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

ÁDRIA BARROS DE OLIVEIRA

Fault tolerance characterization of RISC-V processors in SRAM-based FPGAs for aerospace applications

Thesis presented in partial fulfillment of the requirements for the degree of Doctor of Microelectronics

Advisor: Prof[a]. Dr[a]. Fernanda Lima Kastensmidt

Porto Alegre
May 2023

CIP — CATALOGING-IN-PUBLICATION

Oliveira, Ádria Barros de

Fault tolerance characterization of RISC-V processors in SRAM-based FPGAs for aerospace applications / Ádria Barros de Oliveira. – Porto Alegre: PGMICRO da UFRGS, 2023.

194 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR–RS, 2023. Advisor: Fernanda Lima Kastensmidt.

1. RISC-V. 2. Rocket. 3. NOEL-V. 4. Soft processor. 5. SRAM-based FPGAs. 6. Fault tolerance. 7. Fault injection. 8. SEE. 9. Heavy ions. 10. Protons. I. Kastensmidt, Fernanda Lima. II. Título.

“Coragem é seguir o coração mesmo com medo. É abraçar as vulnerabilidades e saber que o resultado das nossas ações não nos define.”

— ELISAMA SANTOS

ACKNOWLEDGMENT

First, and more importantly, I thank Jeckson, my partner, friend, and love. Thank you for all the support, care, and love. You held my hand during the most challenging times and helped me to have the strength to complete this thesis.

All gratitude and love to my parents and grandmothers for believing in me since always. I owe you everything.

Many thanks to my advisor for helping with the development of this thesis.

Thanks to CAPES, CNPq, and FAPERGS for supporting education in Brazil. Thanks to PGMICRO/UFRGS, LAFN-USP, and RADNEXT for making this thesis possible. Thanks to Frontgrade Gaisler for supporting this thesis.

Many thanks to all the friends that I have made at UFRGS and Gaisler.

ABSTRACT

Aerospace applications, such as small satellites, demand a certain level of reliability due to Single Event Effects (SEE). At the same time, Commercial Off-The-Shelf (COTS) devices are frequently used in New Space missions. In this context, using soft processors implemented in COTS SRAM-based Field Programmable Gate Arrays (FPGAs) requires design flexibility to apply the most suitable fault tolerance techniques for improving the system's reliability. Upsets in the FPGA configuration memory can be persistent. They may change the architectural implementation of the soft processor, which can cause control-flow errors, leading to Single Event Functional Interrupt (SEFI), and wrong computations, defined as Silent Data Corruptions (SDCs). This thesis aims to characterize the SEE susceptibility of RISC-V soft processors embedded in SRAM-based FPGAs and understand how combining fault tolerance techniques can significantly reduce system vulnerability. The investigation addresses the problems of using soft processors in SEE-prone environments and the complexities and trade-offs behind mitigation methods. The case studies are the open source RISC-V Rocket and NOEL-V soft processors and the commercial fault tolerant RISC-V NOEL-VFT. The processors are embedded in the Xilinx Zynq-7000 APSoC (28 nm CMOS), Zynq UltraScale+ MPSoC (16 nm FinFET), and Kintex UltraScale (20 nm CMOS), respectively. Protection is applied at the design level targeting the FPGA configuration memory, processor core, and embedded memories. Combined techniques such as Triple Modular Redundancy (TMR), scrubbing, periodic reset, watchdog, and memory refresh are used during the investigation. An external FPGA supervisor was also developed to increase fault coverage, reduce the chance of soft errors in the scrubbing interface, and provide better visibility of upsets in the FPGA configuration memory. The soft processors susceptibility to soft errors is assessed under emulation fault injection - mainly targeting the FPGA configuration memory - and accelerated ground testing - targeting the entire device under heavy ion and proton irradiations. The study on the Rocket's and NOEL-V's L1 cache proved that the application error rate is reduced at a more frequent memory refresh, making the use of larger memories feasible to reach better performance. The unprotected Rocket soft processor achieved about 88% of correctness under single faults, and the processor susceptibility under irradiation could be improved more than 51 times using a combination of fault tolerance techniques. Results on the open source NOEL-V showed the effectiveness of distributed TMR, scrubbing, and memory refresh to reduce the cross section by about 11 times. Applying triplication and

scrubbing to the commercial fault tolerance NOEL-VFT soft processor boosted the reliability almost 85 times, and this combination reveals an in-orbit error rate of more than 45 years between SEFIs. Overall, results are promising for using RISC-V soft processors in new generations of FPGAs and aerospace missions.

Keywords: RISC-V. Rocket. NOEL-V. soft processor. SRAM-based FPGAs. fault tolerance. fault injection. SEE. heavy ions. protons.

Caracterização de tolerância à falhas de processadores RISC-V em FPGAs baseadas em SRAM com foco em missões aeroespaciais

RESUMO

Aplicações aeroespaciais, tais como pequenos satélites, exigem um certo nível de confiabilidade devido aos efeitos ionizantes de curto prazo ("Single Event Effects" (SEE)). Ao mesmo tempo, dispositivos comerciais prontos para uso (COTS) são frequentemente usados em missões no "New Space". Nesse contexto, o uso de processadores configuráveis implementados em COTS FPGAs (arranjo de porta programável em campo) baseados em SRAM requerem flexibilidade de projeto para aplicar as técnicas de tolerância à falhas mais adequadas para melhorar a confiabilidade do sistema. Distúrbios na memória de configuração do FPGA podem ser persistentes. Estes podem alterar a implementação arquitetural do processador, o que pode causar erros de fluxo de controle e cálculos incorretos. Esta tese visa caracterizar a suscetibilidade a SEE em processadores RISC-V embarcados em FPGAs baseados em SRAM e entender como a combinação de técnicas de tolerância à falhas pode reduzir significativamente a vulnerabilidade do sistema. A investigação aborda os problemas do uso de processadores configuráveis em ambientes propensos a SEE e as complexidades e impactos por trás dos métodos de mitigação. Os estudos de caso são os processadores RISC-V Rocket e NOEL-V de software livre e o RISCV NOEL-VFT comercial e tolerante à falhas. Os processadores estão embarcados no APSoC Zynq-7000 (28 nm CMOS), MPSoC Zynq UltraScale+ (16 nm FinFET) e Kintex UltraScale (20 nm CMOS), respectivamente. A proteção é aplicada no nível do projeto visando a memória de configuração do FPGA, núcleo do processador e memórias incorporadas. Combinações de técnicas de proteção são usas durante a investigação, tais como triplicação, varredura periódica ("scrubbing"), reiniciamento periódico, monitoramento de tempo e atualização de memória. Um supervisor de FPGA externo também foi desenvolvido com a finalidade de aumentar a cobertura à falhas, reduzir a chance de errors na interface de configuração e fornecer uma melhor visibilidade dos erros. A suscetibilidade a erros é avaliada sob injeção de falha e teste acelerado em solo. O processador Rocket alcançou 88% de corretude sob falhas simples, e a susceptibilidade sob radiação foi reduzida mais de 51 vezes usando uma combinação de técnicas de tolerância. Resultados do NOEL-V mostraram a eficácia em combinar métodos de proteção, reduzindo a susceptibilidade em 11 vezes. Tais técnicas aplicadas ao processador NOEL-VFT comer-

cial melhoraram a confiabilidade em 85 vezes, alcançando uma taxa de erros em órbita de mais de 45 anos entre falhas. De mode geral, os resultados são promissores para o uso de processadores RISC-V em novas gerações de FPGAs e missões aeroespaciais.

Palavras-chave: RISC-V. Rocket. NOEL-V. processadores configuráveis. FPGAs baseadas em SRAM. tolerância à falhas. injeção de falhas. SEE. ions pesados. prótons..

LIST OF ABBREVIATIONS AND ACRONYMS

AES Advanced Encryption Standard.

AHB AMBA High-performance Bus.

ALU Arithmetic-Logic Unit.

AMBA Advanced Microcontroller Bus Architecture.

APB Advanced Peripheral Bus.

APSoC All Programmable System-on-Chip.

ASIC Application Specific Integrated Circuit.

AVF Architectural Vulnerability Factor.

AXI Advanced Extensible Interface.

BCH Bose–Chaudhuri–Hocquenghem.

BGA Ball Grid Array.

BOOM Berkeley Out-of-Order Machine.

BRAM Block RAM.

CB Connection Block.

CE Correctable Error.

CGTMR Coarse Grain TMR.

CLB Configurable Logic Block.

CMOS Complementary Metal–Oxide–Semiconductor.

CMT Clock Management Tile.

COTS Commercial Off-The-Shelf.

CRAM Configuration memory RAM.

CRC Cyclic Redundancy Check.

CSR Control Status Register.

CVF Cache Vulnerability Factor.

DCCU Double-Cycle Corrected Upset.

DDR Double Data Rate.

DICE Dual Interlocked Cell.

DMR Dual Modular Redundancy.

DRAM Dynamic Random Access Memory.

DSP Digital Signal Processing.

DUT Device Under Test.

DWC Duplication With Comparison.

ECC Error Correction Code.

EDAC Error Detection And Correction.

EDDI Error Detection by Duplicated Instructions.

eFPGA embedded FPGA.

ESA European Space Agency.

FAR Frame Address Register.

FD-SOI Fully Depleted Silicon-On-Insulator.

FDTMR Fine Grain Distributed TMR.

FF Flip-Flop.

FFC Full Frame Check.

FFT Fast Fourier Transform.

FI Fault Injection.

FIFO First In First Out.

FinFET Fin Field-Effect Transistor.

FPGA Field Programmable Gate Array.

FPU Floating-Point Unit.

FT Fault Tolerance.

GEO Geostationary Orbit.

GPIO General Purpose Input/Output.

HDL Hardware Description Language.

HKMG High-K Metal Gate.

I/O Input/Output.

IC Integrated Circuit.

ICAP Internal Configuration Access Port.

IOD In-Orbit Demonstration.

IP Intellectual Property.

ISA Instruction Set Architecture.

IU Integer Unit.

KUS Kintex UltraScale.

LAB Logic Array Block.

LEO Low Earth Orbit.

LET Linear Energy Transfer.

LETeff Effective LET.

LIF Light Ion Facility.

LMB Local Memory Bus.

LTMR Fine Grain Local TMR.

LUT Look-Up Table.

MBU Multiple Bit Upset.

MCU Multiple Cell Upset.

MEBF Mean Executions Between Failures.

MFTF Mean Fluence to Failure.

MMC Memory Management Controller.

MMIO Memory-Mapped I/O (MMIO).

MMU Memory Management Unit.

MPSoC Multi-Processor System-on-Chip.

MTBF Mean Time Between Failures.

MTTF Mean Time to Failure.

MULXFx Wide-Function Multiplexes.

MWBF Mean Workload Between Failures.

MxM Matrix Multiplication.

NMR N-Modular Redundancy.

OBC On-Board Computer.

PC Program Counter.

PL Programmable Logic.

PLL Phase-Locked Loop.

PRNG Pseudo-Random Number Generator.

PS Processing System.

PULP Parallel Ultra-Low-Power.

Qsort Quicksort.

Rad-Hard Radiation Hardened.

RADNEXT RADiation facility Network for the EXploration of effects for indusTry and research.

RPP rectangular parallelepiped.

RTC Real Time Clock.

RTL Register Transfer Level.

S-SETA Selective Software-only Error-detection Technique using Assertions.

SB Switch Block.

SCCU Single-Cycle Corrected Upset.

SDC Silent Data Corruption.

SECDED Single Error Correction and Double Error Detection.

SEE Single Event Effects.

SEFI Single Event Functional Interrupt.

SEL Single Event Latch-up.

SEM-IP Soft Error Mitigation Intellectual Property.

SET Single Event Transient.

SEU Single Event Upset.

SIHFT Software-Implemented Hardware Fault Tolerance.

SMAP SelectMap.

SMT Simultaneous Multithreading.

SoC System-on-Chip.

SONOS Silicon-Oxide-Nitride-Silicon.

SPI Serial Peripheral Interface.

SRAM Static Random-Access Memory.

SRL Shift Register Logic.

SV Sensitive Volume.

SWIFT Software Implemented Fault Tolerance.

TC Test Controller.

TCM Tightly Coupled Memory.

TID Total Ionizing Dose.

TLR Thread-Level Redundancy.

TMR Triple Modular Redundancy.

TRIM Transport of Ions in Matter.

TSMC Taiwan Semiconductor Manufacturing Company.

UART Universal Asynchronous Receiver-Transmitter.

UAV Unmanned Aerial Vehicle.

UCL Université Catholique de Louvain.

UE Uncorrectable Error.

UNACE Unnecessary for Architecturally Correct Execution.

VHDL Very High Speed Integrated Circuit Hardware Description Language.

ZUS+ Zynq UltraScale+.

LIST OF FIGURES

Figure 2.1 Description of nanosatellites launches by type over the years. 35
Figure 2.2 Island-style FPGA architecture overview. 37
Figure 2.3 Artix-7 FPGA architecture overview. 38
Figure 2.4 Zynq-7000 APSoC functional blocks overview. 39
Figure 2.5 DSP48E1 slice overview. 43
Figure 2.6 FPGA power saving using clock gating. 47

Figure 3.1 Configuration memory RAM (CRAM) static cross section for Xilinx 28 nm CMOS Zynq-7000 and 16 nm FinFET Zynq UltraScale+ FPGAs under proton testing. 52
Figure 3.2 FPGA bits description by criticality. The essential bits are all bits used in the design implementation, and critical bits are bits that will lead to errors in case of upsets. 53
Figure 3.3 Example of SEE in soft processors 54

Figure 4.1 Power overhead on N-Modular Redundancy (NMR) systems as a function of the number of redundancies and ratio between FPGA dynamic and static power. 61
Figure 4.2 Analytical reliability of nonredundant and TMR systems. The TMR with repair plot is based on a theoretical Markov chain, considering an optimal repair rate without common mode failures. 62
Figure 4.3 Coarse Grain TMR (CGTMR) implementation. 64
Figure 4.4 Fine Grain Distributed TMR (FDTMR) implementation with feedback voters. 64
Figure 4.5 Cache data vulnerability related to different accesses: data read, data write, and cache eviction. 72
Figure 4.6 Cache vulnerability for different topologies: (a) varying instruction cache configuration with constant 16 KB data cache; (b) varying data cache configuration with constant 16 KB instruction cache. 74

Figure 6.1 Architecture description of the Rocket processor pipeline. 86
Figure 6.2 LowRISC SoC description. 86
Figure 6.3 Evaluation setup of LowRISC SoC implementation into Zynq-7000 AP-SoC. 88
Figure 6.4 Xilinx Zynq-7000 (XC7Z020-CLG484 part) microscopic view: (a) top surface, and (b) transversal section. 91
Figure 6.5 Heavy ion testing setup: (a) ZedBoard inside the vacuum chamber, and (b) view of the irradiation room. 92
Figure 6.6 Empiric reliability obtained by emulation fault injection for the unhardened Rocket soft processor running MxM, AES, and Qsort benchmarks at 20 MHz. 94
Figure 6.7 Empiric reliability obtained from the Oxygen irradiation of the unhardened Rocket soft processor running MxM benchmark at 20 MHz. 96
Figure 6.8 Empiric reliability obtained from the Oxygen irradiation of the unhardened Rocket soft processor without and with scrubbing running MxM benchmark at 20 MHz. 98

