consumer processor from the era, Intel Pentium M 740 can clock up to 1.7 GHz and performs approximately 7400 MIPS (Sun, 2006).

Several projects existed during the 2000s, aiming to build a new generation of space-capable microprocessors, such as ESA's Next Generation Multipurpose Microprocessor project. The project resulted in the LEON series of processors based on Sun Microsystems' open-source SPARC architecture and capable of various configurations (Andersson, Gaisler & Weigand, 2010). Today, Cobham Gaisler continues to develop LEON processors in addition to the NOEL-V project which is based on RISC-V architecture and uses the same fault tolerance measures as LEON. LEON

microprocessors are usually built using radiation-hardened processes or radiation-tolerant FPGAs (Cobham Gaisler AB, 2021).

Besides the process-based measures, radiation-resistant versions of the LEON and NOEL-V processors use triple modular redundant flip-flops and ECC-protected registers (Andersson et al., 2010). Non-fault-tolerant versions of the LEON and NOEL-V processors are also available open-source with a GPL license. As of today, many successful spacecraft have used LEON microprocessors (European Space Agency, 2013).

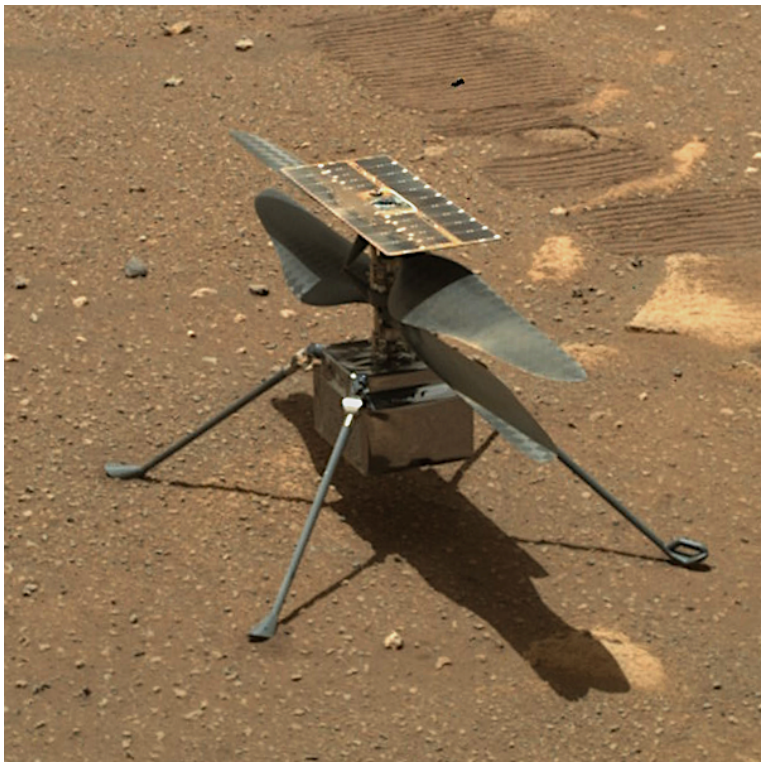


Figure 1.2 Martian helicopter Ingenuity mainly uses COTS parts. Such as a Qualcomm Snapdragon 801 mobile phone SoC for image processing (Tzanetos, Aung, Balaram, Grip, Karras, Canham, Kubiak, Anderson, Merewether, Starch, Pauken, Cappucci, Chase, Golombek, Toupet, Smart, Dawson, Ramirez, Lam, Stern, Chahat, Ravich, Hogg, Pipenberg, Keennon & Williford, 2022).

In the last decade, decreasing the cost of space access paved the way for soaring numbers of commercial and academic small spacecraft projects (Camps, 2019). These spacecraft are often designed using COTS hardware due to budgetary reasons. As the COTS hardware does not have the inherent radiation resistance required by spacecraft, a different approach is needed. Some of the methods currently used by COTS-based fault-tolerant CubeSat hardware are listed below:

- Using flash-based non-volatile FPGAs instead of SRAM-based FPGAs which are more susceptible to SEEs.

- Scrubbing on SRAM-based FPGAs.
- TMR and other fault-tolerant design methods on FPGA logic.
- Hardware and software error correction on volatile and non-volatile memory.
- Using inherently radiation-resistant memory technologies such as MRAM and FRAM instead of DRAM, SRAM, and flash.
- Multiple layers of software and hardware watchdogs.
- Board-level current-limiting against SELs.

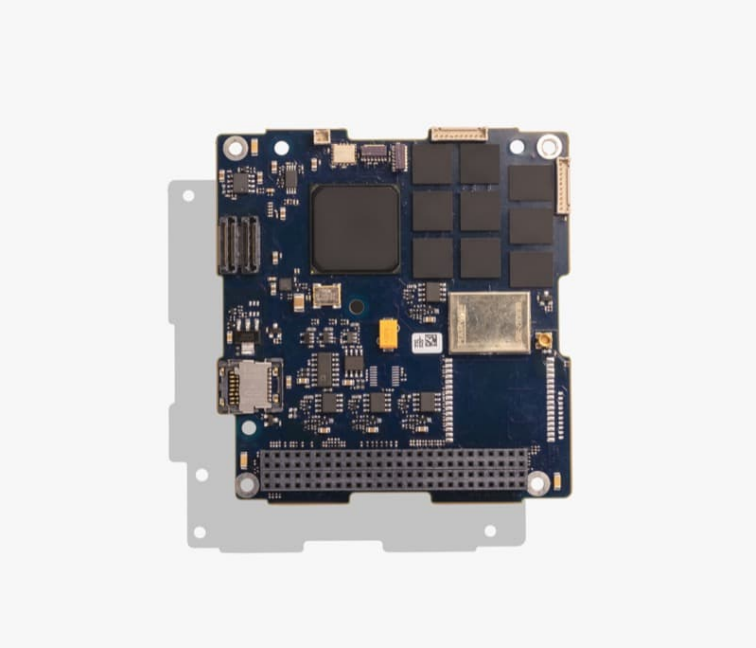


Figure 1.3 AAC Clyde Space's KRYTEN-M3 satellite on-board computer uses COTS parts and achieves fault-tolerance using a combination of the listed measures (AAC Clyde Space, 2020).

Currently, there are several RISC-V-based microprocessor designs claiming fault tolerance, such as Klessydra (Barbirotta, Cheikh, Mastrandrea, Menichelli, Vigli & Olivieri, 2021), DuckCore (Li, Zhang & Bao, 2022), and NOEL-V (Andersson, 2020). Although realizing such designs using FPGAs is possible, with the increasing availability and decreasing cost of low-volume ASIC production, some academic and commercial institutions might also be interested in realizing fault-tolerant microprocessor designs in ASICs. PULP project from ETH Zurich and the University of Bologna have already started working on such modular redundant RISC-V-based ASIC designs (Milanovic, Rogenmoser & Egimann, 2022; Rogenmoser & Jiang, 2022).

Although low-volume ASIC production is usually not available in cutting-edge nodes, radiation-tolerant designs are already better off using several generations behind SOI nodes due to their lower susceptibility to SEEs. Google and SkyWater Technology recently announced that they are releasing the open-source PDK (process design kit) for SKY90-FD process. SKY90-FD is a 90 nm fully depleted silicon on insulator process which is especially suited for radiation tolerant applications (Google & Skywater, 2022).

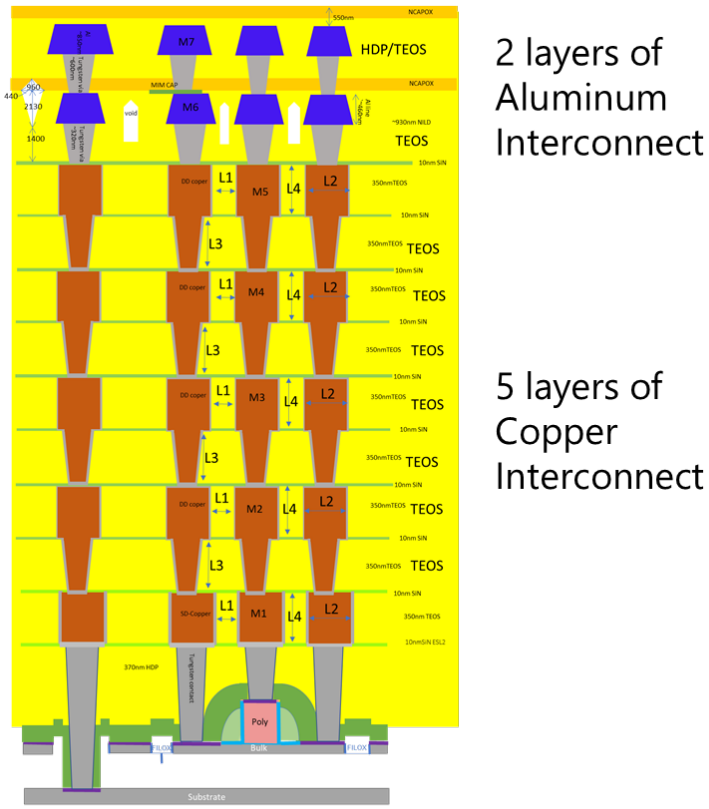


Figure 1.4 SKY90-PD stack topology diagram (Euphrosine & Mahintorabi, 2014).

Looking ahead to these developments, it can be beneficial to visit the core-level TMR approach applied to a simple non-redundant RISC-V core. The design might prove useful as a co-processor or a housekeeper in an FPGA or ASIC design.

1.2 Thesis Organization

The thesis starts by introducing the problem and the objective. Chapters 2, 3, and 4 explain the key concepts such as radiation effects on circuits, modular redundancy, and RISC-V instruction set architecture respectively. The system designed to study the concept is explained in detail in chapter 5. In chapter 6, the synthesis and benchmark results of the conceptual system are evaluated. Finally, the conclusion and possible future work are discussed in the last chapter.

2. Radiation Effects on Circuits

The performance and survivability of devices operating in radiation environments such as in space, aircraft, and particle accelerators are threatened by the effects of radiation and energetic particles on microelectronics. Radiation affects the functionality of the microelectronics mainly by disrupting the normal operation of the transistors and memory cells in various ways. The impact of radiation on electronic circuits can be classified as cumulative and single-event effects (George, 2019).

2.1 Cumulative Effects

Cumulative effects arise from the gradual buildup of charges (TID) and crystal lattice defects (DDD) caused by the ionized particles. Mitigation of the cumulative effects is usually done at the process or part levels, such as by using wide-bandgap materials and careful layout design (George, 2019).

2.1.1 Total Ionizing Dose

In MOS structures, passing ionizing radiation can create free electron-hole pairs. While these free electrons can be quickly dissipated by the switching electric fields thanks to their high mobility, holes can be trapped in the gate oxide. This accumulation results in a gradual lowering of the threshold voltage, eventually causing NMOS devices to stuck on and PMOS devices to stuck closed (George, 2019).