Figure 6.9 Empiric reliability obtained by emulation fault injection for the unhardened and CGTMR Rocket soft processor running MxM, AES, and Qsort benchmarks at 20 MHz. 101
Figure 6.10 Empiric reliability obtained from the heavy ion test campaign with the unhardened, CGTMR, and FDTMR Rocket soft processor running MxM benchmark at 20 MHz. 103
Figure 6.11 Total dynamic cross section and MWBF results from heavy ion testing for unhardened and protected Rocket soft processor running MxM benchmark at 20 MHz. 104
Figure 6.12 Empiric reliability obtained from the heavy ion testing for the unhardened and protected Rocket soft processor running MxM benchmark at 20 MHz. 106
Figure 6.13 Total dynamic cross section and MWBF results from heavy-ions experiments for the unhardened Rocket processor without and with scrubbing running Matrix Multiplication (MxM) benchmark at 20 and 50 MHz. 107
Figure 6.14 I+D data memory error rate (bars related to the left axis) and application execution time (line related to the right axis). 113
Figure 6.15 Error rate for the data memories per instruction and data cache over the different configurations. 115
Figure 6.16 Percentage of failures related to the application time window (in c.c.). .. 115
Figure 6.17 I+D control memory error rate per cache configuration. 117
Figure 6.18 Error rate for the control memories per instruction and data cache over the different configurations. 117
Figure 6.19 Total error rate for I+D data and control memories per cache configuration. 118
Figure 6.20 Total error rate (bars related to the left axis) and execution time (lines related to the right axis) per cache topology and flush period. 119
Figure 6.21 Error rate for the data memories per instruction and data cache over the different configurations and flush periods. 121

Figure 7.1 NOEL-V processor 7-stage dual-issue integer pipeline. 126
Figure 7.2 NOEL-V SoC implemented into the Zynq UltraScale+ FPGA. 127
Figure 7.3 Zynq UltraScale+ proton testing setup. 132
Figure 7.4 Reliability to failure as a function of accumulated injected faults. 136
Figure 7.5 Weibull fit of the reliability to failure as a function of accumulated injected faults. The Weibull parameters are described per design (shape; width). 136
Figure 7.6 Events per application execution until a permanent failure or the end of the test run. Results are presented per design and test board (BN1/BN2). 139
Figure 7.7 Total cross section per device with 95% confidence interval for 53 MeV proton testing. Results are presented per design and test board (BN1/BN2). 140
Figure 7.8 MTBF and MEBF are presented in bars related to the left axis, with the horizontal dashed lines highlighting the results for NV_1KB design as a reference. The red line shows the execution time, in seconds, related to the right axis. Results are presented per design integrated from both test boards. ... 142

Figure 8.1 Example of a system including GRSCRUB. 147
Figure 8.2 NOEL-VFT SoC implemented into the Kintex UltraScale FPGA. 150
Figure 8.3 NOEL-VFT test system. 151
Figure 8.4 Kintex UltraScale proton testing setup. 154

Figure 8.5 CRAM cross section per bit for the Kintex UltraScale XCKU060: GRSCRUB SCCU and DCCU, Bendel fit of GRSCRUB SCCU, and Bendel fit of the XCKU060 reference data. 157
Figure 8.6 BRAM cross section per bit for the Kintex UltraScale XCKU060: L1 cache CE, L2 cache CE and UE, Bendel fit of L1 and L2 caches CE, and Bendel fit of the XCKU060 reference data. 158
Figure 8.7 Total cross section per device with 95% confidence interval for different proton energies. 159

LIST OF TABLES

Table 2.1 Small satellite subcategories classified by mass (KULU, 2022). 34
Table 2.2 CubeSat size and mass defined by type (CPCL, 2018). 34
Table 2.3 Overview of ESA CubeSat missions (PEREZ, 2021). 35
Table 2.4 Comparative between hard and soft core processors. 40
Table 2.5 Soft processors architecture description. 41
Table 2.6 Resource usage of soft processors implemented in the Artix-7 FPGA.[1] 45

Table 4.1 MTTF comparative between non-redundant, TMR, and TMR with repair systems. 62

Table 6.1 Benchmark applications characteristics running on the Rocket soft processor. 89
Table 6.2 Ions description. 92
Table 6.3 Dynamic power and resource usage with the utilization percentage of the unhardened Rocket soft processor core and lowRISC System-on-Chip (SoC) implemented in the Zynq-7000 Programmable Logic (PL).[1,2] 93
Table 6.4 Emulation fault injection results for the unhardened Rocket soft processor per application benchmark at 20 MHz. 94
Table 6.5 Dynamic cross section and Mean Workload Between Failures (MWBF) results from heavy ion testing for the unhardened Rocket soft processor running MxM benchmark at 20 MHz. 95
Table 6.6 Dynamic cross section and MWBF results from heavy ion testing for the unhardened Rocket soft processor without and with scrubbing running MxM benchmark at 20 MHz. 98
Table 6.7 Dynamic power and resource usage with the utilization percentage of the unhardened, CGTMR, and FDTMR Rocket soft processor core and lowRISC SoC implemented in the Zynq-7000 PL.[1,2] 99
Table 6.8 Emulation fault injection results for the CGTMR Rocket soft processor per application benchmark at 20 MHz. 101
Table 6.9 Dynamic cross section and MWBF results from the heavy ion testing for the unhardened, CGTMR, and FDTMR Rocket soft processor running MxM benchmark at 20 MHz. 102
Table 6.10 Dynamic cross section and MWBF results from the heavy ion testing for the unhardened and protected Rocket soft processor running MxM benchmark at 20 MHz. 104
Table 6.11 Cross section comparison of soft processors implemented in SRAM-based FPGAs from this thesis and literature. 109
Table 6.12 Architecture of different Rocket L1 cache configurations implemented in the Zynq-7000. 111
Table 6.13 Normalized memory area used for fault injection targeting all L1 cache configurations. 112
Table 6.14 Emulation fault injection on the Rocket L1 instruction and data cache: targeting I+D data memories. 113
Table 6.15 Emulation fault injection on the Rocket L1 instruction and data cache: targeting I+D control memories. 116
Table 6.16 Emulation fault injection on the Rocket L1 instruction and data cache: target data memories protected with periodic flush. 119

Table 7.1 NOEL-V test designs description. 129

Table 7.2 NOEL-V test designs information: resource usage per processor core and entire SoC; number of essential bits (percentage of FPGA usage); and application execution time (in seconds). 133
Table 7.3 Fault injection result table describing the number of failures per design, the error rate with 95% confidence interval, and the mean faults to failure. 134
Table 7.4 Proton testing table describing actual flux and fluence by design per test board. 137
Table 7.5 53 MeV proton testing result table describing the observed failures and total cross section with 95% confidence interval (±10% fluence uncertainty). 137

Table 8.1 NOEL-VFT test designs description. 152
Table 8.2 NOEL-VFT test designs information: SoC resource usage and number of essential bits (percentage of FPGA usage). 155
Table 8.3 Proton testing result table describing the proton energies and fluence, observed failures, and total cross section. 156
Table 8.4 Orbital SEFI rate and MTTF of NOEL-VFT test designs. 161

Table 9.1 This thesis comparison against the state-of-the-art. 164
Table 9.2 Resource usage and tested frequency for Rocket (Zynq-7000), NOEL-V (Zynq UltraScale+), and NOEL-VFT (Kintex UltraScale). 166
Table 9.3 Fault injection results for Rocket (Zynq-7000) and NOEL-V (Zynq UltraScale+). 167
Table 9.4 Comparison between NOEL-V and NOEL-VFT dynamic failure cross section. Zynq UltraScale+ (ZUS+) 53 MeV are results from chapter 7, Kintex UltraScale (KUS) 62 MeV are results from chapter 8, and Kintex UltraScale (KUS) estimated are the computed cross section values. 169

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS 11
1 INTRODUCTION 27
1.1 Objectives and contributions 29
1.2 Structure of this thesis 32
2 USING SOFT PROCESSORS IN AEROSPACE MISSIONS 33
2.1 Aerospace missions overview 33
2.2 COTS SRAM-based FPGAs 36
2.3 Embedded soft processors 40
2.4 Soft processor implementation in FPGAs 42
2.4.1 Processor configurability 42
2.4.2 Arithmetic operations 43
2.4.3 Memories 43
2.4.4 Resource usage 44
2.4.5 Optimizations 45
2.4.6 Placement 46
2.4.7 Power consumption 46
3 SINGLE EVENT EFFECTS AND PROBLEM DEFINITION 49
3.1 Radiation-induced effects 49
3.1.1 Overview of SEE on integrated circuits 49
3.1.2 Soft errors in SRAM-based FPGAs 50
3.1.3 Soft errors in soft processors 53
3.2 Problem definition: the use of soft processors implemented into COTS SRAM-based FPGAs in SEE-prone environments 55
4 BACKGROUND AND RELATED WORKS 59
4.1 Background of fault tolerance techniques 59
4.1.1 Configuration memory scrubbing 59
4.1.2 Redundancy-based techniques 60
4.1.2.1 Modular redundancy 60
4.1.2.2 Error correction codes 63
4.1.2.3 Lockstep technique 65
4.1.3 Software-based techniques 66
4.1.4 Hardware monitors 66
4.1.5 Summary 67
4.2 Related works 67
4.2.1 Fault tolerance solutions for soft processors 67
4.2.2 L1 cache vulnerability 72
5 RADIATION CHARACTERIZATION METHODOLOGY 75
5.1 Single Event Effects testing 75
5.1.1 Accelerated ground testing 75
5.1.2 Emulation Fault injection 77
5.2 Evaluation metrics 79
5.2.1 Cross section 79
5.2.2 Fault injection error rate 80
5.2.3 SEU error rate estimation 80
5.2.4 Mean time to failure and mean time between failures 80
5.2.5 Mean fluence to failure 81
5.2.6 Mean executions and workload between failures 81
5.2.7 Empiric reliability 82

5.3 SEE error prediction 82
6 EXPLORING THE COTS RISC-V ROCKET SOFT PROCESSOR UNDER RADIATION EFFECTS 85
6.1 SEE characterization of the COTS Rocket soft processor 85
6.1.1 COTS RISC-V Rocket soft processor 85
6.1.2 Investigation methodology 87
6.1.2.1 Platform setup 87
6.1.2.2 Software benchmarks 88
6.1.2.3 Failure definition 89
6.1.2.4 Emulation fault injection 90
6.1.2.5 Heavy ion testing 91
6.1.3 Rocket soft processor under faults 92
6.1.3.1 Unhardened Rocket soft processor 92
6.1.3.2 Rocket soft processor protected by scrubbing 96
6.1.3.3 Rocket soft processor protected by TMR 98
6.1.3.4 Rocket soft processor protected by combined fault tolerance techniques 102
6.1.3.5 Influence of the processor frequency in the soft error susceptibility 106
6.1.3.6 Discussion 107
6.2 L1 cache susceptibility investigation 109
6.2.1 Investigation methodology 109
6.2.1.1 L1 cache configurations 110
6.2.1.2 Emulation fault injection 110
6.2.2 Results 112
6.2.2.1 Data memory error rate 112
6.2.2.2 Control memory error rate 116
6.2.2.3 Total error rate 117
6.2.2.4 Instruction cache protected by periodic flush 118
6.3 Discussion 122
7 EXPLORING THE COTS RISC-V NOEL-V SOFT PROCESSOR UNDER RADIATION EFFECTS 125
7.1 COTS RISC-V NOEL-V soft processor 125
7.2 Investigation methodology 126
7.2.1 Platform setup 126
7.2.2 Mitigating the COTS NOEL-V soft processor 127
7.2.2.1 Scrubbing 127
7.2.2.2 TMR for the central processing unit 128
7.2.2.3 L1 cache protection 128
7.2.2.4 Designs description overview 129
7.2.3 Software benchmark 130
7.2.4 Failure definition 130
7.2.5 Emulation fault injection 131
7.2.6 Proton testing 131
7.3 Results 132
7.3.1 FPGA resource usage and performance 132
7.3.2 Fault injection results 133
7.3.3 Proton testing results 135
7.3.3.1 Failure cross section 140
7.3.3.2 MTBF and MEBF results 141
7.3.4 Discussion 142

8 EXPLORING THE COMMERCIAL FAULT TOLERANT RISC-V NOEL-VFT SOFT PROCESSOR COMBINED WITH EXTERNAL FPGA SUPERVISOR UNDER RADIATION EFFECTS 145
8.1 RISC-V NOEL-VFT soft processor 145
8.2 External FPGA supervisor 146
8.2.1 System integration 147
8.2.2 Operational modes 147
8.2.2.1 Scrubbing operation 148
8.2.2.2 Configuration interface integrity check 148
8.3 Investigation methodology 149
8.3.1 Platform setup 149
8.3.2 Additional fault tolerance 151
8.3.3 Designs description overview 152
8.3.4 Software benchmark 152
8.3.5 Failure definition 153
8.3.6 Proton testing 154
8.4 Results 155
8.4.1 FPGA resource usage 155
8.4.2 Proton testing results 155
8.4.2.1 Configuration memory cross section 156
8.4.2.2 User memory cross section 157
8.4.2.3 Failure cross section 158
8.4.2.4 MTTF and error rate in-orbit 160
9 COMPARATIVE ANALYSIS BETWEEN SOFT PROCESSORS 163
9.1 Comparison to the state-of-the-art 163
9.2 Comparative results between processors 165
9.2.1 Resource usage comparison 165
9.2.2 FI comparison: Rocket vs. NOEL-V 166
9.2.3 Cross section comparison: NOEL-V vs. NOEL-VFT 167
10 CONCLUSIONS 171
10.1 Contributions 171
10.1.1 RISC-V Rocket soft processor investigation 171
10.1.2 RISC-V NOEL-V soft processor investigation 172
10.1.3 RISC-V NOEL-VFT soft processor investigation 173
10.1.4 Summary 174
10.2 Future works 174
REFERENCES 177
APPENDIX A — PUBLICATIONS 193

1 INTRODUCTION

Aerospace missions involve atmospheric, in-orbit, and outer space applications. Satellites are expressive space applications with hundreds of launches yearly (UNOOSA, 2022). Small satellites, such as CubeSats, present a restricted area and power constraint due to their size. This thesis is motivated by the ascending increase of small satellite launches in the past few years and the frequent use of Commercial Off-The-Shelf (COTS) components in such New Space missions.

In this scenario, processors are usually required to perform complex operations while maintaining a low energy budget and restricted size, which sometimes need customization of the processor. Soft processors are synthesizable hardware models of processor cores. The primary benefits of using soft processors are the possibility of customization to meet predefined requirements and the flexibility to address design changes during development. Several soft processors have been developed over the last decades. They implement different Instruction Set Architectures (ISAs) and differ in complexity, ranging from simple microcontroller implementations to complex multi-core System-on-Chips (SoCs). Some examples of soft processors currently available at the academic level, i.e., free-of-charge, are: LEON line cores (SPARC ISA) from Frontgrade Gaisler (FRONTGRADE GAISLER, 2022b); Cortex-M0, Cortex-M1, and Cortex-M3 (ARM ISA) cores from ARM (YIU, 2019); MicroBlaze (RISC ISA) from Xilinx (XILINX, 2021a); and a plethora of soft processors that implement the RISC-V ISA, such as the Rocket (ASANOVIĆ et al., 2016), NOEL-V (FRONTGRADE GAISLER, 2022b), and PULP (PULLINI et al., 2019).

Soft processors can be synthesized to Field Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASICs). The implementation in FPGAs offers many advantages to aerospace systems compared to ASICs. Reconfiguration is a primary benefit since the soft processor characteristics can change in-orbit. FPGAs are highly used in satellites and a variety of space missions (GARDENYES, 2012; NASA, 2020; HARDGROVE et al., 2020), which further motivates its use. Nonetheless, these devices may require a high power consumption that needs to be considered during the device selection to meet the mission's power budget. Another advantage of soft processors in FPGAs is allowing the evaluation of the mission requirements trade-offs such as resource utilization, architecture, performance, and cost.

Aerospace systems are prone to radiation-induced errors. Soft processors synthesized into COTS Static Random-Access Memory (SRAM)-based FPGAs are susceptible to Single Event Effects (SEE) at device and design levels. At the device level, Single Event Upsets (SEUs) may occur in the FPGA Configuration memory RAM (CRAM) and its embedded memories (WIRTHLIN, 2015). SEUs in the CRAM can be persistent and may change the architectural implementation of the soft processor, which can cause control-flow errors, leading to Single Event Functional Interrupts (SEFIs), and wrong computations, defined as Silent Data Corruptions (SDCs). These faults can be corrected by reconfiguration. At the design level, SEUs and Single Event Transients (SETs) can have complex effects on the processor architecture and its executing software (Quinn, 2014). Such effects impact processors by modifying values stored in memory elements, leading to data-flow errors and SDCs or causing control-flow errors and SEFIs. In general, these events can be cleared out by resetting the design.

The SEE sensitivity of SRAM-based FPGAs depends on the SRAM cells' technology. The susceptibility to upsets is related to the intrinsic characteristics of the transistor's physical structure. For instance, Complementary Metal–Oxide–Semiconductor (CMOS) is known as having higher SEU sensitivity than Fin Field-Effect Transistor (FinFET) (AZIMI et al., 2022). The fabrication process of the memory cells also impacts SEU sensitiveness. Different cell processes on the FPGA CRAM and Block RAM (BRAM) memories lead to distinct radiation sensitivity (TONFAT et al., 2017). In some FPGAs, the BRAM cross section can be twice the CRAM cross section (MAILLARD et al., 2019). Nevertheless, due to the expressive amount of bits, the CRAM has a higher probability of upsets leading to design failures.

Aerospace applications demand a certain level of reliability due to the SEE. Therefore, one must also evaluate the system's behavior in the presence of faults. A typical qualification methodology is performing accelerated ground testing and emulation fault injection. Depending on the mission requirements, fault mitigation needs to be employed to allow the correct operation in a SEE-prone environment. However, applying fault tolerance techniques may increase the system's resource usage and power consumption, and reduce performance. A system with area or power limitations may restrict the feasibility of complete protection.

Due to the complexity of its components, each part of the soft processor requires appropriate fault tolerance methods. Triple Modular Redundancy (TMR) is usually applied to the central processing unit for error masking, while the user memories can be

protected using Error Correction Codes (ECCs). However, system limitations in terms of area usage and power consumption may interfere with the feasibility of fully protecting the design. Applying fault tolerance techniques to a soft core processor synthesized into SRAM-based FPGAs might be challenging due to the complexity of the design and the FPGA characteristics. For instance, the number of FPGA available resources is a constraint when applying a TMR to the embedded system. Low-density FPGAs might not have enough resources for triplicating the entire SoC. Additionally, TMR increases the probability of faults in the design due to the increase in resource usage. Repair methods, such as scrubbing the CRAM and adding TMR feedback voters, further enhance the error masking capability. In this context, using soft processor-based SoCs might require design flexibility to select suitable techniques for critical parts, which can lead to reduced area and minimal power penalties.

1.1 Objectives and contributions

The goal of this thesis is to characterize the SEE susceptibility of RISC-V soft processors embedded in SRAM-based FPGAs. Our mission is to understand how employing combined fault tolerance techniques can significantly improve system reliability under soft errors. For that goal, this thesis aims at performing a deep investigation through accelerated SEE ground testing, using heavy ion and proton particles, and emulation fault injection to understand the radiation effects on RISC-V soft processors implemented in SRAM-based FPGAs, focusing on the FPGA configuration memory, processor core, and embedded memories. The work provides extensive novel data on these scenarios and discusses their benefits and trade-offs.

The discussion addresses the following questions:

1. A high vulnerability of the FPGA configuration memory is expected, but will it be the root of failures? Would well-known fault tolerance techniques be suitable for such complex processor SoCs? Will the scrubbing method be enough to protect the system? Besides scrubbing, is redundancy an appropriate choice?
2. How does the employment of fault tolerance techniques impact the system? Is it feasible to protect the entire SoC? What are the limitations? Would applying fault tolerance to only a few components be relevant to save resources while improving reliability?

3. How does the L1 cache memory affect the overall SEE susceptibility? Would the cache topology affect the reliability?

RISC-V soft processors have recently emerged as suitable for computing-intensive applications due to their performance-enabler architecture and reduced instructions set complexity, allowing designers to customize both processor cores and compilers. A more customized soft processor facilitates the adjustments on resource usage and power consumption if needed. Additionally, RISC-V soft processors are expected to be used in future space applications (DI MASCIO et al., 2019; ROGENMOSER; TORTORELLA, 2022). As an example, the Trisat-R nanosatellite, launched in July 2022, includes a Sky-Labs NANOhpm On-Board Computer (OBC) featuring a RISC-V NOEL-V processor (ESA, 2022; COX, 2022). For those reasons, this thesis performs the investigation on RISC-V soft processors. The selected RISC-V cores are the Rocket processor, well-known in the literature, and the cutting-edge NOEL-V processor.

This thesis contributes to the community by addressing the problems and challenges of employing SEE protection on RISC-V soft processors embedded in SRAM-based FPGAs implemented in different technologies, understanding the feasibility of using RISC-V-based processors in space, and performing the SEE characterization under heavy ion and proton testing, which are the most relevant particles present in-orbit, such as Low Earth Orbit (LEO) and Geostationary Orbit (GEO). The SEE characterization data gathered on cutting-edge RISC-V soft processors on the latest generations of SRAM-based FPGAs will provide guidance to the community for future aerospace missions.

Initially, we explore the FPGA configuration memory susceptibility to SEE through emulation fault injection and heavy ion testing. The case study scenario consists of the COTS open source Rocket processor embedded in a Xilinx Zynq-7000 All Programmable System-on-Chip (APSoC), whose FPGA fabric is based on 28 nm CMOS technology. Combined techniques such as TMR, scrubbing, periodic reset, and watchdog are used during the investigation. Further analysis is performed on the processor's L1 cache to demonstrate its impact on the soft processor failure rate, evaluating different cache topologies (size and associativity) under emulation fault injection.

In sequence, this thesis performs the SEE characterization of the COTS open source NOEL-V soft processor in the Xilinx Zynq UltraScale+ Multi-Processor System-on-Chip (MPSoC), built on 16 nm FinFET technology. The qualification includes proton testing and emulation fault injection to confirm that strategies such as cache refreshing, scrubbing, TMR, and duplication with comparison are effective in mitigating soft errors.

This thesis also presents an investigation performed under proton testing on the commercial fault tolerant NOEL-VFT soft processor embedded in a Xilinx Kintex UltraScale FPGA, fabricated on 20 nm CMOS technology. To our knowledge, this thesis is the first work to publish the SEE characterization of the commercial fault tolerant NOEL-VFT processor. An external FPGA supervisor targeting the Kintex UltraScale is developed to program end scrub the FPGA through the SelectMap interface. The FPGA supervisor aims to increase fault coverage, reduce the chance of soft errors in the scrubbing interface, and provide better visibility of upsets in the FPGA configuration memory.

Finally, this thesis includes a roughly comparison of the COTS open source Rocket and NOEL-V soft processors and the commercial fault tolerant NOEL-VFT soft processor. Results are compared to the state-of-the-art works. The assessment explores how employing fault tolerance impacts the system and improves the SEE sensitiveness.

In summary, the main contributions of this thesis lie in the following:

- We demonstrate the vulnerability to permanent and transient faults and how mitigation techniques can be combined to improve the reliability of COTS RISC-V soft processors – Rocket and NOEL-V – in COTS SRAM-based FPGAs built over different technologies – CMOS and FinFET. The investigation lies on the Rocket soft processor embedded in the Zynq-7000 APSoC, based on 28 nm CMOS technology, and the open source NOEL-V soft processor implemented in the Zynq UltraScale+ MPSoC at 16 nm FinFET node.
- We investigate how the L1 cache configuration (size and associativity) impacts the Rocket soft processor reliability through emulation fault injection targeting the BRAMs of the Zynq-7000 APSoC.
- An SEE characterization of the commercial fault tolerant NOEL-VFT is performed in a Kintex UltraScale FPGA, built on 20 nm CMOS technology. An external FPGA supervisor is also developed. The supervisor is responsible for FPGA programming and scrubbing via the SelectMap interface to improve the CRAM fault coverage.
- An in-depth analysis and discussion on the scenarios above, exploring benefits and trade-offs and helping guide future development in the field.
- Overall, we explore different combinations of fault tolerance techniques to improve the soft processors' reliability and assess their impact on the system by performing SEE testing under heavy ion and proton beams, which are the most relevant particles for in-orbit space missions.

1.2 Structure of this thesis

This thesis is organized as follows:

- Chapter 2: Presents the motivation of this thesis, describing aerospace mission characteristics; introduces the concepts of SRAM-based FPGAs and soft processors; and describes the steps for implementing soft processors in FPGAs.
- Chapter 3: Surveys the radiation-induced effects on devices, focusing on SRAM-based FPGAs and soft processors; and overviews the problems of implementing soft processors in SRAM-based FPGAs in SEE-prove environments.
- Chapter 4: Gives background of fault tolerance solutions; and overviews relevant and state-of-the-art works related to this thesis.
- Chapter 5: Presents a qualification methodology for system evaluation, SEE ground testing, evaluation metrics, and estimation of orbit error rate.
- Chapter 6: Investigates the SEE susceptibility of the RISC-V Rocket soft processor implemented in a Zynq-7000 APSoC; discusses the limitation on applying fault tolerance solutions in the design; evaluates the system under emulation fault injection and heavy ion testing. Additionally, the influence of the L1 cache topology on processor susceptibility is assessed by emulating faults in user memories.
- Chapter 7: Explores the SEE susceptibility of the RISC-V NOEL-V processor synthesized into the SRAM-based Zynq UltraScale+ FPGAs under proton testing and emulation fault injection.
- Chapter 8: Presents the SEE characterization under proton testing of the commercial fault tolerant RISC-V NOEL-VFT soft processor implemented in a Kintex UltraScale FPGA combined with the extra protection of an external FPGA supervisor.
- Chapter 9: Performs a comparison between this thesis and the state-of-the-art works; presents a high-level comparison between the soft processors results.
- Chapter 10: Presents the conclusions of this thesis and final remarks.
- Appendix A: Presents the publications from the Ph.D. student.

2 USING SOFT PROCESSORS IN AEROSPACE MISSIONS

This chapter presents an overview of aerospace mission characteristics, primarily focusing on small satellite exploration. The concepts of SRAM-based Field Programmable Gate Arrays (FPGAs) and soft processors are introduced in sequence. Finally, the steps for implementing soft processors in FPGAs are described, discussing the configurability, resource usage, and power consumption.

2.1 Aerospace missions overview

Aerospace missions involve atmospheric, in-orbit, and outer space applications. It covers a broad range of applications, such as interplanetary missions, satellites, Unmanned Aerial Vehicles (UAVs), and tactic drones.

Satellites are one of the most expressive space applications. Hundreds of satellites are launched every year, and the numbers tend to grow (UNOOSA, 2022). Nowadays, small satellites are responsible for most of these launches. The miniaturization of conventional satellites brought many advantages and made small satellites a good alternative for space exploration. Being cost a primary concern, the size of small satellites make them far more affordable, which has opened the market for new private and civilian applications. The launching costs are also directly affected by the satellite size, stimulating the increase of constellations. At the same time, the use of Commercial Off-The-Shelf (COTS) devices in space missions has gained popularity mainly due to performance requirements (ESPOSITO et al., 2015), which has also helped to reduce the mission costs (LAPPAS; KOSTOPOULOS, 2020). In 2019, about 20% of the components in European Space Agency (ESA) satellites were COTS, and it is expected an increase in the usage for future missions (NIKULAINEN, 2019).

Many opportunities emerge from small satellites exploration. Some examples are:

- Educational proposes.
- Validation and test of technologies - In-Orbit Demonstration (IOD).
- Space environment observation.
- Solar system explorations.

Small satellites are limited to 500 kg maximum, and can be classified in different subcategories, as described in Table 2.1 (KULU, 2022). As a note, the mass definition may vary depending on the source, as example, NASA (2017) defines a maximum weigh of 300 kg for small satellites.

Nanosatellites are a dominant type of small satellites launches. The most well-known subtype of nanosatellite is the CubeSat, which is defined by its standard size and weight. CubeSats vary in size based on the standard 10 cm unit (defined by U). $1U$ CubeSat is an approximated cube with dimensions of $10 \times 10 \times 11.3$ cm (CPCL, 2018). The CubeSat type is defined by its length. Table 2.2 gives some examples of CubeSat types, described by size and mass. Figure 2.1 shows the announced nanosatellites launches by type over the years. Different CubeSat types are described from $0.25U$ to $16U$, being $3U$ the most predominant.

Table 2.1 – Small satellite subcategories classified by mass (KULU, 2022).

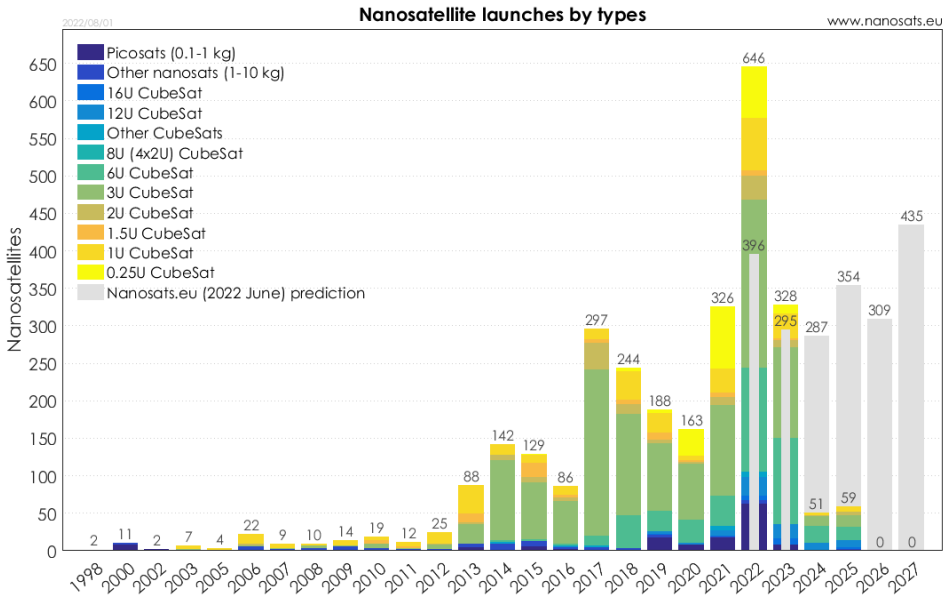
Small satellite type	**Mass**
Minisatellite	100 kg to 500 kg
Microsatellite	10 kg to 100 kg
Nanosatellite	1 kg to 10 kg
Picosatellite	100 g to 1 kg
Femtosatellite	10 g to 100 g
Attosatellite	1 g to 10 g
Zeptosatellite	0.1 g to 1 g

Table 2.2 – CubeSat size and mass defined by type (CPCL, 2018).

CubeSat size	**Size (cm)**	**Mass (kg)**
1U	$10 \times 10 \times 11.3$	< 1.33
1.5U	$10 \times 10 \times 17$	< 2
2U	$10 \times 10 \times 22.7$	< 2.66
3U	$10 \times 10 \times 34$	< 4

Size is not the only restriction on those applications. The power source usually relies on solar panels and limited batteries. Solar panels supply the satellite power and charge the batteries while illuminated directly by the Sun. When the Earth shadows the satellite, the batteries need to sustain the energy. The shadow interval varies depending on the satellite's orbit (ISMAIL et al., 2015). Due to the restricted size of the solar panels, CubeSats are limited to a small power budget. Therefore, the solar panels' power production and battery capacity must be well stipulated based on the mission requirements.