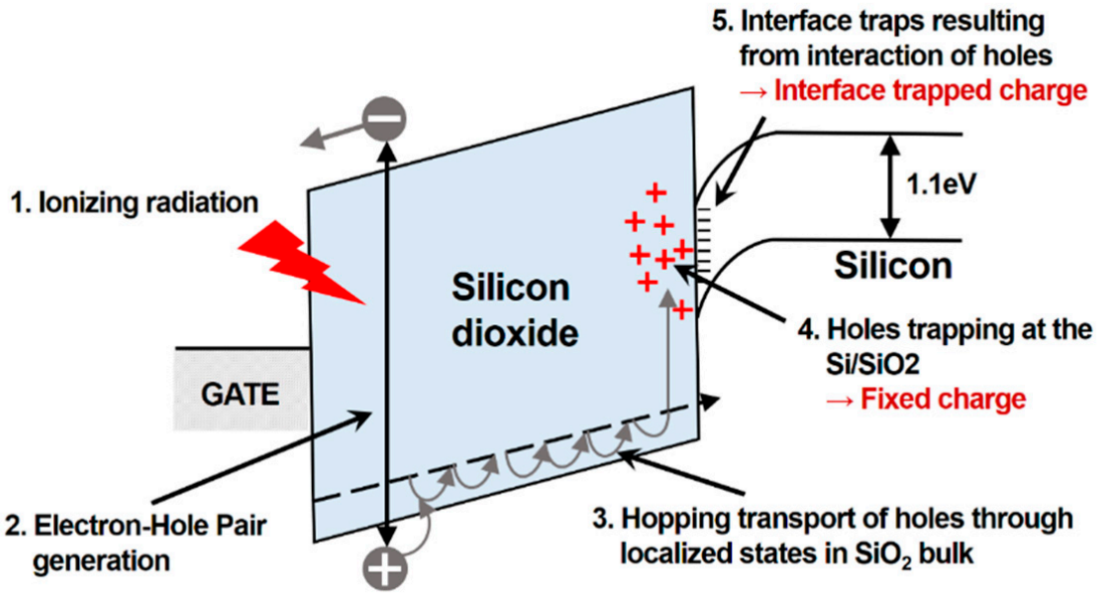


Figure 2.1 Total ionizing dose cause trapped charges inside the silicon (Lee, Lee, Kim, Hwang & Cho, 2021).

2.1.2 Displacement Damage Dose

When passing through a semiconductor device, even the non-ionizing high-energy particles can knock some of the atoms out of their crystal lattice structure and cause dislocations. These dislocations disrupt the electronic band structure of the semiconductor material. These kinds of defects lower the gain of the amplifier-type structures, eventually causing performance degradation in linear circuits (George, 2019).

2.2 Single-Event Effects

Single-event effects (SEE) are caused by a single passing particle, which in turn ionizes the material along its track and can cause various disruptive and destructive effects. Mitigation of SEEs can be done at various stages including circuit and software levels. Examples of SEE mitigation strategies are presented in section 2.3.

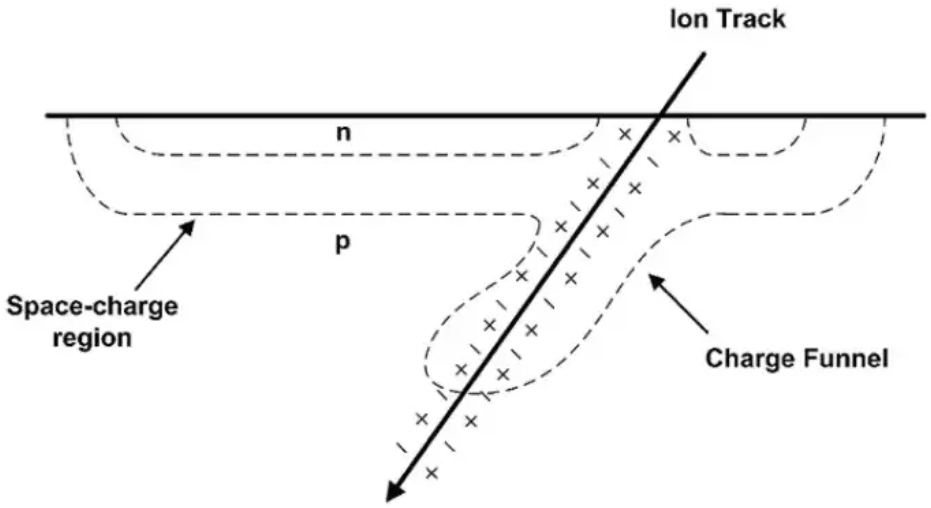


Figure 2.2 A charged particle ionizes the material around it while passing through a transistor. This can cause tunneling of the charges to unwanted regions (Payne, 2014).

2.2.1 Destructive Single-Event Effects

Some of the single-event effects can be energetic enough to create an ionized track across two opposite polarity regions of the device. This can create a short circuit with a positive feedback loop that cannot be broken until the destruction of the device or a power cycle. Single event latch-up (SEL), single event burnout (SEB), and single event gate or dielectric rupture (SEGR/SEDR). These events can make the device nonoperational or cause latent performance problems (George, 2019).

2.2.2 Non-destructive Single Event Effects

Non-destructive single-event effects are caused by ionizing radiation or particles that momentarily disrupts the operation of semiconductor devices. These effects can cause a bit-flip in memory devices (single-event upset) or a transient signal (single-event transient) that can travel across the circuit causing momentary incorrect logic states (George, 2019).

Single-event upset (SEU) is one of the most commonly occurring recoverable radia-

tion effects in microelectronic circuits (George, 2019). SEUs mainly affect memory cells which are commonly occurring structures in integrated circuits. After an SEU occurs, the circuit continues to operate, but since the stored data is changed, the device outputs incorrect results. SEUs are also known as "soft errors" since their effect can be masked or corrected via error correction and redundancy.

The charge injected into a node via an ionizing particle can cause a voltage spike. This voltage spike is known as a single-event transient (SET). While single-event transients do not cause errors by themself, they can propagate through connected nodes. SETs can damage sensitive circuitry and be captured as upsets when captured by sequential circuits (George, 2019). SETs propagation speed is directly proportional to the clock speed of the circuit. Thus, it is one of the leading factors that limit the operating speed of radiation-resistant circuits (Dodd, Shaneyfelt, Schwank & Felix, 2010).

2.3 Mitigation of Radiation Effects

Mitigation of the cumulative radiation effects is mainly done by using an appropriate process such as Silicon-on-Insulator (SOI) or by specialized layout techniques such as the placement of guard rings. Mitigation of destructive SEEs can be also done using a robust process technology (eg. larger nodes). Higher-level methods are also used frequently to protect a circuit from single-event latch-up events. For example, some of the space-qualified CubeSat hardware uses external circuitry to detect current spikes caused by an SEL event. Such external hardware can be placed at the power rails of a COTS integrated circuit to act as a resettable fuse and protect the IC in an event of SEL before it is destroyed (Kafi, Maeda, Kim, Masui & Cho, 2017). Protection methods frequently used for COTS-based spacecraft hardware are discussed in chapter 1.

Mitigation of non-destructive single-event effects (SEU, SEL) is getting more important as the popularity of using COTS hardware in spacecraft increases due to cost reasons. One of the most frequently used methods for SEU mitigation is error detection and correction. Since SEU mainly affects memory, different error correction schemes applied in hardware or software can overcome the effects of radiation on memory devices. Such error correction code (ECC) memory devices are widely used by industrial products in addition to radiation-tolerant hardware. While the

ECC can solve the majority of the memory-related soft errors, CPUs have many internal SRAM-based structures such as registers and other sequential logic. An effective way to protect other sequential logic from single-event effects is modular redundancy. Modular redundancy-based protection methods are discussed in chapter 3.

3. Redundant Systems

Redundancy can be described as having excess resources than necessary to use them in case of failure. Critical systems, such as spacecraft, often use different forms of redundancy to increase reliability.

3.1 Types of Redundancy

Redundancy can be classified under four different forms: hardware, software, information, and time (Koren & Krishna, 2007). These redundancy techniques are often used in conjunction with each other to further improve reliability. Although this work examines them from the processor's point of view, redundancy is a general concept and it can be applied to many different systems.

3.1.1 Hardware Redundancy

To implement hardware redundancy, additional hardware is used in the system to detect or mitigate the effects of an erroneous unit. Hardware redundancy can be static or dynamic. One of the most common forms of static hardware redundancy is triple modular redundancy, where hardware units are triplicated and compared against each other to detect and override the failed unit. In dynamic hardware redundancy, duplicated units are spare and they are only activated when a unit fails. Combining static and dynamic hardware redundancy is also possible and it is called hybrid hardware redundancy (Koren & Krishna, 2007). Avionics and automotive systems commonly use hardware reliability techniques where high reliability is required and extra overhead is acceptable.

3.1.2 Information Redundancy

Information redundancy is one of the most commonly used redundancy forms. Error-correcting codes (ECC) are the best-known redundancy method where extra bits of information are used to detect and correct the bulk data. It is used in everyday computers to wired and wireless communication systems. It is usually the first line of defense against bit flips since it can be easily implemented in hardware or software with comparatively low overhead. Almost every mission-critical system and many industrial computer systems use ECC memory. Flash-based non-volatile memory components even require ECC to work in normal conditions since the amount of data is so vast and the construction of the flash cells is too error-prone.

3.1.3 Time Redundancy

Since most of the failures are transient by their nature, they are unlikely to affect the same part of the system in a small timeframe. Time redundancy can be used to increase reliability by executing the same instructions or transmitting the same data multiple times. Time redundancy is easy to implement in software and it does not require extra hardware. However, it has a large performance penalty. Time redundancy can be used with other forms of redundancy to increase performance and reliability. Lockstep CPUs are one of the best examples where hardware and time redundancy are used together to increase reliability and performance at the expense of some hardware overhead. Lockstep CPUs execute the same program in two identical cores using a shifted or delayed clock. Since execution happens at slightly different times, it is unlikely to be affected by transient failures. Execution results are then compared at the output to catch transient faults.

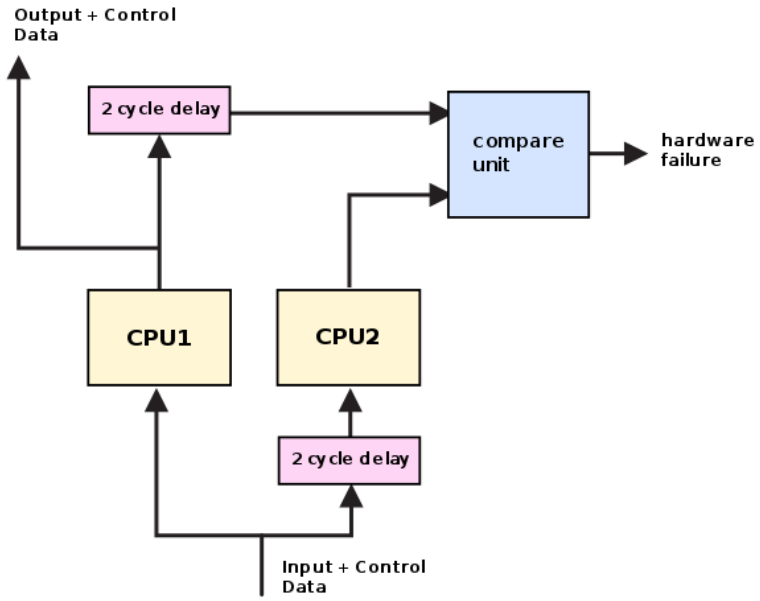


Figure 3.1 Lockstep computing is an example of hardware-based time redundancy.

3.1.4 Software Redundancy

Systems can be protected against software failures using software redundancy. Software faults, such as bugs, can be mitigated using two different versions of the software developed by a different team using different architectures (Koren & Krishna, 2007). A simpler and less accurate version of the main software can also be used as a backup or as a reference to check the main software. Software redundancy methods can also be used in conjunction with other forms of redundancy.

3.2 N-Modular Redundancy

N-Modular redundancy is a form of hardware redundancy where an M-of-N system consists of N modules and requires at least M modules to operate correctly. Triple modular redundancy (TMR) is a form of N modular redundancy. In TMR, three separate systems run independently with the same input, and the outputs of the three systems are compared against each other using a voting circuit. In case of a failure in one of the three systems, the other two still produce the correct output, and

the voting circuitry can mask the failing system, outputting the correct result. After that, the failing system can be isolated permanently or can be recovered depending on the system design. This ability gives TMR systems the ability to silently correct a single error without any disruption to the system. After a system is permanently disabled due to a recurring failure, the TMR system can only work in a reduced redundancy mode, where another failure can be detected but not corrected. More reliable systems can be constructed using more than three redundant systems but because of the size, power, and cost limits, TMR is the most commonly used form of modular redundancy. Early practical examples of TMR computer systems are used in the Saturn launch vehicle starting in 1961 (Kuehn, 1969).