Figure 2.1 – Description of nanosatellites launches by type over the years.



Source: (KULU, 2022).

Table 2.3 – Overview of ESA CubeSat missions (PEREZ, 2021).

Mission	CubeSat size	Power (max)	Launch
XATCOBEO	1U	2 W	2012
GOMX-3	3U	6 W	2015
GOMX-4B	6U	12 W	2018
GOMX-5	12U	100 W	2022
SCOUT-1	3x12U	30 W	2023
MILANI	6U	60 W	2024

A common approach is periodically setting the satellite to power-saving mode to recover the batteries (NIETO-PEROY; EMAMI, 2019). Another strategy is the limitation of the payload's power consumption, such as shutting down unused circuits and pre-defining an execution sequence.

Table 2.3 depicts an overview of the ESA CubeSat missions with launching dates from 2012 to 2024 (PEREZ, 2021). The presented data show the low-power requirements of those missions, with maximum power ranging from 2 W (in a $1U$) to 100 W (in a $12U$). Although the maximum power consumption may be more flexible by increasing the CubeSat size, the mission goal and available resources define the restrictions.

Size and power constraints in CubeSats make the employment of high-computing processing systems challenging. The On-Board Computer (OBC) and payload are restricted to a small power budget and area, which conditions the selection of components

and circuitry. In this context, the employment of FPGAs in those missions requires an assessment of their power consumption and the energy availability of the system. The study of Arnold, Nuzzaci and Gordon-Ross (2012) estimates the power reserve budget related to the payload operational time in different orbits for CubeSats with integrated FPGAs. The power budget depends on the CubeSat size ($1U$ and $3U$ tested), the solar panels' production, the size of batteries, and the payload subsystem.

FPGA are highly used in satellites and a variety of space applications. Gardenyes (2012) showed the percentage of usage of FPGA in different ESA missions, which could reach up to 75% of programmable devices in a single mission. FPGAs can be part of many workloads. For instance, in the Mars2020 mission Perseverance rover (NASA, 2020), FPGAs are used in the radar transceiver, navigation systems, motor controllers, and computer vision applications. As another example, an FPGA is used to acquire data from a photomultiplier tube and monitor time and temperature in a lunar exploration CubeSat mission (HARDGROVE et al., 2020).

At the same time, the use of soft processors implemented in FPGAs is attractive for such applications primarily due to flexibility and a short development process. The following sections address the concepts of FPGAs, focusing on Static Random-Access Memory (SRAM)-based devices, and soft processors. Aerospace applications are particularly susceptible to radiation effects or Single Event Effects (SEE), further described in chapter 3. This thesis devotes its attention to SEE mitigation methods on soft processors embedded in FPGAs and their possible limitations and impacts on the system.

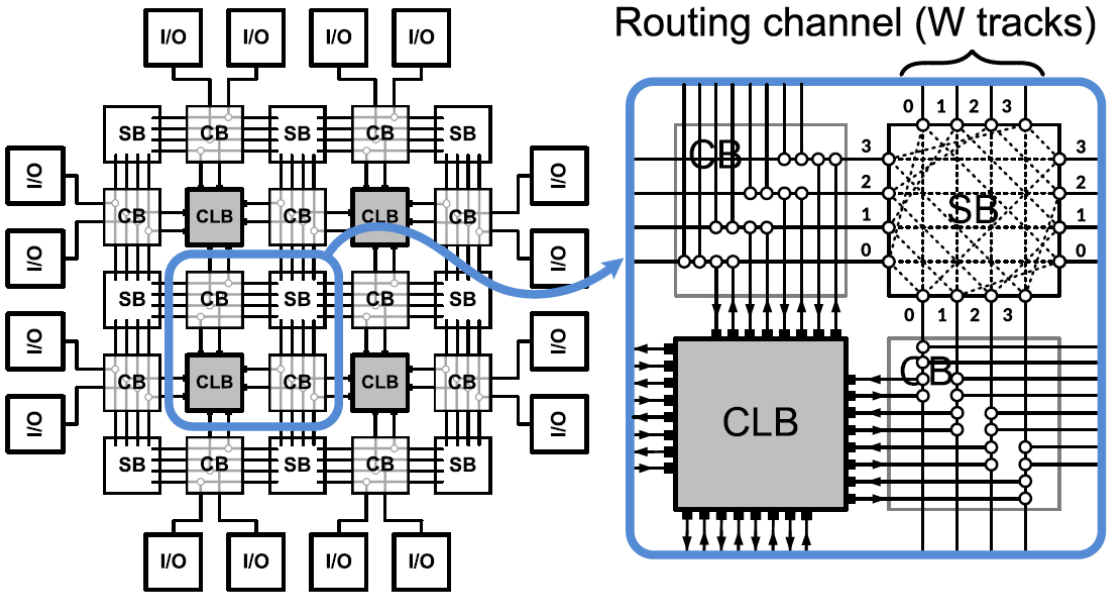
2.2 COTS SRAM-based FPGAs

FPGAs are programmable devices that can be configured to implement different circuitry. The designer can describe the hardware behavior via Hardware Description Language (HDL), such as Very High Speed Integrated Circuit Hardware Description Language (VHDL) and Verilog.

The FPGA programmability is provided through configurable elements that store the user-defined configuration. Logic blocks, also described as Configurable Logic Blocks (CLBs) - for Xilinx FPGAs (XILINX, 2021d) - or Logic Array Blocks (LABs) - for Intel Altera FPGAs (ALTERA, 2011), are used to implement arbitrary digital logic. Most of commercial FPGAs are based in Look-Up Tables (LUTs) to compute logical operations and Flip-Flops (FFs) as storage elements (HAUCK; DEHON, 2008). The logic block

inputs and outputs are connected to routing tracks via Connection Blocks (CBs), while the Switch Blocks (SBs) are programmable switches or multiplexes responsible for setting the route. Figure 2.2 presents an overview of a generic island-style FPGA architecture. In addition to the basic structure (i.e., CLBs, CBs, SBs, and routing tracks), FPGAs usually include other resources such as Input/Output (I/O) blocks, clock management system – clock control and Phase-Locked Loop (PLL), Block RAMs (BRAMs), and Digital Signal Processing (DSP) blocks. Each FPGA vendor presents a specific architecture differing in structure and available resources.

Figure 2.2 – Island-style FPGA architecture overview.



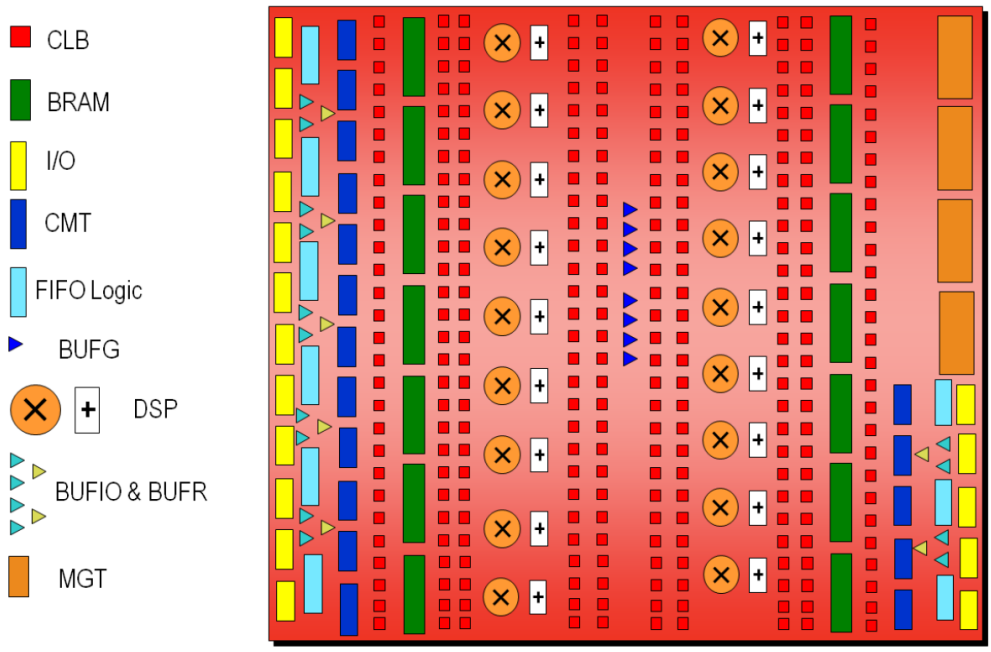
Source: (QIN et al., 2018).

The FPGA configuration is programmed using a bitstream – a bit file that contains the mapping of all programmable bits in the FPGA fabric. The FPGA configuration memory may be implemented in the following cell architectures: antifuse, flash memory, Silicon-Oxide-Nitride-Silicon (SONOS), and SRAM cells. Antifuse-based FPGAs have write-once memory elements, flash-based and SONOS are reconfigurable and non-volatile, and SRAM-based presents a Configuration memory RAM (CRAM), which is reprogrammable but volatile. SRAM-based FPGAs are the most widely used due to the high reconfiguration flexibility, competitiveness costs, and capability of integrating complex systems on the same component (HAUCK; DEHON, 2008).

The Xilinx 7-series FPGA family (XILINX, 2020) is a set of SRAM-based FPGAs built over 28 nm Complementary Metal–Oxide–Semiconductor (CMOS) technology, targeting different application requirements, such as high-performance and low-power. The

7-series family includes Spartan-7, Virtex-7, Kintex-7, and Artix-7 FPGAs. In general, these FPGAs differ in the logic resources density, memory availability, and performance. Figure 2.3 shows the Artix-7 FPGA architecture. The Programmable Logic (PL) is composed of CLB slices, 36 Kb dual-port BRAMs, DSPs, and Clock Management Tiles (CMTs). The CLBs present two slices (XILINX, 2016a). Each slice comprises of four 6-input LUTs, eight FFs, Wide-Function Multiplexes (MULXFx), and carry logic. Some slices also have memory capability and can be used as distributed RAM or Shift Register Logic (SRL).

Figure 2.3 – Artix-7 FPGA architecture overview.

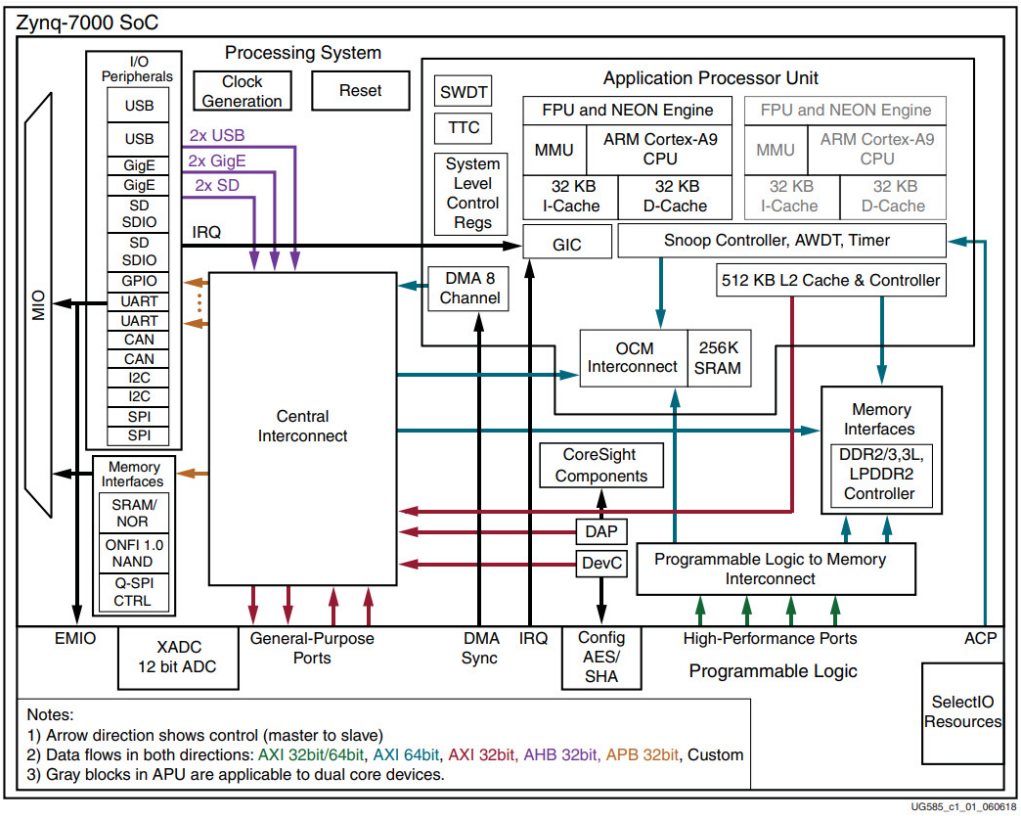


Source: (XILINX, 2013).

Over the years, FPGA devices increased in complexity, and even more System-on-Chips (SoCs) based on heterogeneous architectures were introduced in the market. Heterogeneous systems include a PL combined with embedded hard core processors, memories, and high-speed peripherals. An example of All Programmable System-on-Chip (APSoC) is the Xilinx Zynq-7000 (XILINX, 2021d) that combines the PL, same Artix-7 FPGA layer, with a Processing System (PS), SoC around a dual-core ARM Cortex-A9 processor. Figure 2.4 presents an overview of the Zynq-7000 functional blocks.

Modern SRAM-based FPGAs, such as the Xilinx UltraScale family, implemented on 20 nm CMOS technology, include high-performance and high-density devices (XILINX, 2022c). SRAM-based FPGAs are also fabricated using Fin Field-Effect Transistor (FinFET) and Fully Depleted Silicon-On-Insulator (FD-SOI) technologies. The Xilinx

Figure 2.4 – Zynq-7000 APSoC functional blocks overview.



Source: (XILINX, 2021d).

UltraScale+ architecture includes FPGAs built over 16 nm FinFET technology (XILINX, 2022c). The Lattice CertusPro-NX (LATTICE, 2021) is a low-power FPGA family based on 28 nm FD-SOI technology.

When used in radiation-prone environments, FPGAs are susceptible to SEE (further described in chapter 3). Radiation Hardened (Rad-Hard) FPGAs provide intrinsic hardness to SEE, such as Dual Interlocked Cell (DICE); Error Detection And Correction (EDAC); clock tree duplication; and a built-in scrubber. An example of Rad-Hard FPGA is the NanoXplore NG-ULTRA based on 28 nm FD-SOI technology (NANOXPLORE, 2022). The primary drawbacks of Rad-Hard FPGAs are the excessive price and lower performance compared to modern COTS FPGAs. COTS FPGAs may also be suitable for aerospace applications when fault tolerance methods are applied (further described in section 4). This thesis focuses its investigation on COTS SRAM-based FPGAs.

2.3 Embedded soft processors

Soft processors are synthesizable hardware models developed in HDL, such as VHDL and Verilog. The customized processor Intellectual Property (IP) description may be synthesized to either an FPGA or a dedicated Application Specific Integrated Circuit (ASIC). In the ASIC version, the processor is hardwired physically into the chip, where all the components are integrated and manufactured. In this case, the processor is called hard core since it no longer can be customized. Table 2.4 presents a comparative overview between hard and soft core processors based on the works performed by Mondragón-Torres and Christman (2012), and Jayakrishnan and Parikh (2019). The processor implementation in FPGA provides flexibility to address changing design requirements during the development process. Other advantages of soft processors are a more straightforward integration, portability, and low cost compared to buying or manufacturing an ASIC. However, hard cores are usually more energy efficient and achieve better performance (MONDRAGóN-TORRES; CHRISTMAN, 2012).

Table 2.4 – Comparative between hard and soft core processors.

	Power consumption	**Performance**	**Flexibility**	**Time to market**	**Cost**
Hard core	Low	High	Low	High	High
Soft core	Middle / High	Low	High	Low	Low

Source: (MONDRAGóN-TORRES; CHRISTMAN, 2012; JAYAKRISHNAN; PARIKH, 2019).

During co-processing, a processor may use an FPGA custom logic as a hardware accelerator to improve the system's capabilities and performance. For a hard core processor (ASIC implementation), the co-processing can be performed with an FPGA chip on the same PCB as the processor. Modern devices can have a hybrid architecture, including an embedded FPGA (eFPGA) within the same chip as the hard core processor. The co-processing can also be implemented internally to the FPGA with a soft processor and hardware accelerators synthesized into the FPGA fabric.

Several soft processors have been developed in the last decades. They implement different Instruction Set Architectures (ISAs) and differ in complexity, ranging from simple microcontroller implementations to complex multicore SoCs. Some examples of soft processors available in the open community are: LEON3 (SPARC ISA) (FRONTGRADE GAISLER, 2022b); Cortex-M3 (ARM ISA) (ARM, 2005); MicroBlaze (RISC ISA) (XILINX, 2021a); and a plethora of soft processors implementing the RISC-V ISA, such as the Rocket (ASANOVIĆ et al., 2016), NOEL-V (FRONTGRADE GAISLER, 2022b),

and Ibex (previously zero-riscy) (LowRISC, 2018) that is part of the Parallel Ultra-Low-Power (PULP) platform (PULLINI et al., 2019). Table 2.5 gives an overview of the architecture of those soft processors, describing the ISA, pipeline, on-chip memory, and presence of Floating-Point Unit (FPU).

Table 2.5 – Soft processors architecture description.

Soft processor	ISA	Pipeline	On-chip memory	FPU
MicroBlaze[1]	32-bit RISC	3, 5, or 8-stage single issue	LMB[6] / Cache (opt.)	Opt.
Cortex-M3[2]	32-bit ARMv7-M	3-stage single issue	TCM[7] (opt.)	No
LEON3[3]	32-bit SPARC v8	7-stage single issue	Cache (opt.)	Opt.
NOEL-V[3]	32 or 64-bit RISC-V	7-stage single/dual issue	Cache	Opt.
Rocket[4]	32 or 64-bit RISC-V	5-stage single issue	Cache	Opt.
Ibex[5]	32-bit RISC-V	2, or 3-stage single issue	Inst. cache (opt.)	No

1. (XILINX, 2021a)
2. (ARM, 2005)
3. (FRONTGRADE GAISLER, 2022b)
4. (ASANOVIĆ et al., 2016)
5. (LowRISC, 2018)
6. Local Memory Bus (LMB) interface to optional 2 KB local memory (fixed size)
7. Tightly Coupled Memory (TCM)

Many applications have been using soft processors. Examples range from web servers (AMIN et al., 2011), real-time video systems (ATITALLAH et al., 2005), and avionic, defence, and space systems (ANDERSSON et al., 2017). RISC-V-based processors have been used in a wide variety of terrestrial applications, and there is an emerging interest in using those processors also in future space missions (DI MASCIO et al., 2019; ROGENMOSER; TORTORELLA, 2022). For example, the Trisat-R nanosatellite, launched in July 2022, includes a SkyLabs NANOhpm OBC featuring a RISC-V NOEL-V processor (ESA, 2022; COX, 2022).

The RISC-V ISA is a standard open architecture, highly extensible, and flexible that allows the development of general-purpose computing systems (WATERMAN et al., 2014). The standard was established for the design implementation not to be architecture- or technology-dependent. For those reasons, this thesis focuses its investigation on RISC-V Rocket and NOEL-V soft processors.

To support the reader about the concepts and configurations exercised in this thesis, an overview of the implementation of soft processors in FPGAs is presented in the following section.

2.4 Soft processor implementation in FPGAs

When a soft processor IP is synthesized to an FPGA logic, the components and the netlist of signals and interconnections are mapped to the resources of the target FPGA. Every FPGA vendor's tool has its own implementation flow, which, in general, is defined by the following (XILINX, 2021c):

- translating the incoming netlist and design constraints to internal primitives;
- mapping the logic into the FPGA elements; and
- placing and routing the design components and interconnections considering the timing constraints.

After the implementation phase, the FPGA programming file is generated (i.e., bitstream generation), and the FPGA device can be configured.

The designer may customize and constrain the soft processor to achieve the requirements of a target system or mission. This section further discusses the design possibilities when implementing a soft processor in FPGAs.

2.4.1 Processor configurability

Configurability is the capability of adding, removing, or changing resources that implement a specific feature in a processor (LEIBSON, 2006). The amount of configurable parameters varies depending on the soft processor. Examples of customized features usually available are: cache size and configuration; enable/disable FPU; the use of debug unit; memory management; allow exceptions and interruptions; and add or remove peripherals. When the soft processor's source is available, the designer may update the code based on their needs. Such adaptability facilitates the employment of techniques aiming for better performance, lower resource usage, or fault tolerance. However, adapting a processor source faces many challenges and complexities. LEON3, NOEL-V, and Rocket are examples of open source soft processors. Although Cortex-M3 and MicroBlaze are freely accessible, the source of these soft processors is unavailable, and the configurability is limited to a few features.

Some soft processors can also provide internal configuration optimizations. For instance, the MicroBlaze supports three different pipeline optimizations (XILINX, 2021a):

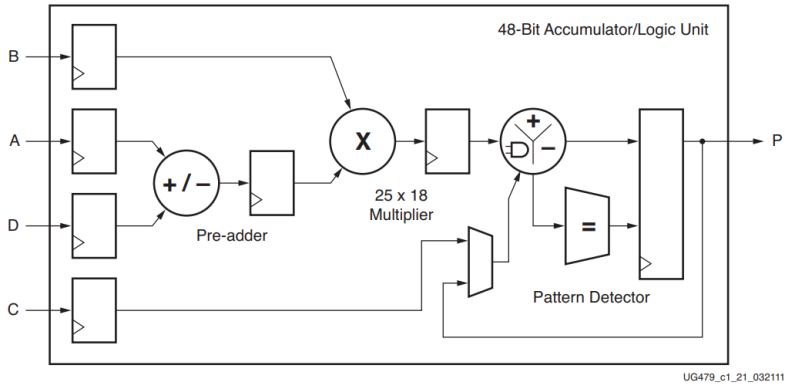
- Area: 3-stage pipeline with minimum execution units enabled.
- Performance: 5-stage pipeline for better performance.
- Frequency: 8-stage pipeline to reach higher frequencies.

LEON3, NOEL-V, Rocket, and Cortex-M3 do not support pipeline optimizations.

2.4.2 Arithmetic operations

Arithmetic operations, such as additions, subtractions, multiplications, divisions, and accumulations, may be implemented in the FPGA logic, DSPs, or combined. The designer may set constraints to the soft processor implementation to force or not use DSPs for specific operations. Otherwise, the FPGA vendor's tool freely infers DSP slices based on default configurations. The DSP slices may also implement logic operations, such as AND/NAND, OR/NOR, XOR/XNOR, and NOT. Figure 2.5 shows the dedicated elements of a DSP48E1 slice in Xilinx FPGAs (XILINX, 2018).

Figure 2.5 – DSP48E1 slice overview.



Source: (XILINX, 2018).

Specifying the DSP usage directly in the design (i.e., direct instantiation) usually leads to reduced resource usage and higher operating frequency (RONAK; FAHMY, 2012). Another advantage of using DSPs instead of combinational implementation is the higher robustness against SEE (OLIVEIRA et al., 2019).

2.4.3 Memories

Soft processors present several memory elements, including the main memory, caches, register file, First In First Out (FIFO), and buffers. The main memory usually

requires a larger area and can be implemented externally in a Double Data Rate (DDR) Dynamic Random Access Memory (DRAM), for instance, but it can also be featured as an on-chip RAM, depending on the SoC requirements.

External memories present a high access latency, requiring many clock cycles from the request until receiving the data, while the internal BRAMs have a great performance advantage. Bansal et al. (2018) assessed the memory subsystem in a Xilinx Zynq Ultrascale+ MPSoC, comparing the PL access latency to internal BRAM and external DRAM. The authors found that the latency for the external memory can be up to 1.5 (i.e., 75 ns) more than the internal BRAM.

Cache memories are much smaller and with faster access than the main memory and are usually implemented in the internal BRAMs. The cache latency is about 3 ns in the referenced work (BANSAL et al., 2018). Small memory elements such as register files, FIFOs, and buffers can be inferred as BRAMs but also as FFs and distributed RAMs (LUTRAMs).

2.4.4 Resource usage

Each FPGA has different available resources. High-density FPGAs, such as the Xilinx Virtex UltraScale+ VU29P with more than 1.7M LUTs (XILINX, 2022c), can implement large designs. Low-density FPGAs, such as the Xilinx Spartan-7 XC7S6 including less than 4K LUTs (XILINX, 2020), have capacity for small designs only. The designer must evaluate the area requirements of the system and select a suitable FPGA with available resources. As recommended, the design should use a maximum of 75% FPGA resources since a larger area increases the complexity of placement and timing closure (XILINX, 2021b).

The resource usage of the soft processor IP is provided by the FPGA vendor's tool, and it is usually described in terms of LUTs, FFs, carry logic, BRAMs, and DSPs. Moreover, I/Os, routing, clocking, and peripherals are essential resources to be verified. High-density FPGAs are usually costly, and the designer might not have the flexibility to select the target FPGA. Hence, if the target FPGA does not feature enough resource availability, the design area may be reduced by removing components, reducing memory size, and enabling optimizations. On the other hand, if the design requires just a small area and most of the FPGA resources are not used, selecting a smaller device would avoid wasting unnecessary power.

Table 2.6 – Resource usage of soft processors implemented in the Artix-7 FPGA.[1]

Soft processor[2]		Resource usage[5]					
		FF	**LUT**	**MUXFx**	**Carry**	**BRAM**	**DSP**
MicroBlaze[3]	Area	4,109	4,380	34	64	10	3
	Perf.	4,458	4,772	110	76	10	3
	Freq.	5,914	6,295	5	161	10	3
LEON3		2,143	5,211	68	69	13	4
NOEL-V		9,927	24,142	669	897	22	4
Rocket		8,875	13,279	257	462	13	4
Cortex-M3		5,680	13,757	220	258	6	3

1. FPGA vendor's tool: Xilinx Vivado 2021.1; Target FPGA: Xilinx Artix-7 (xc7a100tcsg324-1).
2. Default configuration of soft processor IPs, except: 8 KB cache/TCM; no FPU; integer multiplier and divider enabled; 50 MHz.
   The Rocket soft processor is the 64-bit version from the lowRISC implementation.
   The NOEL-V soft processor is the single-issue 32-bit version with a lite configuration.
3. MicroBlaze internal optimizations (different pipeline): area (3-stage); performance (5-stage); and frequency (8-stage).
5. Estimated resource usage per soft processor.

Table 2.6 presents the resource usage of the MicroBlaze, Cortex-M3, LEON3, NOEL-V, and Rocket soft processors implemented in the Xilinx Artix-7 FPGA. For generality, the soft processor IPs were synthesized with few adjusts from the default configurations, such as: 8 KB cache/TCM enabled; no FPU; integer multiplier and divider enabled; and 50 MHz clock frequency. The resource usage described in table 2.6 considers the soft processors' default top-level module but does not include information of an entire SoC.

2.4.5 Optimizations

Optimization is a design enhancement towards a specific goal, such as improving performance, reducing area, or saving power. The designer may manually constraint the design for particular optimizations during the synthesis or implementation phase. Moreover, the FPGA vendor's tool usually provides automatic strategies for design optimization. For instance, the Xilinx Vivado tool supports different strategies targeting power, resource usage, placement, routing, or timing (XILINX, 2021c).

2.4.6 Placement

During the implementation phase, the placement stage is the distribution of the defined logic blocks to the FPGA fabric. The FPGA vendor's tool uses placement algorithms to reduce the delay of critical paths and optimize the design routing (GROVER; K.SONI, 2012).

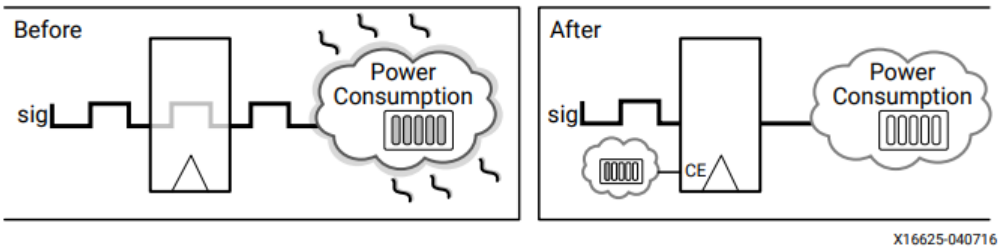
The designer can condition the soft processor design to be placed in a specific area of the FPGA. Partial blocks of the design can also be placed in different regions. This approach is usually helpful to avoid mixing or reusing resources in the same area, such as when implementing design replication. However, restricting the placement might lead to timing issues or have lack of available resources in the area. If no placement constraints are applied, the FPGA vendor's tool will freely place the design in the FPGA fabric, optimizing paths and resources as predefined.

2.4.7 Power consumption

The power consumption in devices is established by a result of static and dynamic power. The static power is determined by transistors leakage, and the dynamic power is a result of switching activity, operating frequency, capacitance charging, and supply voltage (GROVER; K.SONI, 2012). The fabrication technology of the FPGA device affects the leakage current and resources capacitance. FPGAs usually present higher power consumption than ASICs due to the high number of programmable bits. The extra circuitry not used also significantly contributes to the static power consumption. Moreover, the higher the junction temperature, the higher the transistor leakage and, therefore, the higher the static power (KLEIN, 2005). The implemented design directly impacts the dynamic power by resource usage and operating frequency, whereas it affects the static power by the amount of unused circuitry.

Clock gating is an effective approach to reduce power consumption. The input clock of inactive elements is temporarily turned off, avoiding unnecessary switching and saving power. Figure 2.6 describes a clock gating implementation. The gating is controlled by an additional logic that should present low power consumption. One should notice that applying clock gating may affect the performance of the design. Enabling power implementation optimizations from the FPGA vendor's tool is a straightforward method for automatic clock gating generation. For instance, the *Power_ExploreArea* strategy of

Figure 2.6 – FPGA power saving using clock gating.



Source: (XILINX, 2021c).

the Xilinx Vivado tool combines area optimization with clock gating (XILINX, 2021c). Additional system level design techniques may also be applied to soft processors to reduce dynamic power. Dimond, Mencer and Luk (2006) proposed a framework, combining instruction recording and power-aware scheduling methods, that achieved a mean dynamic power reduction of 28.37% for VirtexIIPro and 9.31% for Spartan3 FPGA.

3 SINGLE EVENT EFFECTS AND PROBLEM DEFINITION

This chapter brings an overview of the radiation-induced effects on devices, focusing on SRAM-based FPGAs and soft processors, and discusses the problems of implementing soft processors in SRAM-based FPGAs in SEE-prove environments.

3.1 Radiation-induced effects

Electronic devices in space missions are particularly susceptible to Single Event Effects (SEE) caused by ionized particles from the space environment (NICOLAIDIS, 2011). Radiation particles mainly originate from solar activity, cosmic rays, and trapped particles. Space mission devices are not the only systems affected by radiation. Terrestrial radiation is also a concern due to secondary particles generated by the collisions of the primary particles with the air molecules in the atmosphere (NORMAND, 2001; BAUMANN, 2005).