Although the main goal of building modular redundant systems is to make the system more reliable than the simplex counterpart, it is possible to make a system less reliable by introducing modular redundancy as seen from figure 3.2. Since the module reliability is often much larger than 0.5 we can see that N-modular redundancy makes the system more reliable. However, poorly designed N-modular redundant systems are prone to common mode failures affecting all the modules. For a processor system, common mode failure can be the clock or the voting logic.

The general formula for the reliability of an N-modular redundant system can be expressed using equation 3.1 where q_{qor} is the probability of a common failure.

$$R^{cor}_{M\_of\_N}(t) = (1 - q_{qor}) \sum_{i=M}^{N} \binom{N}{i} R^i(t)[1 - R(t)]^{N-i} \tag{3.1}$$

If we only consider the voter as the only source of a common failure, equation 3.1 becomes equation 3.2 for a TMR system where R_{voter} is the reliability of the voting logic.

$$R_{TMR}(t) = R_{voter}(t) \sum_{i=2}^{3} \binom{3}{i} R^i(t)[1 - R(t)]^{3-i} \tag{3.2}$$

3.2.1 Granularity

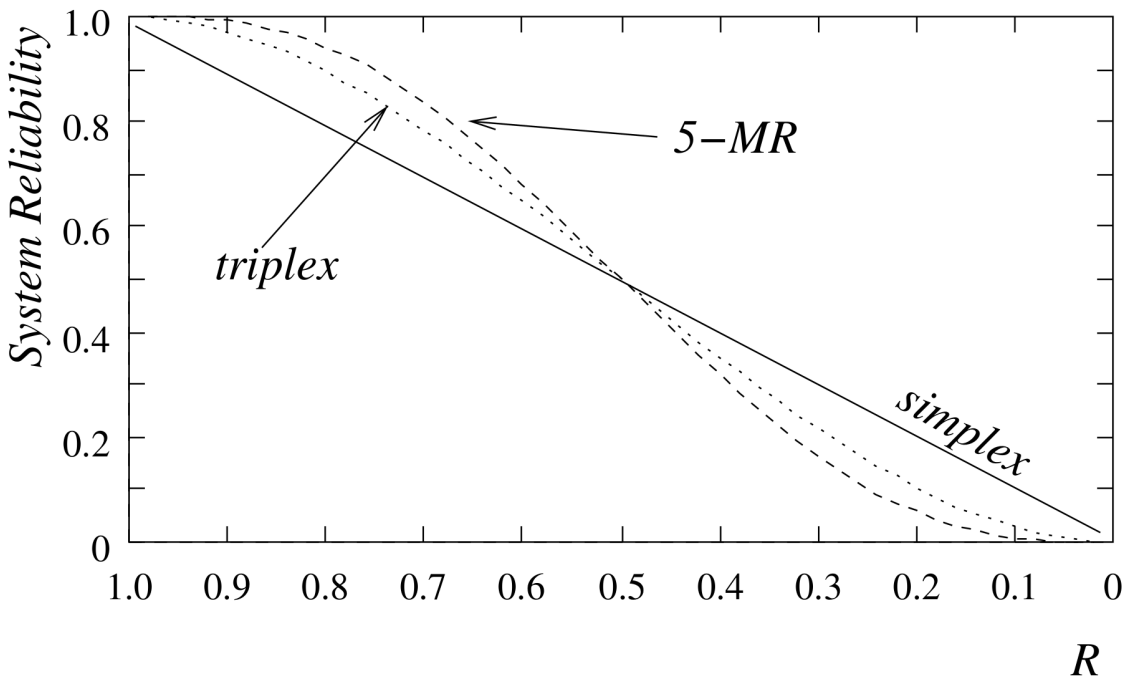


Figure 3.2 Simplex, triplex (TMR), and 5-MR systems were compared for system reliability (Koren & Krishna, 2007).

Modular redundancy can be applied with different granularity. Coarse-grain modular redundancy is applied at IP or module level by applying voters at the output. Finer-grain redundancy can be applied on logic or flip-flop level by applying intermediate voting at multiple stages. While finer grain modular redundancy has low overhead and simple to apply, fine-grain modular redundancy can be more robust and flexible. Optimizing these parameters by using medium-grain redundancy is also possible (Kretzschmar, Astarloa, Lázaro, Garay & Del Ser, 2012). Some examples of modular redundancy at different granularites are given in figure 3.3. This work a coarse-grain triple modular redundancy at the processor level, where the complete CPU is triplicated without modification.

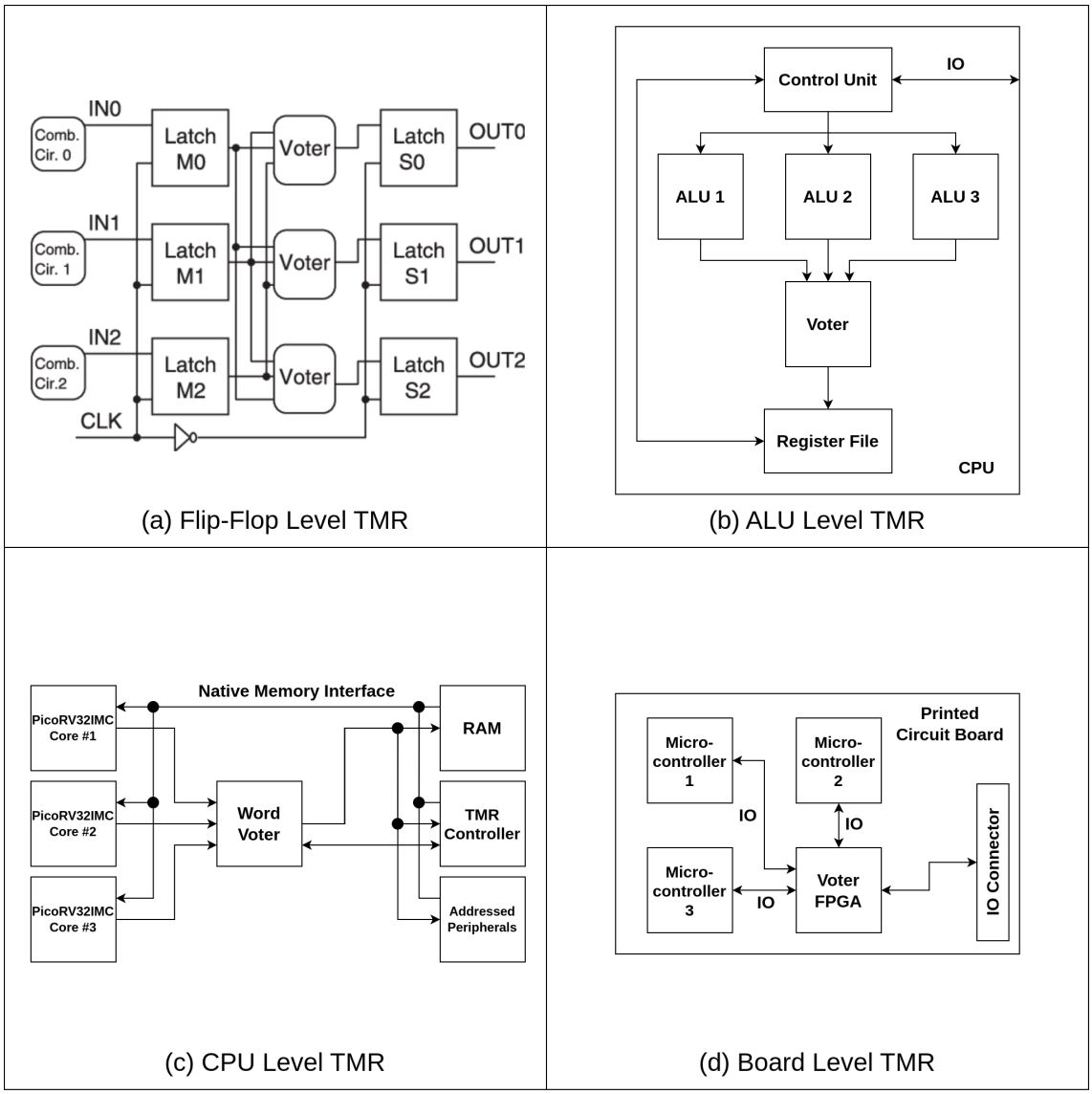


Figure 3.3 Examples of hardware redundancy applied at different granularity levels. Figure a is taken from Furuta, Kobayashi & Onodera (2010).

3.3 Voters

Voters are the devices that receive inputs from redundant modules and generate a representation of the output. Voters can be exact or approximate depending on the application. For simple modular redundant systems, where each module is the exact copy of each other, bit-by-bit or word voters can be used. Approximate voters are useful for complicated systems where modules can generate slightly different outputs (Balen, González, Oliveira, Schvittz, Added, Macchione, Aguiar, Guazzelli, Medina

& Butzen, 2021).

Voter circuits create a common failure point for redundant systems. This weakness can be remedied using different approaches. The simplest approach is to increase voter reliability by design. Since the voter circuit is generally much simpler than the rest of the system, their reliability is higher. Special logic gates, which are more durable to SEU by design, can be used to construct voter circuitry.

Replicating the voters with the rest of the system is another approach. Although this approach increases the complexity, it is commonly used on mission-critical systems where complexity and cost are insignificant.

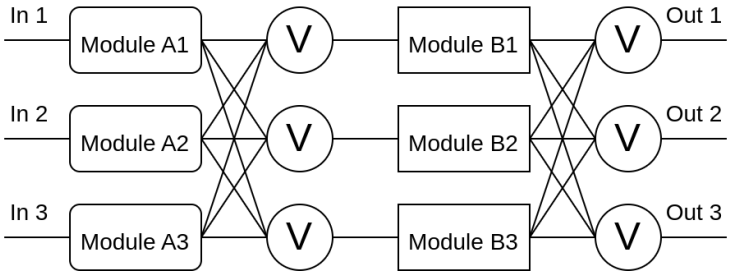


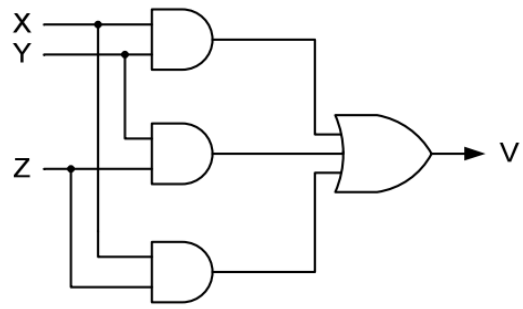
Figure 3.4 Triplicated voters prevent single point of failure.

Another way to increase voter reliability is self-checking voters. Self-checking voters significantly increase the system's reliability without as much overhead as replicating voters (Afzaal & Lee, 2018; Cazeaux, Rossi & Metra, 2004).

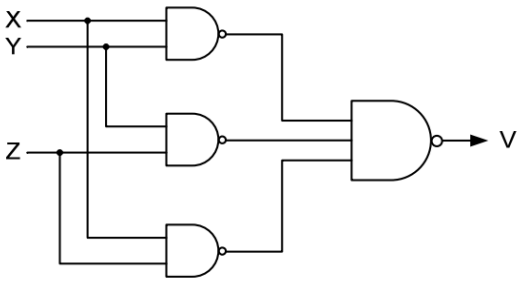
3.3.1 Majority Voter

The majority function is a function that evaluates whether more than half of the inputs are true or false. In Boolean algebra, the majority function is represented using equation 3.3. The majority voter is a logic block that realizes the majority function using logic gates. The majority voter is one of the fundamental blocks used in modular redundant systems. The majority voter circuits can be implemented using different logic blocks (Balasubramanian & Mastorakis, 2016).

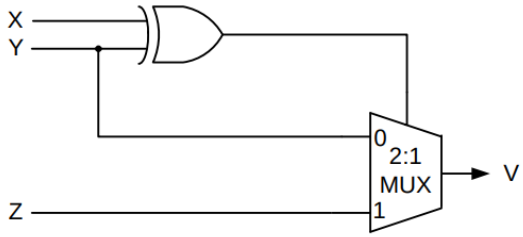
As voters have critical importance for the reliability of the system, voting circuitry can be optimized for speed, area, and SEU immunity using specialized logic blocks. The advantages and disadvantages of different majority voter designs are explored by Aguiar, Wrobel, Autran, Leroux, Saigné, Pouget & Touboul (2020).



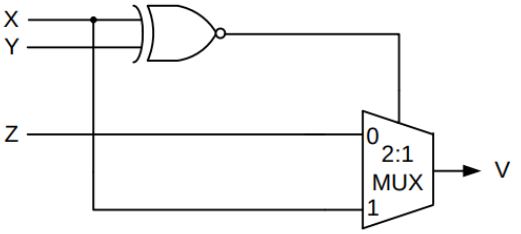
(a) Conventional AND OR type majority voter.



(b) Conventional NAND type majority voter.



(c) XOR type Ban-Naviner majority voter.



(d) XNOR type Ban-Naviner majority voter.

Figure 3.5 Majority voters can be realized using a variety of gates. Each has a different area, power, speed, and radiation tolerance (Balasubramanian & Mastorakis, 2016).

$$\langle p_1, \cdots, p_n \rangle = Majority(p_1, \cdots, p_n) = \left\lfloor \frac{1}{2} + \frac{(\sum_{i=1}^{n} p_i) - 1/2}{n} \right\rfloor \tag{3.3}$$

3.3.2 Word Voter

Using a bit-by-bit majority voter on a wide memory bus has some disadvantages. Since the majority voter compares each bit of the bus individually, it can mask some errors even when the data integrity of the bus is actually compromised. This can cause errors to hide longer and delay the corrective action, decreasing overall system reliability. An example situation where word voter can reveal hidden errors is represented in table 3.1.

Mitra & McCluskey (2000) came up with the word voter to solve this problem. Word voter design adds additional circuitry to conventional voters for generating an error

Module	Fault-free Output	Faulty Output 1	Faulty Output 2	Faulty Output 3
A	0 0 0	0 0 1	0 0 1	1 0 0
B	0 0 0	0 0 0	0 1 0	0 0 1
C	0 0 0	0 0 0	0 0 0	0 1 0
Bit-by-bit Voter	0 0 0	0 0 0	0 0 0	0 0 0
Word Voter	0 0 0	0 0 0	No majority	No majotiy

Table 3.1 A 3-bit example where conventional voters can mask actual errors.

signal when the bus integrity fails even by a single bit. Although this adds some logic overhead, it is shown that using word voters instead of conventional bit-by-bit voter significantly increases the data integrity (Ló, Kastensmidt & Beck, 2014). Properties of the word voters are explored more deeply in section 5.2.

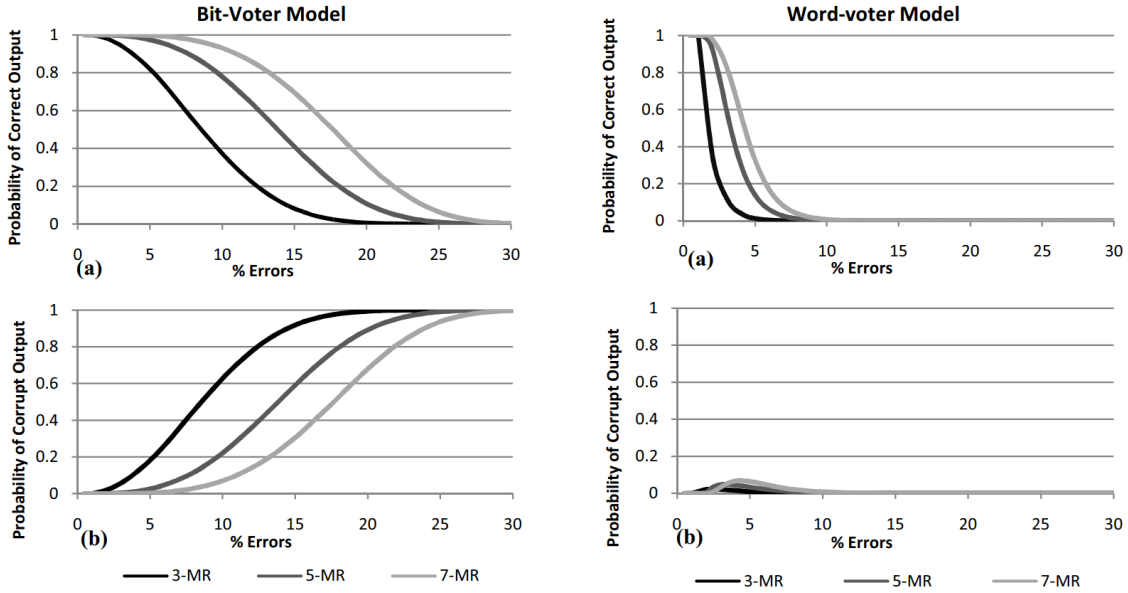


Figure 3.6 Probability of correct and corrupt outputs of NMR bit and word voters compared (Ló et al., 2014).

4. RISC-V

RISC-V is an open-source instruction set architecture designed using RISC (Reduced instruction set computer) principles. RISC-V instruction set is free and provided under open-source licenses. Started development in 2010 at the University of California Berkeley, RISC-V ISA gained broad support from many industry partners (RISC-V International, 2022). It started appearing in commercial products, aside from many open-source and academic projects.

RISC-V ISA consists of a base instruction set and instruction set extensions, providing modular design for a wide range of applications from embedded microcontrollers to out-of-order superscalar processors. RISC-V ISA also supports custom instruction set extensions for hardware acceleration at the instruction level for specialized applications.

As RISC-V is open-source, extensible, and commonly used in industry and academia, in this work an open-source RISC-V compatible processor core is used to demonstrate TMR functionality.

4.1 Instruction Set

RISC-V is a RISC (Reduced Instruction Set Computer) type of ISA, which aims to reduce the complexity and the number of available instructions. Unlike the CISC (Complex Instruction Set Computer) architecture, a reduced instruction set allows simpler hardware and code but requires more instructions to perform complex operations. This disadvantage can be offset by using higher clock speeds and pipelining.

RISC-V instruction set consists of a base ISA and optional extensions (Waterman, Lee, Patterson & Asanović, 2011). The general structure of a RISC-V instruction can be seen in table 4.1. As of the writing of this work, some of the extensions are

Format	Bit																															
	31							25	24				20	19				15	14		12	11				7	6						0
Register	funct7								rs2					rs1					funct3			rd					opcode						
	31												20	19				15	14		12	11				7	6						0
Immediate	imm [11:0]													rs1					funct3			rd					opcode						
	31																				12	11				7	6						0
Upper immediate	imm [31:12]																					rd					opcode						
	31							25	24				20	19				15	14		12	11				7	6						0
Store	imm [11:5]								rs2					rs1					funct3			imm [4:0]					opcode						
	31	30					25	24					20	19				15	14		12	11		8	7	6						0	
Branch	[12]	imm [10:5]						rs2						rs1					funct3			imm [4:1]			[11]	opcode							
	31	30								21	20	19									12	11				7	6						0
Jump	[20]	imm [10:1]									[11]	imm [19:12]										rd					opcode						

Table 4.1 Structures of the RISC-V base instruction formats. Where opcode field specifies the format, funct field specifies the operation type, and rd, rs1, and rs2 specify the target and destination registers (Waterman & Asanović, 2019).

still open and subject to improvements. These instruction sets with their current status can be seen in table 4.2.

4.2 PicoRV32

PicoRV32 is an open-source CPU core developed by Claire Xenia Wolf and YosysHQ (YosysHQ, 2019). PicoRV32 supports the RISC-V RV32IMC instruction set. It is a simple in-order size-optimized design that can be used as an auxiliary CPU in FPGA and ASIC applications. PicoRV32 also supports a co-processor interface and can use a native, AXI-Lite, or Wishbone memory interface. Some of the key capabilities of the PicoRV32 are listed below.

- Configurable with RV32I, RV32IC, RV32IM, RV32IMC or RV32E instruction set support.
- Built-in optional custom interrupt controller.
- Selectable native, AXI4-Lite master or Wishbone master memory interface.
- Optional Co-Processor interface.
- Single-port or dual-port register file implementation.
- Configurable trace output.
- Optional barrel shifter.
- Optional look-ahead memory interface.

Name	Description	Status
Base		
RVMO	Weak Memory Ordering	Ratified
RV32I	32-bit Base Integer Instruction Set	Ratified
RV32E	32-bit Embedded Base Integer Instruction Set (16 registers)	Open
RV64I	64-bit Embedded Base Integer Instruction Set	Ratified
RV128I	128-bit Embedded Base Integer Instruction Set	Open
Extension		
M	Integer Multiplication and Division	Ratified
A	Atomic Instructions	Ratified
F	Single-Precision Floating-Point	Ratified
D	Double-Precision Floating-Point	Ratified
Zicsr	Control and Status Register (CSR)	Ratified
Zifencei	Instruction-Fence Fetch	Ratified
G	Short for IMAFDZicsr_Zifencei	-
Q	Quad-Precision Floating-Point	Ratified
L	Decimal Floating-Point	Open
C	Compressed Instructions	Ratified
B	Bit Manipulation	Ratified
J	Extension for Dynamically Translated Instructions	Open
T	Extension of Transactional Memory	Open
P	Packed-SIMD Instructions	Open
V	Vector Operations	Frozen
K	Extension for Scalar Cryptography	Ratified
N	User-Level Interrupts	Open
S	Supervisor-level Instructions	Ratified
Zam	Misaligned Atomics	Open
Ztso	Total Store Ordering	Frozen

Table 4.2 RISC-V Instruction Set Architecture base and extensions (Waterman & Asanović, 2019; Waterman, Asanović & Hauser, 2021).

PicoRV32 is designed to be simple and area-optimized, not performance. It is capable of achieving high clock speeds, allowing it to be used within high-speed systems as a controller without crossing clock domains.

In this thesis, the PicoRV32 core is used to build a TMR system to demonstrate a fault-tolerant CPU for spacecraft use. PicoRV32 is selected among other similar RISC-V CPUs because of its simple, proven design. PicoRV32 is validated and optimized to work on many FPGA architectures (YosysHQ, 2019). PicoRV32 is also made ASIC-proven thanks to the eFabless Raven SoC, which is using X-FAB's XH018 0.18 micron process (eFabless, 2018,1).