This section presents an overview of the radiation-induced effects on Integrated Circuits (ICs) and further describes the soft error characteristics in SRAM-based FPGAs and soft processors.

3.1.1 Overview of SEE on integrated circuits

Single Event Effects (SEE) are a consequence of the interaction of radiation particles with the IC's material. The interaction of energetic ions with silicon can occur by direct or indirect ionization, which can produce charge collection in the ion track (BAUMANN, 2005). The Linear Energy Transfer (LET) is characterized as ionization due to the particle energy lost in the penetration path (VELAZCO; FAURE, 2007). The particle LET increases with penetration and the maximum loss occurs at the Bragg peak (BUCHNER et al., 2011). The minimal charge affecting a Sensitive Volume (SV) is defined as a critical charge. The charge deposition can be enough to disturb a circuit node and change a logic state, leading to SEE. There are different subcategories of SEE. This thesis follows the primary definitions established in the ESCC25100 standard (ESA, 2014).

Single Event Upset (SEU) is a single ion strike that flips a logic cell, also called upset or bit-flip. Multiple Cell Upset (MCU) is classified when a single particle affects multiple cells. If the cells are from the same word, it is define as Multiple Bit Upset (MBU). SEUs, MCUs, MBUs are soft errors, which means non-destructive events that can be recovered by rewriting or resetting the cell.

Single Event Transient (SET) is a transient pulse generated at a susceptible transistor node. For instance, this pulse may propagate through the logic and be captured by a flip-flop. SET is a soft error that can be logical, electrical, or latch window masked (BAUMANN, 2005).

Single Event Functional Interrupt (SEFI) occurs when a soft error leads to device malfunction, usually only recovered by reset or power cycle. SEFIs are common in complex systems since soft errors affecting control bits or state machines might lead to hangs or crashes in the device.

Single Event Latch-up (SEL) is a potentially destructive state resulting from a transistor parasitic effect that leads to a high draining current and, consequently, high operating current consumption. A micro latch-up event is defined when there is a stepwise increase instead of the high-current phenomena (TAUSCH et al., 2007). SEFIs might also result in a step rise in the current consumption.

Radiation may also lead to degradation effects in the integrated circuits. The Total Ionizing Dose (TID) refers to the amount of accumulated radiation-induced trapped charge in the transistors gate oxides (OLDHAM; MCLEAN, 2003). TID degradation effects may cause loss of functionality and higher current consumption, mainly due to higher leakage currents. Particles may also lead to atomic displacement due to the kinetic energy transferred via nonionizing events (WARNER et al., 2005). The total nonionizing dose deposited in a material is defined as displacement damage.

3.1.2 Soft errors in SRAM-based FPGAs

SRAM-based FPGAs are particularly susceptible to soft errors due to the inherent characteristic of SRAM cells. Soft errors in SRAM-based FPGAs can lead to transient or permanent errors (KASTENSMIDT et al., 2004; ASADI; TAHOORI, 2005). Transient errors result from soft errors on dynamic memory elements, such as FFs, BRAMs, and LUTRAMs. A SET pulse on the combinational logic can propagate and be loaded by an FF, or SEUs can directly flip bits in memory cells. The data stays temporarily erroneous

until the memory element is rewritten. This bit error might lead to a system failure, be masked by design, or be corrected by fault tolerance techniques.

Permanent errors occur when soft errors affect the FPGA CRAM, changing the RTL architectural implementation of the design. SEUs in LUTs can change the logic function the LUT is implementing. Bit-flips in the routing system can enable or disable switches, change paths, or create or destroy interconnections. The FPGA clock tree can also be affected by soft errors, leading to failures. Additionally, some CRAM bits control the functionality of other elements, for instance, whether a LUT is a logic function, a shift register, or a LUTRAM (ASADI; TAHOORI, 2005). When a control bit is flipped, the element under its control might change functionality, and multiple other bits can also be inverted due to the newly defined function (BERG et al., 2008). CRAM errors are persistent and can only be fixed by reconfiguration.

The soft error sensitivity of SRAM-based FPGAs is directly affected by the technology of the SRAM cells. For instance, a study performed by Azimi et al. (2022) using proton testing showed that the Xilinx 28 nm CMOS Zynq-7000 is one order of magnitude more sensitive to SEUs than the 16 nm FinFET Zynq UltraScale+. The difference in susceptibility to upsets is due to the intrinsic characteristics of the transistor's physical structure (i.e., planar bulk CMOS and 3D FinFET).

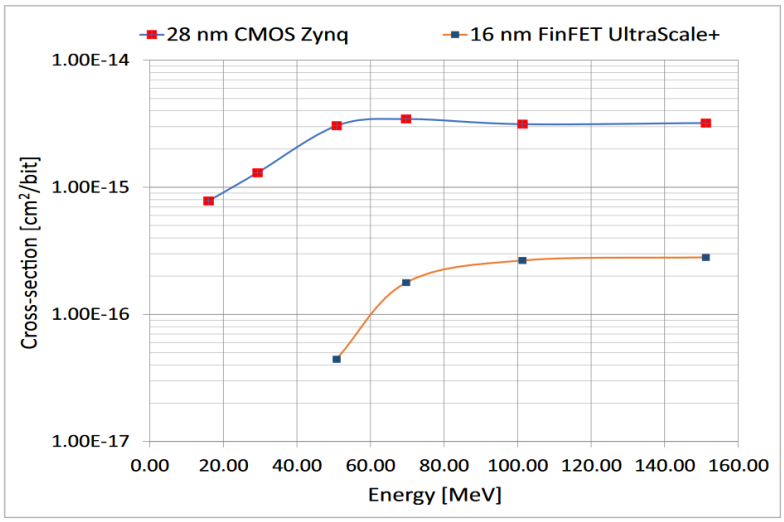
The static cross section[1] evaluates the intrinsic radiation sensitiveness of the FPGA resources. Figure 3.1 shows a comparative between the CRAM SEU static cross section in the 28 nm CMOS Zynq-7000 and 16 nm FinFET Zynq UltraScale+ FPGAs under proton testing (AZIMI et al., 2022).

The SEU sensitivity of the FPGA CRAM and BRAM bits can also be different. Although the SRAM cells are implemented in the same technology, the fabrication process of the memory cells may differ. The fabrication process can target high-performance or low-power characteristics by varying parameters such as the SRAM cell area and transistor threshold voltage. Different cell processes on the FPGA CRAM and BRAM memories directly impact the radiation sensitivity (TONFAT et al., 2017). For instance, the SEU static cross section of the BRAM bits is about twice the CRAM bits in the Xilinx Kintex UltraScale FPGA under heavy ion testing (MAILLARD et al., 2019).

The static cross section of the FPGA resources is considered as the radiation sensitivity worst-case scenario. Nevertheless, SEUs in the FPGA will not necessarily lead to system failures due to the design's inherent masking effects. The dynamic cross section

[1]The definition of cross section and the differences between static and dynamic tests will be further addressed in this chapter.

Figure 3.1 – CRAM static cross section for Xilinx 28 nm CMOS Zynq-7000 and 16 nm FinFET Zynq UltraScale+ FPGAs under proton testing.



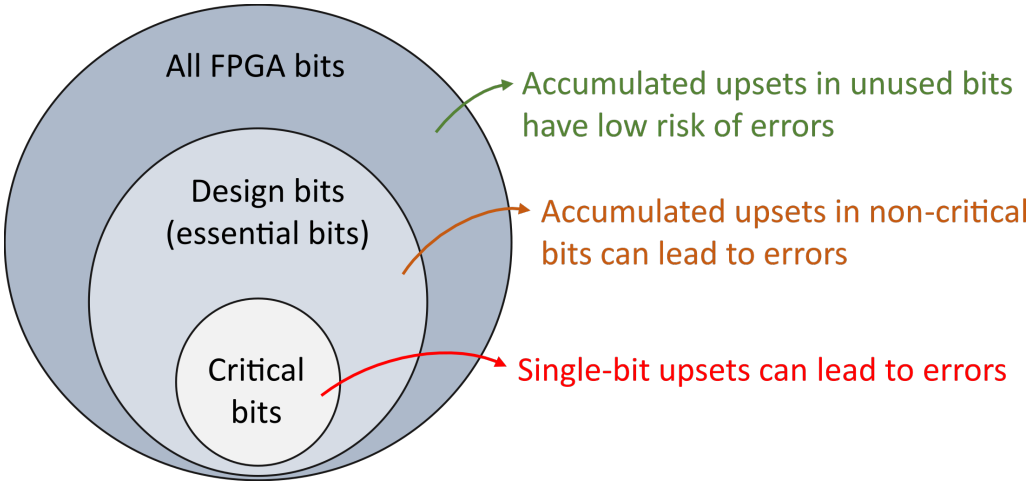
Source: (AZIMI et al., 2022).

is design-dependent and considers the system failures experienced at the user level, such as output data errors and functional interrupts.

The dynamic cross section is expected to be lowered from the static cross section by the design's inherent masking variable. Employing fault tolerance techniques can increase the error masking effects and reduce the dynamic cross section of the design. Chapter 4 describes examples of fault tolerance techniques. The design implementation rules can also affect the dynamic cross section. For instance, implementing arithmetic operations on DSP slices instead of the combination logic reduces the design susceptibility since DSPs are more resilient to soft errors (SANCHEZ-ELEZ et al., 2016; TONFAT et al., 2017; OLIVEIRA et al., 2019).

Figure 3.2 describes the soft error criticality of the FPGA bits to lead to design failures. Upsets in the unused FPGA area have a reduced risk of leading to errors since those bits are not expected to interfere with the design functionality. However, it is not removed the possibility of the design being affected if unused elements get enabled due to soft errors. All bits configured due to the design implementation (either zeros or ones) are called essential bits. Single-bit upsets in essential bits do not have any effects, but accumulated upsets might provoke errors since the design's inherent masking effects can be overcome. Critical bits are defined as bits that can cause errors if flipped. Therefore, a single-bit upset in a critical bit can lead to a system failure. Typical unmitigated designs implemented in Xilinx SRAM-based FPGAs are expected to present between 5% and 10% of critical bits (XILINX, 2022a). In a worst-case scenario, one in ten upsets is expected

Figure 3.2 – FPGA bits description by criticality. The essential bits are all bits used in the design implementation, and critical bits are bits that will lead to errors in case of upsets.



Source: From the author.

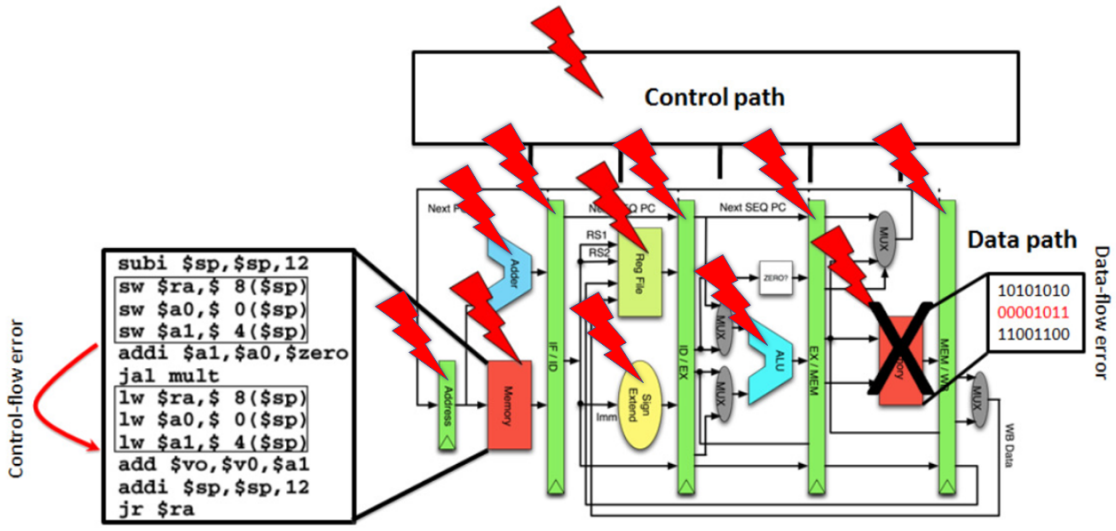
to lead to a system failure.

3.1.3 Soft errors in soft processors

Soft errors affecting processors may lead to data flow or control flow errors, resulting in incorrect application execution or system crashes. Figure 3.3 shows some examples of the soft processor susceptible areas. SEUs and SETs in the embedded memories can have complex effects on the processor architecture and its executing software (Quinn, 2014). The processor data flow is affected by modifying values stored in memory elements, leading to corrupted computations classified as Silent Data Corruptions (SDCs).

Soft errors in the processor control flow may lead to SEFIs and result in application crashes or processor hangs. Examples of control flow errors are the branch errors, such as the creation or deletion of a branch or incorrect branch decisions. The erroneous branch creation occurs when a non-branch instruction is set into a valid branch instruction due to a bit-flip. This fail leads the program flow to a wrong address. On the other hand, a faulty branch deletion occurs when a branch instruction is converted into a different instruction, and the original branch is not taken. A bit-flip in a conditional branch may result in an incorrect decision (i.e., if the branch should be taken or not). Moreover, a soft error can modify a branch instruction's address, assigning a wrong address to the program execution. If the Program Counter (PC) register is upset, the following executed instruction changes, leading the program flow to an incorrect address.

Figure 3.3 – Example of SEE in soft processors



Source: Adapted from (TAMBARA, 2017).

SEE may result in different processor errors, depending on the hardware unit affected. Some errors related to the affected processor element are described below (VELAZCO; FAURE, 2007).

- **Register File:** Upsets may corrupt data, provoking SDCs. On the other hand, SEUs affecting a control register can lead to errors in the execution flow and hangs in the system.
- **Integer Unit (IU), FPU:** SEUs in the pipeline of the arithmetic units may result in incorrect computations, leading to SDCs.
- **Bus Unit:** Bit-flips in the embedded registers, which latches address and data, can lead to incorrect read or write operations.
- **Control Unit:** SEUs in the circuitry, which implements complex algorithms, may provoke exceptions or losses of sequence.
- **Debug Unit:** SEUs can trigger debug execution modes leading to errors in the program execution flow.
- **Instruction Cache:** Upsets may lead to SDCs or SEFIs. The instruction cache is usually divided into memory elements that store the fetched instructions and the tag array to validate or invalidate the fetched program. SEUs in the tag array can invalidate an instruction, leading to a cache miss. This type of fail introduces a delay in the program execution as the instruction has to be fetched again, but the

application would still finish the execution correctly. However, if an SEU validates an incorrect instruction, the program flow will crash. Moreover, SEUs can directly corrupt an instruction. Suppose the tag array validates an incorrect code. In that case, a wrong instruction can be executed, or an exception can be generated if the instruction is no longer in the processor instruction set. On the other hand, if the tag array does not validate the corrupted instruction, the fault is masked, and no incorrect behavior is observed.

- **Data cache:** SEUs may lead to SDCs or SEFIs. The data cache is divided into data memory and tag array. Bit-flips in the tag array may validate out-of-date data, leading to wrong outputs, or invalidate data, introducing a delay in the application due to a cache miss. If an upset affects the data memory, this may lead to a corrupted output. However, if the data is out-of-date, the fault is masked, and no effects are observed. Bit-flips in the data cache may also lead to SEFI depending on the application characteristics.

In addition to the effects described above, soft processors synthesized into COTS SRAM-based FPGAs are susceptible to persistent faults at the device level. SEUs in the FPGA configuration memory can lead to changes in the architectural implementation of the soft processor. These faults can cause SEFIs, leading to control flow errors, and SDCs, erroneous application output. Due to the large number of configuration bits required to implement a soft processor design, CRAM upsets are the most relevant cause of failures. However, soft errors in the embedded memories can also lead to different failures, as previously described. Moreover, clock line or routing circuitry faults can cause incorrect behavior in the soft processor.