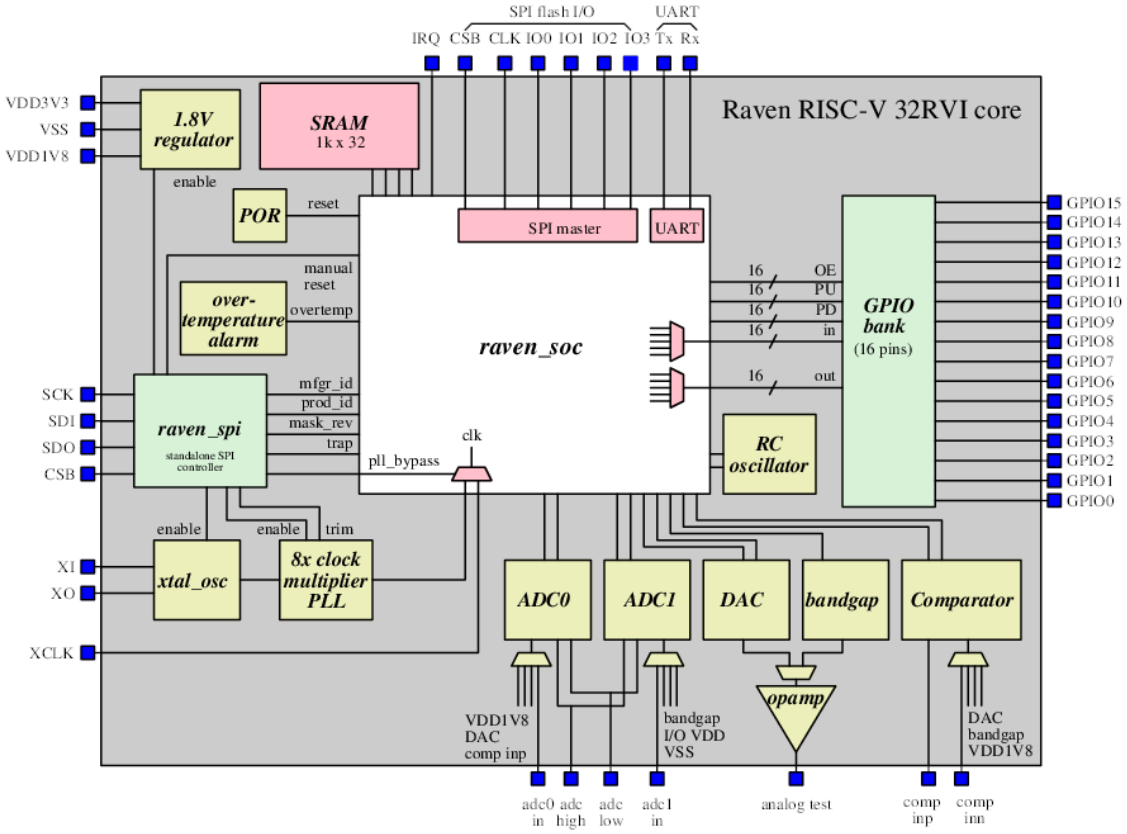


Figure 4.1 Block diagram of eFabless RAVEN SoC, based on the PicoRV32 (eFabless, 2018).

4.2.1 PicoRV32 Core

PicoRV32 is a size-optimized non-pipelined RISC-V RV32IMC core. The core is extensively configurable. Some of the configurable options include instruction set extensions from RV32E to RV32IMC, interrupt support, barrel shifter, dual-port

Instruction	Cycles-per-Instruction (Single Port)	Cycles-per-Instruction (Dual Port)
Direct jump (jal)	3	3
Indirect jump (jalr)	6	6
ALU (imm + reg)	3	3
ALU (reg + reg)	4	3
Branch (not taken)	4	3
Branch (taken)	6	5
Memory load	5	5
Memory store	6	5
Shift operations	4-15	4-14

Table 4.3 Instruction timings of the basic PicoRV32 processor (YosysHQ, 2019).

registers, and many more. PicoRV32's multiplication and division support is implemented using an internal co-processor and not by the integrated ALU. The co-processor interface is explained in detail in section 4.2.3. The instruction timings representing the performance of the core are given in table 4.3.

PicoRV32 supports interrupts. However, the interrupt handling does not follow RISC-V ISA specifications. Interrupt handling uses a small number of custom instructions and 4 additional registers.

4.2.2 Memory Interface

PicoRV32 features a simple native memory interface that can transfer one word per cycle. PicoRV32 also supports AXI-Lite and Wishbone master interfaces optionally. An adapter module that can convert native memory signals to AXI-Lite is provided.

In addition to the main native memory interface, PicoRV32 includes an optional look-ahead memory interface. Look-ahead memory interface has the same basic signals as the native memory interface and it provides information about the next memory cycle. Systems with dual-port RAM can benefit from this look-ahead interface by decreasing memory latency.

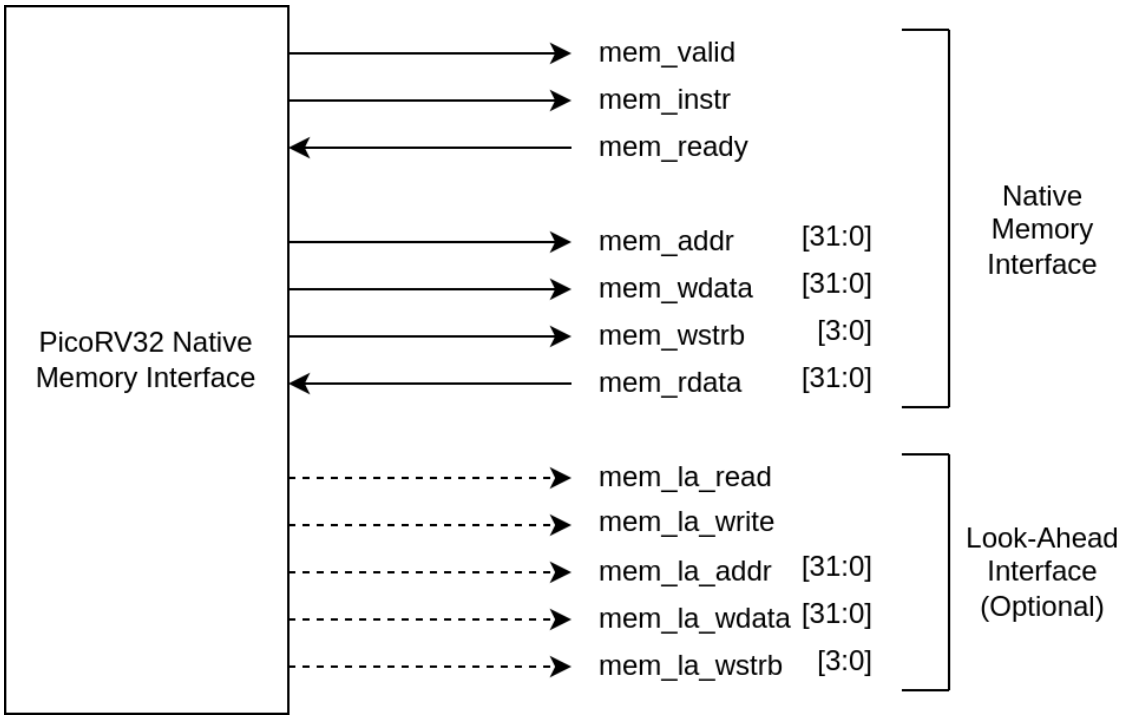


Figure 4.2 Signals of the PicoRV32 native memory interface.

4.2.3 Pico Co-Processor Interface

PicoRV32 implements an optional co-processor interface. Unsupported instructions are presented to this interface when encountered by the CPU. An external co-processor core can decode this instruction and then load the result to the pcpi_rd register when processing finishes. This makes adding custom instructions and accelerating them with external logic extremely easy.

When the PicoRV32 is configured to have M instruction extensions, RISC-V multiplication and division instructions are processed by separate multiplication and division accelerator cores that are connected to the PCPI internally.

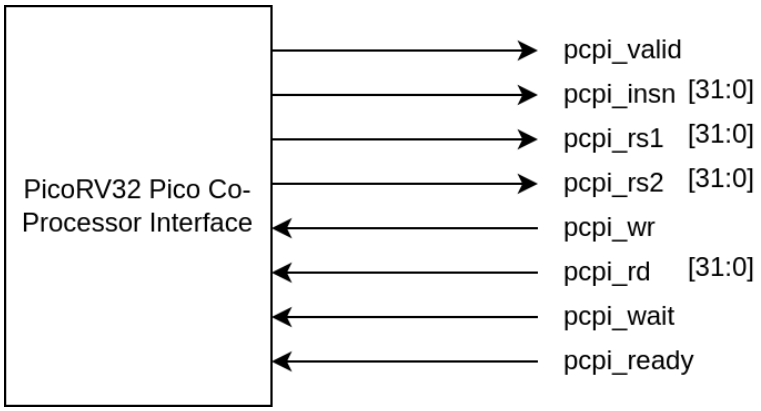


Figure 4.3 Signals of the PicoRV32 Pico Co-Processor Interface.

4.2.4 Compiler

PicoRV32 can use the standard RISC-V GNU toolchain that supports RV32I[M][C] instruction sets. However, since the PicoRV32 uses custom instructions for interrupt handling. These instructions are defined under the PicoRV32 repository using GNU assembler macros.

5. System Design

A concept system that implements triple modular redundancy measures discussed in the previous chapters is implemented. While the system does not necessarily designed to be practical and usable, it is designed to show how a triple redundant system can be developed based on an existing RISC-V processor.

5.1 CPU Design

PicoRV32 CPU in RV32IMC configuration is used in the system which supports hardware multiply-divide and compressed instructions. Interrupt support is also enabled. In this configuration, PicoRV32 is suitable for general-purpose housekeeping applications.

While PicoRV32 can implement AMBA AXI-Lite and Wishbone bus interfaces, a simple native memory interface is used in this design for ease of implementation. For larger and more complex applications where lots of addressable peripheral devices are used, a standard bus interface can be used. Trace output is also enabled for debugging.

Three of the identically specified PicoRV32 cores are placed in the design. Output signals of the bus interface are grouped together and fed to voters. This grouping is done according to the functionality of signals. Each functional group has its own voter. This is done to have more insight into the offending part of the faulty CPU during testing and evaluation. All of the output signals can be also fed into a single voter to decrease area.

Separate register files are included in each CPU module. Implementing a common register file for all CPUs is possible but it should be protected by an appropriate error correction scheme for preserving fault tolerance. Improving error tolerance

further by also applying ECC to separate register files is also possible and used by many fault-tolerant CPUs.

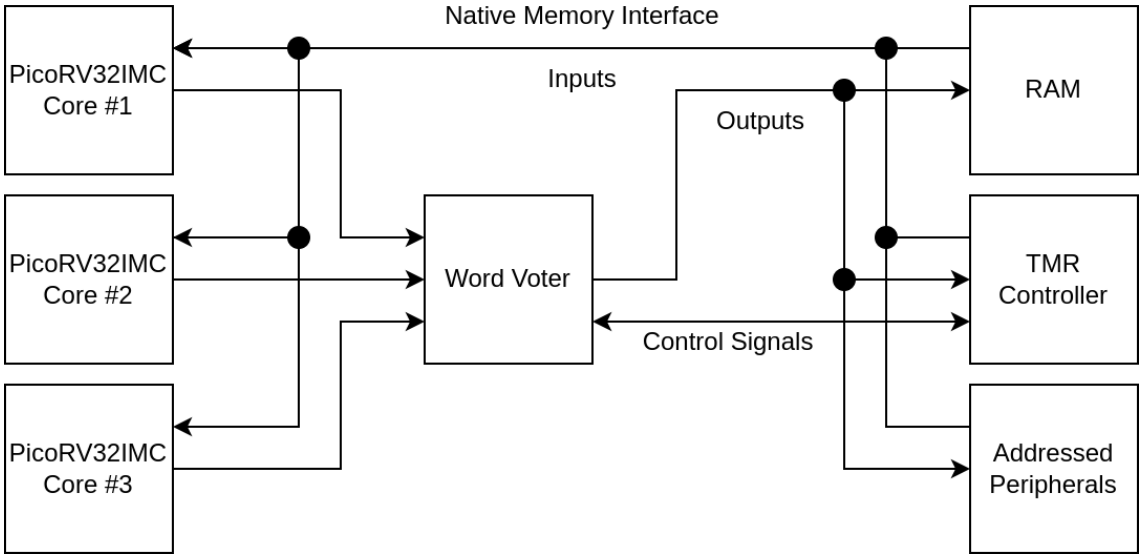


Figure 5.1 Core-level TMR architecture of the system.