3.2 Problem definition: the use of soft processors implemented into COTS SRAM-based FPGAs in SEE-prone environments

The primary concern when using soft processors embedded into COTS SRAM-based FPGAs is the processor's vulnerability to persistent errors in the configuration memory. The first step is to understand the mission requirements, which may have restrictions such as budget, hardware limitations, maximum allowed error rate susceptibility, and criticality. Based on those requirements, the most appropriate device shall be selected, if not already pre-established in the mission.

In the case of SRAM-based FPGAs, the CRAM sensitivity is affected by the technology of the SRAM cells, which are commonly based on CMOS, FinFET, or FD-SOI. The planar bulk CMOS is a technology more sensitive to soft errors due to the intrinsic characteristics of the transistor's physical structure (AZIMI et al., 2022). The SEU susceptibility of the implemented FPGA design can be estimated by combining the technology susceptibility information with the FPGA radiation sensitivity. The number of upsets in the configuration and user memories during the mission's lifetime can be estimated based on the static response of the FPGA fabric to SEUs (i.e., static cross section figures usually provided by the FPGA vendor). The dynamic response of FPGA design (i.e., dynamic cross section) depends on the masking effects of the implemented system. Larger designs use more FPGA resources and, therefore, present more sensitive bits. Depending on the system's characteristics, the design can have more or less critical bits that can lead to failures. The design susceptibility can be assessed through SEE ground testing using accelerated particles. Emulation fault injection is also a powerful strategy for reliability analysis.

Soft processors are complex designs that usually feature a complex state machine with a multi-stage pipeline, user memories, and arithmetic units. Soft errors can affect all these elements and lead to failures, such as SDCs and SEFIs. Additionally, soft processors in SRAM-based FPGAs are prone to architectural errors due to the FPGA configuration memory vulnerability. Protecting such a variety of elements is not straightforward and requires multiple mitigation solutions.

In that context, this thesis investigates the following problems:

- **Problem 1:** Not always the entire soft processor-based SoC can be triplicated due to restricted resources or due to the unfeasibility of triplication of connections and interfaces. In low-density FPGAs, the triplicated design might exceed the FPGA available resources and make it impossible to complete the placement and routing phases. In some cases, the placement is successful, but the design fails to meet timing with high connections slack in the critical path, or requires a reduction of the clock speed. Even in a fully triplicated design, replicating hardwired interfaces is impractical, which will mandatorily narrow the connections to a single line, increasing the chances of cross-domain faults. In those cases, high-level fault tolerance techniques can be employed combined with the Triple Modular Redundancy (TMR) for a more robust system monitoring.

- **Problem 2:** Embedded processors make use of on-chip cache memories to re-

duce memory access time and boost performance. Nonetheless, SEUs in the caches can also contribute to failures and increase the soft processor vulnerability (TAMBARA et al., 2015). The larger the cache size and the higher the number of cache levels (e.g., L1 and L2), the larger the susceptible area. Therefore, an increase in the soft processor susceptibility is expected. On the other hand, a soft processor with a larger cache will present a better performance, speeding up the application benchmark execution. A faster execution reduces the exposure vulnerability in time. Some techniques can be applied to protect cache memories, such as Error Correction Code (ECC), parity, and memory scrubbing. A periodic cache refresh can also be used as a mitigation strategy at the expense of performance overhead.

- **Problem 3:** Scrubbing is a mandatory protection of the FPGA configuration memory. Internal scrubbers are also implemented in the FPGA fabric and, therefore, are susceptible to the same effects as the FPGA design. Soft errors in the exposed scrubbing interface can compromise the correction capabilities and, in the worst case, lead to catastrophic impacts such as washing the entire configuration memory with erroneous data. External scrubbers may provide higher robustness to deal with soft errors since their hardware is not implemented in the FPGA fabric.

The fault tolerance techniques mentioned above are detailed in chapter 4. This thesis addresses those problems through investigations described in chapters 6, 7, and 8.

4 BACKGROUND AND RELATED WORKS

This chapter provides an overview of different fault tolerance techniques that can be applied to soft processors implemented in SRAM-based FPGAs. The following techniques can also be combined for system enhancement. The related works section describes the relevant studies and the state of the art of mitigation methods on soft processors and L1 cache vulnerability.

4.1 Background of fault tolerance techniques

4.1.1 Configuration memory scrubbing

SEUs in SRAM-based FPGAs may lead to persistent errors in the system, changing the architectural implementation of the design, as detailed in section 3.1.2. CRAM scrubbing is a well-known technique for coping with upsets in the configuration memory and avoiding fault accumulation. However, scrubbing does not prevent bit-flips from happening or error build-up due to faults on critical bits. Therefore, additional design-level mitigation techniques are recommended to decrease the number of single points of failure in the system and increase fault masking.

The literature presents several scrubbing implementations that mainly differ in error detection, power consumption, resource usage, and correction speed (HEINER; COLLINS; WIRTHLIN, 2008; BROSSER et al., 2014; TONFAT et al., 2015; STODDARD et al., 2017). Scrubbing can be defined as internal when the scrubber engine is embedded inside the FPGA being monitored and external when the scrubber controller is located externally to the target FPGA in a different component. The Xilinx Soft Error Mitigation Intellectual Property (SEM-IP) (XILINX, 2022b) is an example of internal scrubbing present in most Xilinx FPGAs.

The main advantage of internal scrubbers is the high speed for single error detection and correction. However, internal scrubbers are also susceptible to soft errors affecting the FPGA. Internal scrubbers can get locked and have the correction capability compromised due to faults in the scrubber interface or multiple errors in the configuration memory (BERG et al., 2008). In this context, external scrubbers usually provide higher robustness and the ability to deal with multiple errors.

The CRAM scrubbing does not protect dynamic memory elements, such as BRAMs, distributed memory (LUTRAMs), and FFs. SEUs affecting the dynamic elements can be mitigated by fault tolerance techniques such as redundancy or ECC, as further described later in this chapter. Moreover, a periodic reset may be required to reestablish the system and restore the FFs initial state. A power cycle might be necessary in cases of SEFIs in the FPGA internal control elements or configuration interface.

4.1.2 Redundancy-based techniques

Redundancy-based techniques are error mitigation strategies applied at information, temporal or spatial levels (OSINSKI; LANGER; MOTTOK, 2017). Redundancy at the information level consists of adding extra data to memory elements, such as ECC or data replication on different memory spaces. The temporal redundancy directly impacts performance since it requires additional execution time. For instance, multiple software executions on the same processor for data check. The replication of components is considered spatial redundancy. Some of those redundancy-based techniques are further described below.

4.1.2.1 Modular redundancy

The most basic redundancy is the Dual Modular Redundancy (DMR), or Duplication With Comparison (DWC), which consists of duplicating a component or software and adding a checker for error detection. In this case, the error is not masked or corrected but flagged for an external monitor.

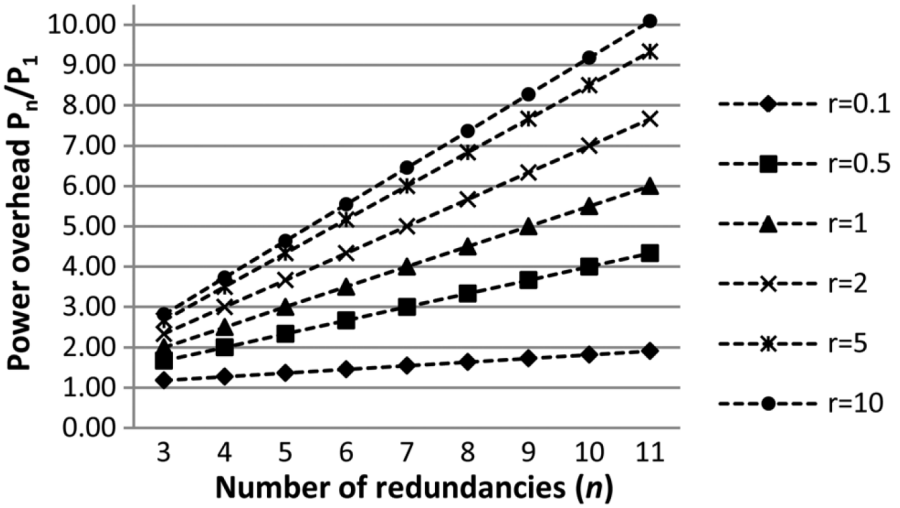
The Triple Modular Redundancy (TMR) is used for error detection and mitigation. Three redundant modules and a voter for comparison are used to mask single faults. The error build-up is avoided, but the faulty module is not corrected. Moreover, voters should also be triplicated to prevent single points of failure.

To improve the fault masking capability, the modular redundancy concept can be expanded to N replicas, N-Modular Redundancy (NMR). For instance, N software replicas can run on different cores in multi-core systems, or N component replicas can be instantiated in an FPGA design. These solutions offer higher fault detection and error masking at the expense of larger area, reduced performance, and power overhead.

On FPGAs, the high resource usage of redundant designs might affect the routing

and placement, and the clock frequency might need to be reduced to meet the timing requirements. Regarding power, Figure 4.1 shows the overhead for NMR systems as a function of *N* replicas and the ratio (r) between the FPGA dynamic and static power (TARRILLO et al., 2014). The overhead is given by the ratio between the power of *N* modules (P_n) and the power of the non-redundant module (P_1). As expected, the higher the redundancies, the greater the power penalties. The overhead is steeper with the increase of the ratio between the FPGA dynamic and static power. The FPGA dynamic power is design-dependent and is expected to increase with more redundancies.

Figure 4.1 – Power overhead on NMR systems as a function of the number of redundancies and ratio between FPGA dynamic and static power.



Source: (TARRILLO et al., 2014).

If upsets accumulate over time in the system, the masking effects of the modular redundancy can be overcome due to multiple faulty modules. Since the area of a replicated design is much higher than a non-redundant design, the former is more susceptible to faults. In fact, the Mean Time to Failure (MTTF)[1] in a TMR system without repair is worse than a non-redundant system (SIEWIOREK; SWARZ, 1998). The TMR elements should be frequently repaired for error correction. The higher the repair rate (μ), the more effective the TMR design and the higher the MTTF.

Table 4.1 describes the MTTF on different systems related to the failure and repair rates. Figure 4.2 shows the analytical reliability plot for non-redundant and TMR systems (SIEWIOREK; SWARZ, 1998). The TMR with repair plot is based on a theoretical Markov chain, considering an optimal repair rate and without common mode failures.

In SRAM-based FPGAs, scrubbing is essential to clean the CRAM upsets that

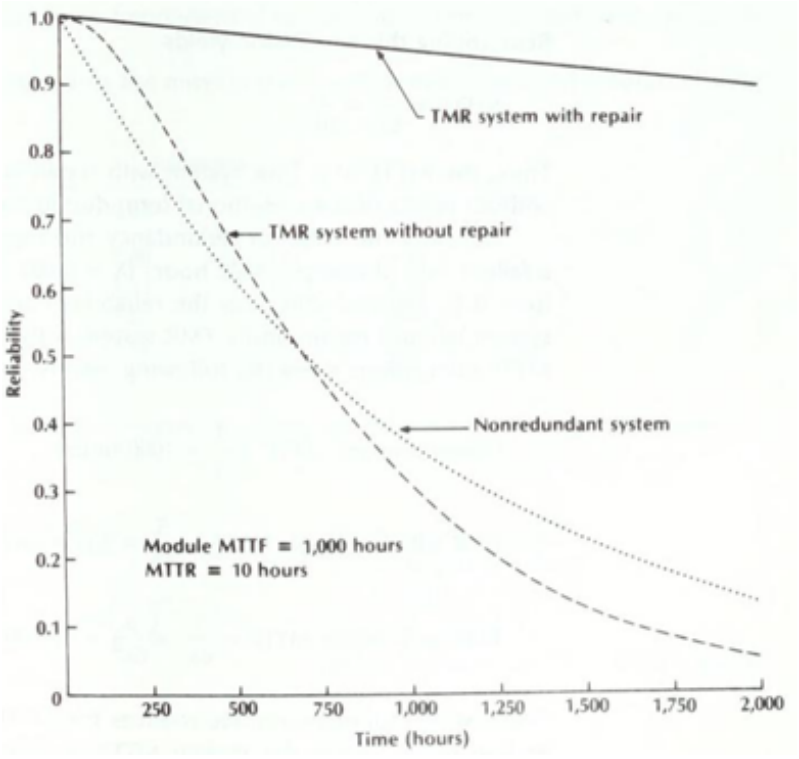
[1]The MTTF metric is described in section 5.2.4.

Table 4.1 – MTTF comparative between non-redundant, TMR, and TMR with repair systems.

	Nonredundant	**TMR**	**TMR with repair**
MTTF[1]	$\frac{1}{\lambda}$	$\frac{5}{6\lambda}$	$\frac{5}{6\lambda} + \frac{\mu}{6\lambda^2}$

1. MTTF defined in section 5.2.4.
(λ) failure rate defined per equation 5.7.
(μ) repair rate.

Figure 4.2 – Analytical reliability of nonredundant and TMR systems. The TMR with repair plot is based on a theoretical Markov chain, considering an optimal repair rate without common mode failures.



Source: (SIEWIOREK; SWARZ, 1998).

might compromise the TMR design. The scrubbing rate directly affects the TMR performance in masking faults. A system reset may also be required to restore the FFs state in the triplicated components. The scrubbing cycle and reset periodicity should be defined based on the SEE rate in the target environment.

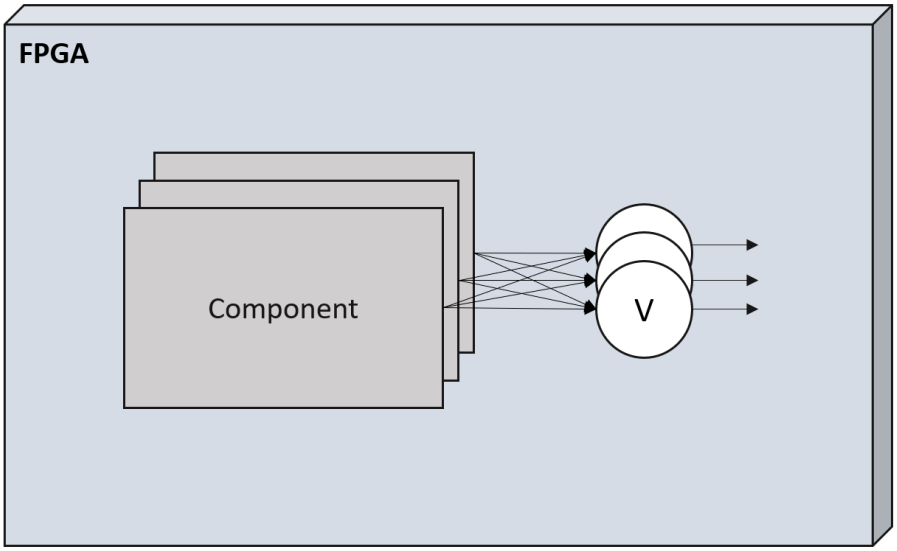
TMR is one of the most used mitigation techniques for SRAM-based FPGAs (KASTENSMIDT et al., 2005; QUINN et al., 2007). Component triplication can be applied at different granularity. Coarse Grain TMR (CGTMR), or block TMR, consists of triplicating the entire component or module as a block and voting only the external outputs, as shown in Figure 4.3. The CGTMR is the most straightforward TMR approach and can be implemented using a tool or manually, depending on the design complexity. In the Fine Grain Local TMR (LTMR), only the FFs are triplicated, and voters are added after each TMRed FF. A Fine Grain Distributed TMR (FDTMR) consists of triplicating all internal sub-modules. Feedback voters can be added to FFs with feedback logic for error correction. Figure 4.4 presents an example of FDTMR implementation. LTMR and FDTMR are more complex to implement and usually require an external tool for the netlist triplication. Examples of commercial and open source tools are the Synopsys Synplify Premier (SYNOPSYS, 2015a) and SpyDrNet TMR (BYU CONFIGURABLE COMPUTING LAB., 2020), respectively.