5.1.1 Native Memory Interface

PicoRV32 has a simple native memory interface which is explained in detail in section 4.2.2. For the demonstration system, a simple direct access memory model is used. The same memory model is converted to block RAM primitives automatically during FPGA synthesis. Look ahead and external PCPI interfaces are not used.

In the demonstration system, the memory does not have error correction capability. For a real fault-tolerant system, the memory would certainly include some error correction capability.

5.2 Word Voter Design

The system uses word voters to implement TMR architecture. Designed word voters have parametric word width. Word voters consist of 3 comparators that can compare two equal-length vectors. The width of the comparators is also defined

parametrically. Each bit of the two input vectors is compared individually in the comparators. The comparison result of each bit then AND'ed together. The comparator outputs high, if the two input vectors match, otherwise it outputs low. A mux selects channel A if channels A and C match, otherwise it selects channel B. Channel C is solely used as a comparison reference and never routed to the output. If all three channels differ, the MUX still selects B, although it does not represent a valid output.

Faulty Module	**Error Output**	**Output Status**
-	0 0 0	Valid
A	1 0 0	Valid
B	0 1 0	Valid
C	0 0 1	Valid
Multiple modules	Other combinations	Invalid

Table 5.1 Truth table for the error output.

Error generation logic generates a 3-bit output. All-low error output shows fault-free operation. Each bit of the error output corresponds to a faulty module. Multiple high bits in the error signal indicates multiple module failure. In that case, output and error signals are invalid and cannot be trusted. The truth table for the possible values and states is included in table 5.1.

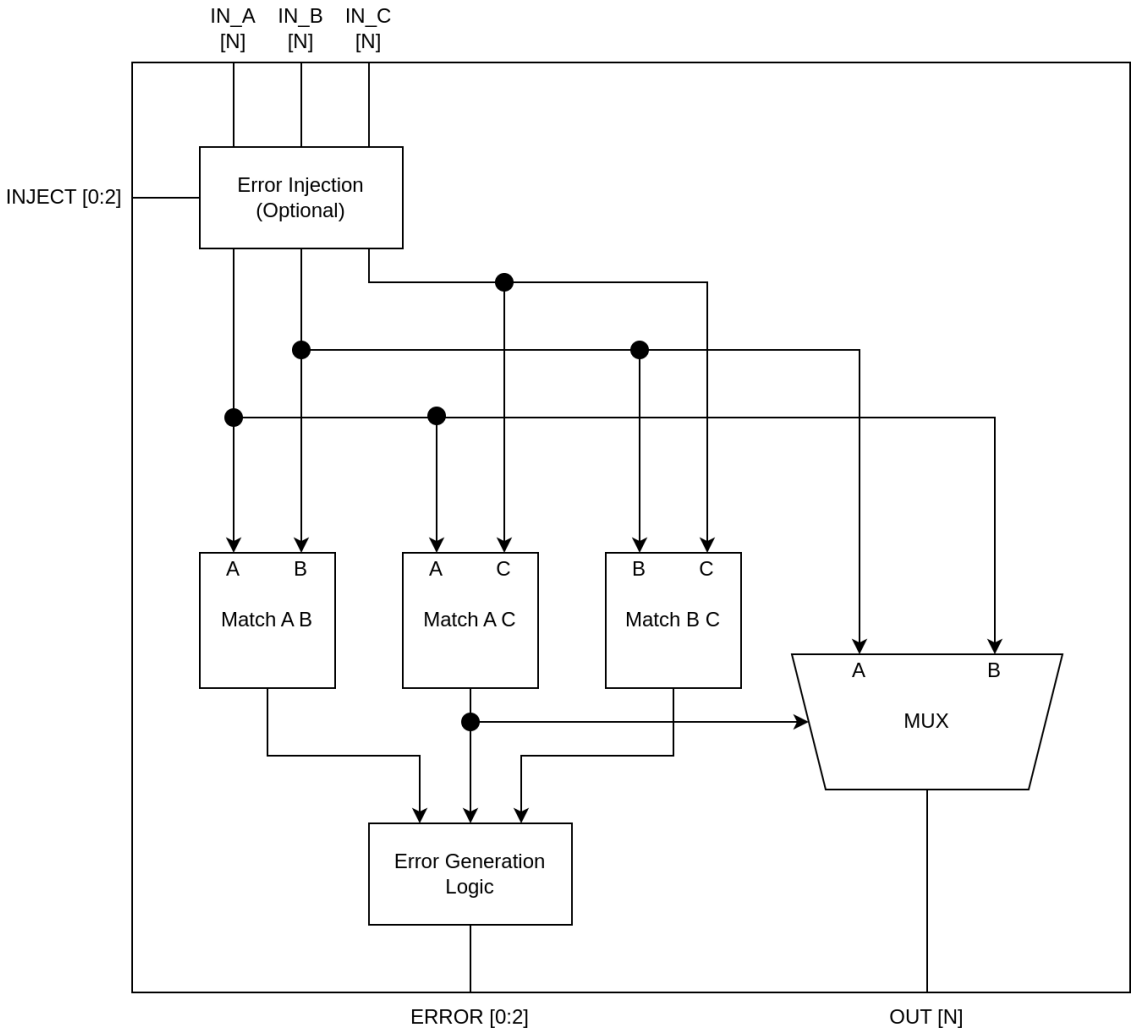


Figure 5.2 Block diagram of the word voter.

Injecting errors into one of the input signals is also possible with the optional error injection input. Inject input is a 3-bit vector. Each bit injects a single-bit error to the corresponding input signal. Error injection logic is disabled by default and can be enabled for simulation and testing.

The RTL level comparators are designed using XNOR and AND gates. It is possible to synthesize the comparators using more optimized logic.

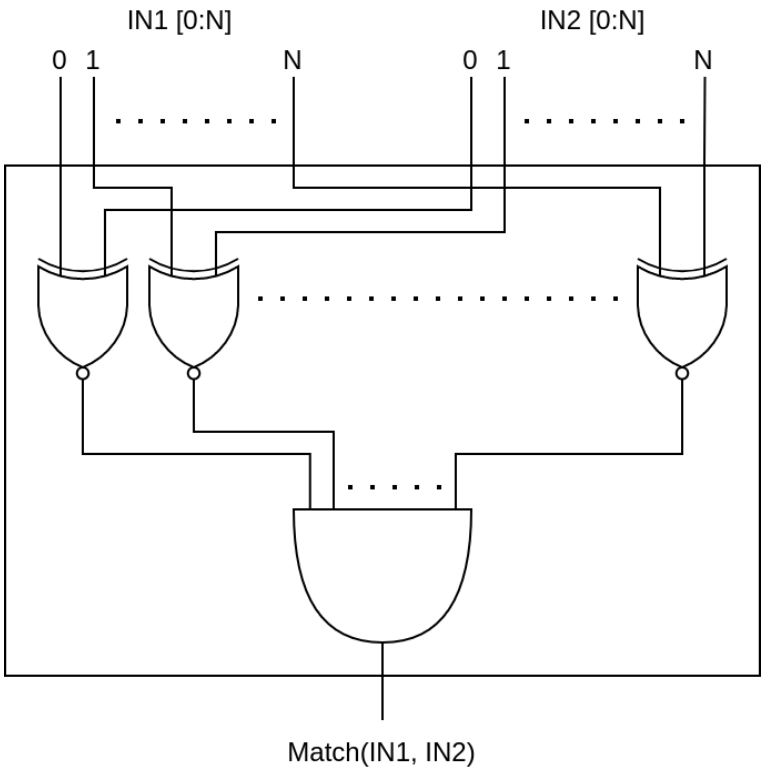


Figure 5.3 Gate-level diagram of the word match circuit.

5.3 TMR Controller

A TMR controller module is proposed for tracking the redundancy status of the system with software. The TMR controller module will be connected to the native memory bus as an addressable peripheral. Software running on the CPU can interface the module using registers.

The software can read the error status of each voter, and optionally inject errors to the voters for testing using the TMR controller's registers. TMR controller can also reset the CPUs, disable a specific core, or generate interrupts in case of an error.

By responding to the interrupt, the software can be aware that the CPU is working in the reduced redundancy mode (when 1 of the 3 modules fails). A specialized piece of code can wait for critical tasks to finish then it can reset the CPU using the TMR controller, trying to clear the faulty module. If the offending module repeatedly fails, the software can disable the module completely using the TMR controller.

The TMR controller module is a useful interface between software and the TMR system as it allows the software to be fault-aware. Software fault awareness can be very beneficial, especially for critical real-time software, as the software can reset the system to clear the error without hampering the critical software processes.

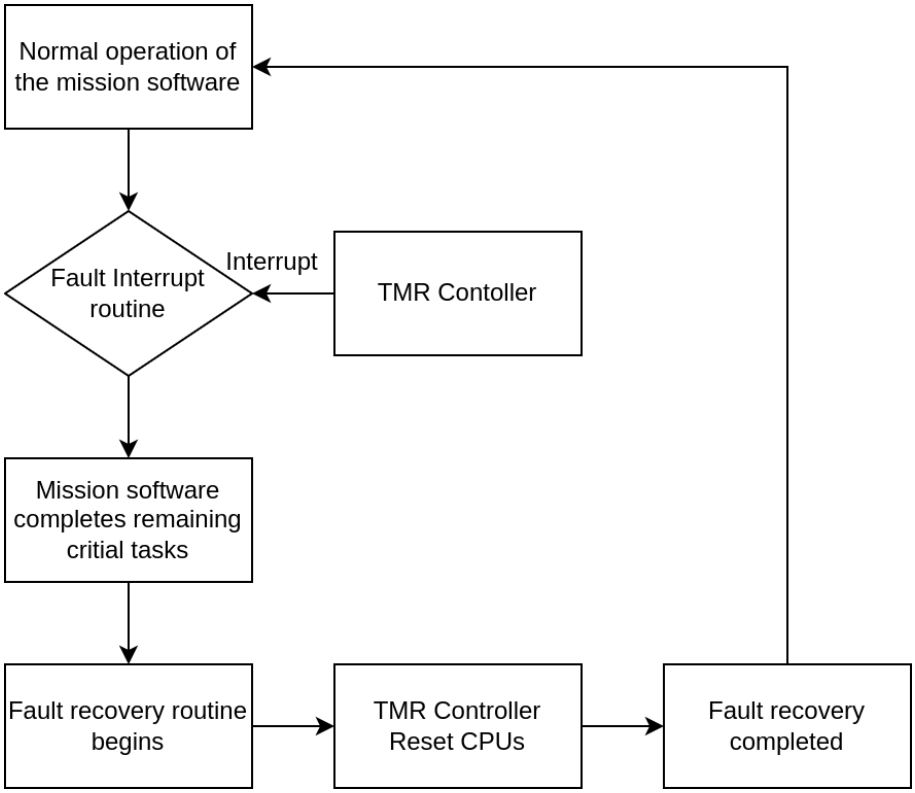


Figure 5.4 Software can initiate fault recovery routine without interrupting critical tasks.

6. Synthesis and Benchmarks

To investigate the feasibility of the RISC-V-based TMR approach, the concept design is simulated and synthesized. Simulation results of the TMR system are compared with the simplex system for functional verification. Synthesis results are used to understand the area and performance penalty compared to other approaches.

6.1 Environment Setup

PicoRV32's open-source GitHub repository includes all the source code written in Verilog HDL. The repository also includes scripts for simulation, validation, and synthesis using a variety of tools. Table 6.1 lists the specifications and versions of the tools used in this study. For TMR implementation, a new repository is created by forking the latest PicoRV32 repository. Voters and other support circuitry are implemented using SystemVerilog HDL. Wrappers and their corresponding testbenches are created to integrate three identically configured PicoRV32 cores into a single module, that is compatible with the original PicoRV32 design.

Tool	**Purpose**	**Version**
Icarus Verilog	Behavioral simulation	12.0
GNU Make	Simulation automation	4.2.1
Xilinx Vivado	FPGA synthesis	v2020.2 (64-bit)
FPGA	FPGA target device for synthesis	Xilinx Artix 7 XC7A100T-1CSG324C

Table 6.1 Tools used for system simulation and synthesis

6.2 Simulation

PicoRV32 repository includes Verilog testbenches for simulating the design. The testbenches can simulate the PicoRV32 with different configurations at the RTL level. The test system uses a makefile so that firmware that will run during the is also compiled before the testbench runs and automatically loaded by the simulation tool during the test. The makefile can run the simulation using Icarus Verilog or Verilator. The makefile also allows setting different configurations to PicoRV32 from the command line. For the TMR design, a new SystemVerilog testbench is created based on the existing ones. The TMR testbench is designed with the ability to inject errors into voters at the desired time and it can show the error status of each voter. Using this testbench, the design is verified functionally including the TMR logic. Icarus Verilog is used to run the simulations. The simulation results of the TMR system are compared with the original simplex system to verify that the design is functionally identical.

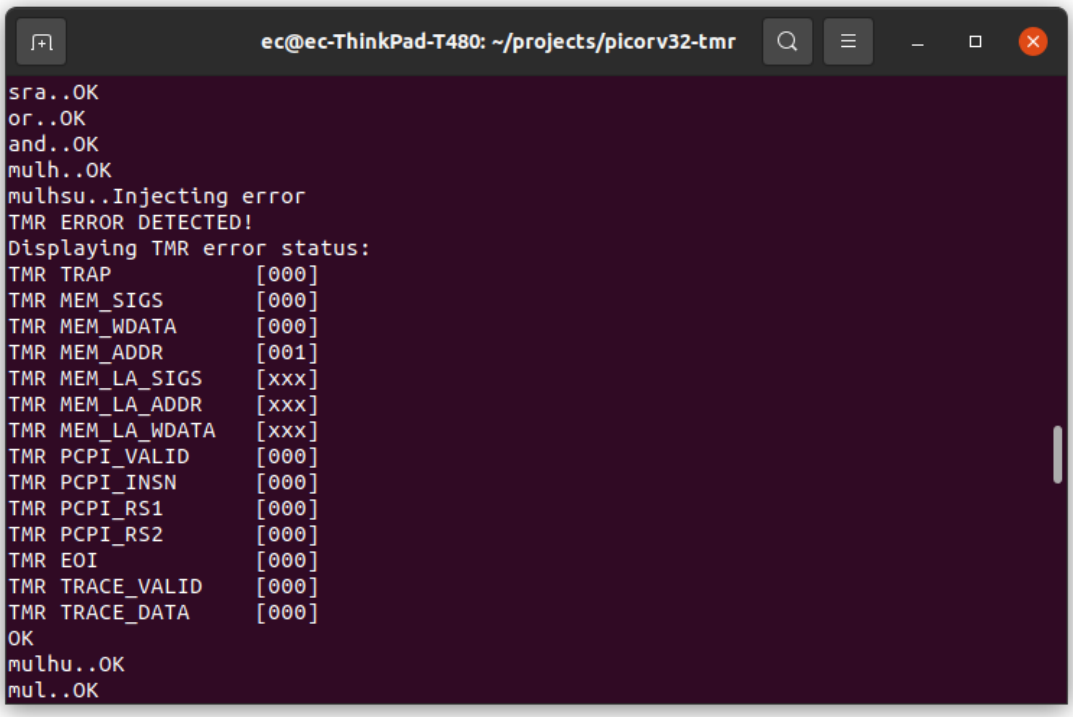


Figure 6.1 Screenshot of error injection and recovery of the TMR system during simulation.

6.3 Area and Performance

Both simplex and TMR system is synthesized using Yosys and Xilinx Vivado to understand the burden of TMR for area and performance. The following results are obtained. The results are also compared with other TMR implementations. Note that the granularity of applied modular redundancy is different for some designs.

Design	**Synthesis**		**Implementation**	
	Maximum Clock	**Worst Negative Slack**	**Maximum Clock**	**Worst Negative Slack**
Simplex	100 MHz	0.739 ns	100 MHz	0.488 ns
TMR	77 MHz	0.734 ns	77 MHz	0.131 ns

Table 6.2 Comparison of the Vivado synthesis and implementation timings for reference simplex and TMR systems.

Design	**Synthesis**		**Implementation**	
	LUT	**Register**	**LUT**	**Register**
Simplex	2135	1155	2151	1155
TMR	7031	3687	6812	3609

Table 6.3 Comparison of the Vivado synthesis and implementation area for reference simplex and TMR systems.

These results show that the application of TMR in this granularity level increased the area more than 3 times and decreased the maximum achievable clock speed. This is expected since the application of TMR at this level basically triplicates the whole circuitry and adds additional voter logic. The goal of this approach is to make the design triple-modular-redundant without modifying the internal logic to simplify development.

Many of the similar TMR processor implementations uses finer grained TMR approach than ours. Several designs uses tools to automate module triplication and voter insertion (Wilson & Wirthlin, 2019,2) while some of them manually modified the CPU architecture (Ramos, Toral, Reviriego & Maestro, 2019; Santos, Luza, Zeferino, Dilillo & Melo, 2020). Properties and tools used for the similar TMR implementations are presented in table 6.4.

In table 6.5 and 6.6, speed and area results of the PicoRV32-TMR design compared with other TMR processor designs. Wilson & Wirthlin (2021) use a Xilinx Artix 7

Work	TMR Implementation	Processors Type
1	Fine-grain TMR using SpyDrNet	MicroBlaze, PicoRV32, Kronos, Taiga, VexRiscv
2	Fine-grain TMR using BL-TMR	Taiga
3	Custom hybrid TMR architecture	Custom RISC-V RV32IM
4	ALU based TMR	Rocket RISC-V
5	TMR ALU and Hamming protected fetch unit and register file	Custom RISC-V RV32I
6	Fine-grain TMR using Mentor Precision Hi-Rel	Rocket RISC-V
7	Coarse-grain and fine-grain distrubuted TMR using Cadence	Rocket RISC-V
8	Distrubuted TMR using Synplify	VexRiscv

Table 6.4 Redundant Processor designs that use different TMR architectures. 1: Wilson & Wirthlin (2021), 2: Wilson & Wirthlin (2019), 3: Shukla & Ray (2022), 4: Ramos et al. (2019), 5: Shukla & Ray (2022), 6: Aranda, Wessman, Santos, Sanchez-Macian, Andersson, Weigand & Maestro (2020), 7: de Oliveira, Tambara, Benevenuti, Benites, Added, Aguiar, Medina, Silveira & Kastensmidt (2020), 8: Minnella (2018)

Design	**Simplex**		**TMR**		**Ratio**	
	LUT	**FF**	**LUT**	**FF**	**LUT**	**FF**
MicroBlaze	2122	2019	8619	6057	4.05×	3×
PicoRV32	985	586	4089	1758	4.15×	3×
Kronos	1612	913	6663	2739	4.13×	3×
Taiga	2896	1626	11532	4878	3.98×	3×
VexRiscv	3095	2789	12363	8367	4×	3×
PicoRV32-TMR (This work)	2151	1155	6812	3609	3.17×	3.12×

Table 6.5 Comparison of TMR area overhead with results from Wilson & Wirthlin (2021).

Design	**Area**		**Frequency [MHz]**
	LUT	**FF**	
Non-hardened RV32I	1613	1024	74
TMR RV32I	2387	1024	66
Hamming RV32I	1854	1216	54
TMR and Hamming RV32I	2784	1216	49
PicoRV32-TMR simplex (This work)	2135	1155	100
PicoRV32-TMR (This work)	7031	3687	77

Table 6.6 Comparison of area and speed results from Santos et al. (2020).

XC7A200T and Santos et al. (2020) use a Xilinx ZYNQ ZC7020 as target FPGAs. For our results, a Xilinx Artix 7 XC7A100T FPGA is used. These devices use the exact same FPGA logic architecture thus results are directly comparable.

From the table 6.5, we can see that a finer-grain TMR architecture has more overhead than our coarse-grain TMR implementation. Our implementation has only a slight overhead of 0.17× in addition to the triplicated logic. Finer-grain implementation used by Wilson & Wirthlin (2021) have almost 1× of overhead which is caused by multiple levels of voters used between combinational logic blocks of the processors. Note that PicoRV32 based system implemented by Wilson & Wirthlin (2021) have a lower area than our PicoRV32 implementation since our implementation uses a fully-featured RV32IMC class PicoRV32 instead of a RV32I.

In table 6.6, we compared our speed and area results with Santos et al. (2020) which implements a custom RV32I class RISC-V CPU using a hybrid TMR and ECC based hardening. From the results, we can see that Santos et al. (2020)'s custom non-hardened RV32I CPU is roughly comparable in area and speed to our PicoRV32 configuration. Since Santos et al. (2020) only applies TMR to ALU of the CPU, its overhead is smaller than our whole processor TMR implementation.

6.4 Reliability Expectations

There are multiple approaches to test the reliability of a fault-tolerant system. We can categorize these methods as:

- Simulation based fault injection.
- FPGA based fault injection.
- Physical irradiation.

or combination of these three.

Simulation based fault injection can be implemented on the RTL or the post-synthesis netlist. In order to inject error during the simulation, an fault injection logic must be designed and integrated into the DUT. This has significant drawbacks since DUT itself must be modified to accept error signals into the registers. In practice, only the most error-prone portion of the circuit can be designed with in mind to fault injection in order to minimize intrusion. For example, a error-tolerant

CPU can be designed to have a register file with fault injection logic for testing since most SEU affects the registers. This still require modification to the actual logic, which can change the actual behavior of the system. Also, simulation based fault injection is usually limited for testing of SEU effects. Some of the simulation tools such as Modelsim and Verilator can be scripted to inject faults to flip-flops during simulation (Kooli & Di Natale, 2014; Na & Lee, 2011). Some environments that can modify the HDL code automatically during simulation to inject faults are also exist for advanced applications (Kooli & Di Natale, 2014).

FPGA emulation is also a widely used method to test redundant systems (Wilson & Wirthlin, 2021). Xilinx Software Error Mitigation IP module can be integrated into the FPGA design to inject faults at the FPGA configuration memory. While, this approach is useful for many FPGA based designs, it is insufficient since it can only inject faults at the configuration memory. Faults effecting the flip-flops and registers cannot be tested using the Xilinx SEM IP. Seperate logic must be implemented to test these building blocks in FPGA. Since FPGA based designs and ASIC designs have wildly different building blocks, their response to faults are different. Becasue of this FPGA based fault injection results may not be very useful for ASIC implementations.

Physical irradiation is the gold standard for testing the fault-tolerant circuits. Neutron radiation or heavy-ion bombardment from cyclotrons can be used to simulate actual space radiation environment (de Oliveira et al., 2020; Wilson, Larsen, Wilson, Thurlow & Wirthlin, 2021; Wilson & Wirthlin, 2019). Both ASIC and FPGA based designs can be tested using this method. Since it creates a real radiation environment, systems response to all kinds of single event effects can be observed. The main disadvantage of this method is cost. It is also not practical for testing during design phase.

For this work, no reliability testing is made. The purpose of the designed system is to show implementation stages of a coarse-grain word voter based TMR architecure to a ready-made RISC-V processor. The reliability estimations will be tightly dependent to used IP, FPGA architecture and ASIC design flow. This work also conceptualize a TMR controller module that can be used to create software recovery routines without disrupting the real-time tasks of the processor.

7. Conclusion

As seen from the results of the conceptual system, a coarse-grain processor-level triple-modular redundant processor can be useful for space applications. The main benefit of using coarse-grain TMR is that the CPU core itself does not need to be modified. This approach is especially useful when one needs to add redundancy to an existing system with little development effort. This is usually the case with commercial CPU IP cores such as ARM as they provide black box IP. Small satellite projects are often required to use non-fault-tolerant general-purpose processors because of cost reasons. Providing redundancy to general-purpose processor IPs using the demonstrated method can be valuable for low-cost spacecraft projects. The TMR-capable PicoRV32 system completely preserves the characteristics of the original PicoRV32 processor. The simple, customizable, and size-optimized design of the PicoRV32 can prove useful for applications such as co-processors, housekeeping, or watchdog-type applications.

Using word voters instead of classical bit-by-bit majority voters provides additional error detection capability. A conceptual TMR controller is developed to track the error status of the cores, generate interrupts, and reset the CPU in order to clear the non-permanent errors. As the optional TMR controller is implemented as an additional peripheral, it allows running software to behave according to the fault status. Legacy software written for the non-redundant design can also run without modification since the processor-level TMR approach does not necessitate any architectural modifications.

7.1 Future Work

Circuit layout plays a significant role in the reliability of the circuit. Careful layout planning should minimize the multi-bit upsets. For our work, each CPU core should

be placed on the die such that an energetic particle cannot cross the two modules at the same time. Using distributed versus clustered logic layouts should also be studied. The distributed layout is where the place & route tool optimizes the placement of gates from all modules in order to enhance timing. The distributed layout may cause the placement of logic blocks from different CPU cores in the same vicinity. This can increase the multi-bit error rate.

The addition of other types of redundancy is also essential for a practical spacecraft application. Implementation of ECC for the memory and register file causes very little overhead compared to TMR and greatly enhances reliability as seen from Santos et al. (2020). The inclusion of inherently radiation-resistant memory technology is also possible. MRAM and FRAM technologies are resistant to radiation by design and they can be licensed as IPs from different vendors. Usage of MRAM and FRAM is getting increasingly common for space applications (AAC Clyde Space, 2020).

BIBLIOGRAPHY

AAC Clyde Space (2020). Command & Data Handling, KRYTEN-M3. https://www.aac-clyde.space/wp-content/uploads/2021/10/AAC_DataSheet_Kryten.pdf.

Afzaal, U. & Lee, J.-A. (2018). A self-checking tmr voter for increased reliability consensus voting in fpgas. *IEEE Transactions on Nuclear Science*, *65*(5), 1133–1139.

Aguiar, Y. Q., Wrobel, F., Autran, J. L., Leroux, P., Saigné, F., Pouget, V., & Touboul, A. D. (2020). Design exploration of majority voter architectures based on the signal probability for tmr strategy optimization in space applications. *Microelectronics Reliability*, *114*.

Andersson, J. (2020). Development of a noel-v risc-v soc targeting space applications. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, (pp. 66–67).

Andersson, J., Gaisler, J., & Weigand, R. (2010). Next generation multipurpose microprocessor. In *European Space Agency, (Special Publication) ESA SP*, volume 682 SP.

Aranda, L., Wessman, N.-J., Santos, L., Sanchez-Macian, A., Andersson, J., Weigand, R., & Maestro, J. (2020). Analysis of the critical bits of a risc-v processor implemented in an sram-based fpga for space applications. *Electronics*, *9*, 175.

BAE Systems (2008). *RAD750 radiation-hardened PowerPC microprocessor*. Manassas, Virginia.

Balasubramanian, P. & Mastorakis, N. (2016). Power, delay and area comparisons of majority voters relevant to tmr architectures.

Balen, T. R., González, C. J., Oliveira, I. F. V., Schvittz, R. B., Added, N., Macchione, E. L. A., Aguiar, V. A. P., Guazzelli, M. A., Medina, N. H., & Butzen, P. F. (2021). Reliability evaluation of voters for fault tolerant approximate systems. In *2021 IEEE 22nd Latin American Test Symposium (LATS)*, (pp. 1–6).

Barbirotta, M., Cheikh, A., Mastrandrea, A., Menichelli, F., Vigli, F., & Olivieri, M. (2021). A fault tolerant soft-core obtained from an interleaved-multi- threading risc- v microprocessor design. In *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, (pp. 1–4).

Camps, A. (2019). *Nanosatellites and Applications to Commercial and Scientific Missions*.

Cazeaux, J., Rossi, D., & Metra, C. (2004). New high speed cmos self-checking voter. In *Proceedings. 10th IEEE International On-Line Testing Symposium*, (pp. 58–63).

Cobham Gaisler AB (2021). LEON3FT Fault-tolerant processor. https://www.gaisler.com/index.php/products/processors.

de Oliveira, B., Tambara, L. A., Benevenuti, F., Benites, L. A. C., Added, N., Aguiar, V. A. P., Medina, N. H., Silveira, M. A. G., & Kastensmidt, F. L. (2020). Evaluating soft core risc-v processor in sram-based fpga under radia-

tion effects. *IEEE Transactions on Nuclear Science*, *67*(7), 1503–1510.

Dodd, P. E., Shaneyfelt, M. R., Schwank, J. R., & Felix, J. A. (2010). Current and future challenges in radiation effects on CMOS electronics. In *IEEE Transactions on Nuclear Science*, volume 57.

eFabless (2018). Raven. `https://platform.efabless.com/design_catalog/asic_platform/116`.

eFabless (2019). Raven: An ASIC implementation of the PicoSoC PicoRV32. `https://github.com/efabless/raven-picorv32`.

Euphrosine, J. & Mahintorabi, E. (2014). SkyWater and Google expand open source program to new 90nm technology. `https://opensource.googleblog.com/2022/07/SkyWater-and-Google-expand-open-source-program-to-new-90nm-technology.html`.

European Space Agency (2013). LEON's first flights. `https://www.esa.int/Enabling_Support/Space_Engineering_Technology/LEON_s_first_flights`.

Furuta, J., Kobayashi, K., & Onodera, H. (2010). An area/delay efficient dual-modular flip-flop with higher seu/set immunity. *IEICE Transactions*, *93-C*, 340–346.

George, J. S. (2019). An overview of radiation effects in electronics. In *AIP Conference Proceedings*, volume 2160.

Google & Skywater (2022). SkyWater SKY90FD Open Source PDK. `https://github.com/google/sky90fd-pdk#sky90fd-process-node`.

International Business Machines Corporation (1964). *Laboratory Maintenance Instructions, Saturn V Launch Vehicle Digital Computer, Volume I*. National Aeronautics and Space Administration.

Kafi, A., Maeda, G., Kim, S., Masui, H., & Cho, M. (2017). Design and implementation of single event latch-up measurement and self- recovery system for birds cubesat.

Kooli, M. & Di Natale, G. (2014). A survey on simulation-based fault injection tools for complex systems. In *2014 9th IEEE International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, (pp. 1–6).

Koren, I. & Krishna, C. (2007). *Fault-Tolerant Systems*. Elsevier Science.

Kretzschmar, U., Astarloa, A., Lázaro, J., Garay, M., & Del Ser, J. (2012). Robustness of different tmr granularities in shared wishbone architectures on sram fpga. In *2012 International Conference on Reconfigurable Computing and FPGAs*, (pp. 1–6).

Kuehn, R. E. (1969). Computer redundancy: Design, performance, and future. *IEEE Transactions on Reliability*, *R-18*(1), 3–11.

Lee, M., Lee, N., Kim, J., Hwang, Y., & Cho, S. (2021). Modeling and simulation-based layout optimization for tolerance to tid effect on n-mosfet. *Electronics*, *10*(8).

Li, J., Zhang, S., & Bao, C. (2022). Duckcore: A fault-tolerant processor core architecture based on the risc-v isa. *Electronics*, *11*(1).

Ló, T. B., Kastensmidt, F. L., & Beck, A. C. S. (2014). 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)*, (pp. 258–263).

Milanovic, L., Rogenmoser, M., & Egimann, M. (2022). Cerberus. `http://asic`

.ethz.ch/2022/Cerberus.html.

Minnella, F. (2018). Protection and characterization of an open source soft core against radiation effects. Presented 09 Apr 2018.

Mitra, S. & McCluskey, E. (2000). Word-voter: a new voter design for triple modular redundant systems. In *Proceedings 18th IEEE VLSI Test Symposium*, (pp. 465–470).

Na, J. & Lee, D. (2011). Simulated fault injection using simulator modification technique. *ETRI Journal*, *33*(1), 50–59.

Payne, D. (2014). Modelling and Analysis of Single Event Effects (SEE). https://semiwiki.com/x-subscriber/silvaco/3646-modeling-and-analysis-of-single-event-effects-see/.

Ramos, A., Toral, R. G., Reviriego, P., & Maestro, J. A. (2019). An alu protection methodology for soft processors on sram-based fpgas. *IEEE Transactions on Computers*, *68*(9), 1404–1410.

RISC-V International (2022). Hisory of RISC-V. https://riscv.org/about/history/.

Rogenmoser, M. & Jiang, Z. (2022). Trikarenos. http://asic.ethz.ch/2022/Trikarenos.html.

Santos, D. A., Luza, L. M., Zeferino, C. A., Dilillo, L., & Melo, D. R. (2020). A low-cost fault-tolerant risc-v processor for space systems. In *2020 15th Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, (pp. 1–5).

Shukla, S. & Ray, K. C. (2022). A low-overhead reconfigurable risc-v quad-core processor architecture for fault-tolerant applications. *IEEE Access*, *10*, 44136–44146.

Sun, C. (2006). Intel Pentium M 740 1.73ghz Socket 479 Processor Review. https://web.archive.org/web/20131029201847/http://www.pcstats.com/articleview.cfm?articleid=2008&page=4.

Tomayko, J. E. (1988). Computers in Spaceflight: the NASA Experience. In *Encyclopedia of Computer Science and Technology*, volume 18.

Tzanetos, T., Aung, M., Balaram, J., Grip, H. F., Karras, J. T., Canham, T. K., Kubiak, G., Anderson, J., Merewether, G., Starch, M., Pauken, M., Cappucci, S., Chase, M., Golombek, M., Toupet, O., Smart, M. C., Dawson, S., Ramirez, E. B., Lam, J., Stern, R., Chahat, N., Ravich, J., Hogg, R., Pipenberg, B., Keennon, M., & Williford, K. H. (2022). Ingenuity mars helicopter: From technology demonstration to extraterrestrial scout. In *2022 IEEE Aerospace Conference (AERO)*, (pp. 01–19).

Waterman, A. & Asanović, K. (2019). *The RISC-V Instruction Set Manual Volume I: Unprivileged ISA*.

Waterman, A., Asanović, K., & Hauser, J. (2021). *The RISC-V Instruction Set Manual Volume II: Privileged Architecture*.

Waterman, A., Lee, Y., Patterson, D. A., & Asanović, K. (2011). The risc-v instruction set manual, volume i: Base user-level isa. Technical Report UCB/EECS-2011-62, Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, California.

Wilson, A. E., Larsen, S., Wilson, C., Thurlow, C., & Wirthlin, M. (2021). Neutron radiation testing of a tmr vexriscv soft processor on sram-based fpgas. *IEEE Transactions on Nuclear Science*, *68*(5), 1054–1060.

Wilson, A. E. & Wirthlin, M. (2019). Neutron radiation testing of fault tolerant risc-

v soft processor on xilinx sram-based fpgas. In *2019 IEEE Space Computing Conference (SCC)*, (pp. 25–32).

Wilson, A. E. & Wirthlin, M. (2021). Fault injection of tmr open source risc-v processors using dynamic partial reconfiguration on sram-based fpgas. In *2021 IEEE Space Computing Conference (SCC)*, (pp. 1–8).

YosysHQ (2019). PicoRV32 - A Size-Optimized RISC-V CPU. `https://github.com/YosysHQ/picorv32`.