As mentioned above, the drawbacks of redundant designs are the area, performance, and power overheads. The LTMR leads to lower resource impact since only the FFs are mitigated. However, the LTMR is not efficient when implemented in SRAM-based FPGAs since the configuration memory is highly susceptible (OLIVEIRA et al., 2019). The LTMR approach is more effective when applied to antifuse-based FPGAs because the configuration memory in those FPGAs is inherently radiation-tolerant (BERG et al., 2006). The FDTMR requires the highest resource usage, with an overhead usually more than three times higher since all internal sub-modules are triplicated with multiple voters. The CGTMR is usually more area and power efficient than FDTMR (GROVER; K.SONI, 2012). The target FPGA size is also a limitation, and applying a CGTMR or FDTMR to the entire design is not always possible. Often, the triplication needs to be restricted to a few components.

4.1.2.2 Error correction codes

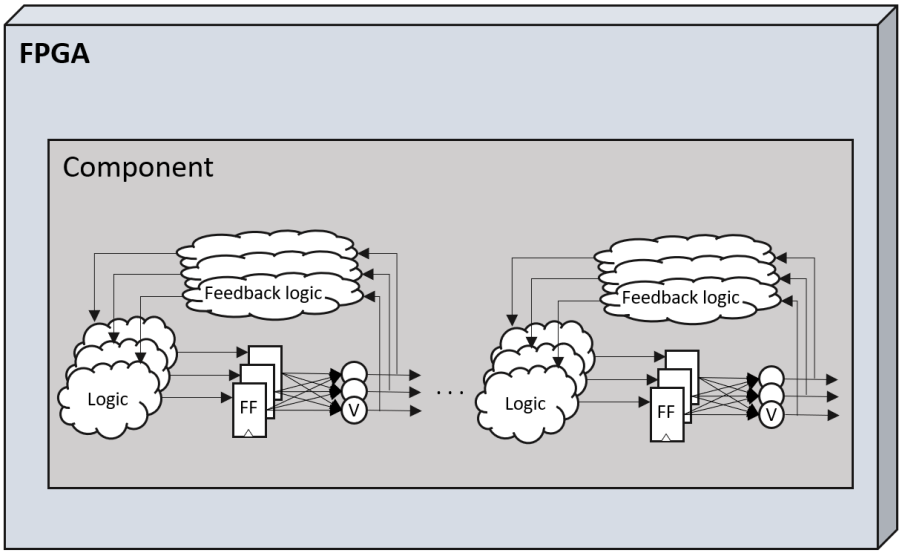
EDAC codes are commonly used to protect user data and rely on the addition of redundant bits. Detection codes, such as parity, can flag an inconsistency in the data and

Figure 4.3 – Coarse Grain TMR (CGTMR) implementation.



Source: From the author.

Figure 4.4 – Fine Grain Distributed TMR (FDTMR) implementation with feedback voters.



Source: From the author.

are often used in communication protocols. The Cyclic Redundancy Check (CRC) is an error detection code based on redundancy and is usually used for a large amount of data.

In data memories, ECC are frequently used for Single Error Correction and Double Error Detection (SECDED). The Hamming code is one of the most spread SECDED codes (KUMAR; UMASHANKAR, 2007). The Bose–Chaudhuri–Hocquenghem (BCH) is abstracted from the Hamming code for multiple error correction, and Reed–Solomon codes are a sub-set of BCH codes (HUFFMAN; PLESS, 2003). The standard error code algorithms can be computed for a variety range of bits. The higher the number of code bits, the more error bits can be detected in the data group.

4.1.2.3 Lockstep technique

Lockstep is a hybrid fault tolerance technique applied to processors based on software and hardware redundancy for error detection and correction (ABATE; STERPONE; VIOLANTE, 2008; VIOLANTE et al., 2011; GOMEZ-CORNEJO et al., 2013; PHAM; PILLEMENT; PIESTRAK, 2013; OLIVEIRA et al., 2018). Similar to DMR, two processors execute the same software, and the outputs are compared for error detection, but additional methods are used for error recovery. Checkpoint and rollback mechanisms are used at the software level, and processor duplication and checker circuits are implemented at the hardware level.

Typical lockstep works by executing the same application simultaneously and symmetrically in two identical processors. At system start-up, the processors are initialized with the same state and inputs (code, bus operations, and asynchronous events). In a fault-free execution, both processors are expected to perform the exact instructions allowing the monitoring of the data, addressing, and controlling buses from clock to clock (BOWEN; PRADHAM, 1993).

Verification points are inserted in the program code to lock the application execution and compare the system outputs. The addition of verification points is intrusive and requires updating the software source. If the processors have the same status at a verification point, the system is considered fault-free, and a checkpoint operation is performed. The checkpoint consists of saving the processors' context in a safe memory (e.g., a memory with ECC protection). The context is defined as all data resources used in the application execution necessary to repair the system in case of errors, such as registers and user data. When the results mismatch, the lockstep system restores the fault-free copy of the context through a rollback mechanism. The processors are recovered to a previous

state without errors and restart the application execution from that point.

4.1.3 Software-based techniques

Software-Implemented Hardware Fault Tolerance (SIHFT) techniques deal with faults on processors by protecting only the software, without hardware modification. These methods usually rely on adding code redundancy and comparison for error detection (GOLOUBEVA et al., 2006).

Software redundancy strategies can be applied at instruction and thread levels (OSINSKI; LANGER; MOTTOK, 2017). Mitigation at the instruction level frequently replicates the program assembly code for fault detection, such as Error Detection by Duplicated Instructions (EDDI) (OH; SHIRVANI; MCCLUSKEY, 2002) and Software Implemented Fault Tolerance (SWIFT) (REIS et al., 2005). The approaches that operate at the thread level are called Thread-Level Redundancy (TLR) and can implement Simultaneous Multithreading (SMT) to provide fault coverage on either single- or multi-core SoCs (REINHARDT; MUKHERJEE, 2000; MUKHERJEE; KONTZ; REINHARDT, 2002).

SIHFT techniques can also be designed to detect SEFIs. For instance, the Selective Software-only Error-detection Technique using Assertions (S-SETA) can be used to detect control-flow faults and put the system in a fail-safe state, allowing the implementation of a top-level recovery solution (CHIELLE et al., 2015). Multiple SIHFT techniques can be combined to mitigate data- and control-flow errors and increase system reliability. Schmidt, French and Flatley (2017) implemented multiple techniques at the software level, including checkpoints, control-flow assertions, heartbeat monitoring, and watchdog timers to protect a PowerPC processor in a flight mission.

4.1.4 Hardware monitors

Hardware monitors can detect errors by monitoring expected processor behavior such as bus traffic, pipeline tree, and data patterns. Watchdog timers, checkers, or dedicated IPs can be used to monitor the system (AZAMBUJA; KASTENSMIDT; BECKER, 2014). A watchdog processor is a module that detects errors by verifying the control-flow and memory access of the target processor (MAHMOOD; MCCLUSKEY, 1988; BENSO et al., 2003). Modern solutions use artificial intelligence and machine learning methods

to build watchdog circuits to detect SEUs in processors (VARAPRASAD et al., 2021). Heartbeat monitoring can also be performed in hardware by checking a toggling General Purpose Input/Output (GPIO), for instance.

Usually, hardware monitors require less area compared to replication techniques. Since the error is only detected, correction techniques can be combined for protection enhancement. Upadhyaya and Saluja (1986) proposed a watchdog processor for error detection combined with a rollback scheme for recovery.

4.1.5 Summary

From the presented fault tolerance techniques, this thesis focuses on periodic scrubbing of the FPGA configuration memory, triplication for the soft processor core, watchdog approach to detect timeouts, and memory duplication and refresh of the L1 cache.

This thesis uses the Xilinx SEM-IP for CRAM scrubbing during the investigations presented in chapters 6 and 7. In chapter 8, an external scrubbing is developed to improve the fault coverage and visibility.

CGTMR and FDTMR are used during the Rocket case study verification in chapter 6. For the NOEL-V and NOEL-VFT characterization, a distributed TMR is employed with feedback voters.

In chapter 6, a periodic reset is used to restore the Rocket processor's flip-flops state, and a watchdog monitors timeouts.

The L1 cache of the NOEL-V is protected with duplication and comparison, and periodic flush, as detailed in chapter 7. Chapter 8 describes the built-in EDAC methods and BRAM scrubbing protection of the NOEL-VFT soft processor.

4.2 Related works

4.2.1 Fault tolerance solutions for soft processors

The literature presents different works investigating soft error mitigation methods on processors. Kasap et al. (2020) revised a variety of techniques to protect the FPGA configuration memory and the user memory of soft processors, focusing on the LEON3

core. The fault tolerance techniques targeting soft processors synthesized into SRAM-based FPGAs usually consider redundant-based solutions such as TMR or DMR to protect errors in the logic and flip-flops; ECC or time-based solutions like flushing or refreshing in the user memories; and periodic scrubbing of the FPGA configuration memory.

The first step is to understand the sensitive parts of the soft processors. Cho (2018) analyzed the soft error susceptibility affecting flip-flops of the RISC-V Rocket and Berkeley Out-of-Order Machine (BOOM) soft processors. Results demonstrated that the most vulnerable parts of the RISC-V cores are the Control Status Register (CSR) and register file. The authors also showed that, although the raw error rates from the two processor cores are different, the error rates depending on the applications have strong correlations.

As described in section 3.1.3, SEUs in the processor register file are critical since they have a high chance of causing SDCs or control flow errors. Various works have proposed EDAC and SECDED approaches to protect the register file and other user memories, such as the caches. The same tendency is observed in commercial soft processors. The Frontgrade Gaisler commercial fault tolerant LEONFT and NOEL-VFT processor lines implement SECDED codes to all embedded memories (FRONTGRADE GAISLER, 2022b).

Ramos et al. (2018a) presented a method to protect the RISC-V Rocket soft processor's register file, which relies on parity error detection and inferred redundancy based on a design tool. The authors showed the method's effectiveness through emulation fault injection in correcting single-bit errors at a considerably lower cost than a traditional TMR approach. Dörflinger et al. (2020) designed an ECC controller with memory scrubbing for Rocket and BOOM processors, which led to an overhead of about 5% for logic and 41% for BRAMs in a Xilinx Virtex UltraScale+ FPGA.

In Heida (2016) work, a hybrid fault tolerance architecture is implemented through redundancy and ECC techniques with SECDED capabilities. Fault injection experiments showed that bit-flips in the pipeline are more likely to lead to errors in the processor register file. Although the system fault tolerance has been established, the design did not meet the expected clock frequency efficiency and the maximum resource usage requirement. These results are consequences of the long paths inherent to a redundant design.

Neri (2021) developed a fault tolerance design of the instruction decode stage in a RISC-V processor using triplication and ECC. No errors were observed in the benchmarks when the Fault Tolerance (FT) features were enabled. Li et al. (2022) refined a novelty SECDED approach applied between the pipelines of a RISC-V processor combined with

rollback for error recovery. The authors show the processor can handle different error conditions under simulation error injection.

Soft errors in the processor's arithmetic unit are also a concern due to the possibility of wrong computations. Gupta et al. (2015) proposed a fault tolerant RISC-V microprocessor architecture based on space and time redundancy to harden the Arithmetic-Logic Unit (ALU), and ECC to protect the registers and memories. The reliability of the system in the presence of single-bit faults was evaluated through fault injection. The proposed scheme was able to harden the ALU and data path against soft and hard errors with a penalty of 20% in area and 25% in performance.

Santos et al. (2020) propose a fault tolerance RISC-V processor implementation based on the RV32I integer instruction-set in which the ALU and control unit are triplicated, and hamming code is applied to the instruction fetch unit (PC) and register file. The initial version of the processor was validated under SEUs and SETs emulated in a Xilinx Zynq-7000 FPGA (SANTOS et al., 2020). Results showed an error propagation reduction of about 16 times. In the continuation of the study, the authors enhanced the proposed processor by adding SECDED to the data memory and a watchdog at the SoC level, which triggers a reset when the processor is non-responsive (SANTOS et al., 2022). Neutron irradiation on a Microchip flash-based Smartfusion2 FPGA demonstrated 97.73% correctness with complete protection on the processor.

Partial TMR has also been investigated for ALU protection in a Rocket processor, in which only one of the ALU operations is triplicated (RAMOS et al., 2019). Emulation fault injection results showed that the partial TMR differs up to 2% of correctness compared to the full triplicated ALU. In the worst case, the difference in protection was almost 5%.

An example of DMR approach for soft processors is presented by the Ferlini et al. (2012) work, where two LEON3 processors run the same application in parallel in two FPGA devices. This redundant implementation is part of the fault tolerance strategy of an OBC. Another work proposed a fault tolerant architecture in a quad-core RV32IM RISC-V processor that was implemented in an ASIC, 32 nm CMOS, and validated in a Xilinx Artix-7 FPGA (SHUKLA; RAY, 2022). The FT mode implies that instructions are executed in two pairs of DMR cores for error detection and re-execution for error recovery. All single-bit injected faults were successfully mitigated.

Heterogeneous solutions also bring the opportunity to increase processor reliability. For example, a lockstep architecture applied to a Rocket soft processor in the FPGA

fabric and a hard core Arm Cortex-A9 is implemented in a Xilinx Zynq-7000 FPGA for system enhancement (RODRIGUES et al., 2019). However, the primary drawback of such solutions is the synchronization complexity due to the distinct processors' architectures and the multiple clock domains.

A coarse-grain lockstep using MicroBlaze soft processors is proposed to protect an OBC of a CubeSat (FUCHS et al., 2019). The system is also enhanced with ECC on memories and CRAM scrubbing. The authors validated the proposed solution in Xilinx Kintex UltraScale and UltraScale+ devices and observed 50% lower power consumption on the UltraScale+ FPGA. The power variation is likely due to the technology fabrication of the devices, with the 16 nm FinFET UltraScale+ requiring less power than the 20 nm planar CMOS UltraScale. No soft error reliability data is provided in the reference.

Lindoso et al. (2017) added a hybrid fault tolerance to a LEON3 soft processor by protecting the memories with SECDED codes, monitoring control-flow errors with a hardware module, combining data-flow duplication and inverted branches techniques at the software level, and performing CRAM scrubbing. Under neutron testing, the mitigated LEON3 could reduce more than four times the cross section compared to the unhardened version.

As discussed in previous sections, CRAM scrubbing is essential on SRAM-based FPGAs. Applying the distributed TMR approach to soft processors can also be highly efficient. Keller and Wirthlin (2017) showed that the distributed triplication reduced eight times the neutron sensitiveness of a LEON3 processor. Moreover, combining TMR and CRAM scrubbing boost improvement to 27 times. Aranda et al. (2020) work investigated the critical bits and essential bits ratio for a Rocket processor in the Xilinx Kintex UltraScale. The study injects single faults in the design's essential bits and checks for failures. The triplicated version of the Rocket processor reached 99.7% of correctness for combined benchmarks, while the unhardened resulted in 90.3%. Walsemann et al. (2023) developed a fault tolerant RISC-V processor based on triplication and SRAM memory scrubbing. The processor was implemented in a custom ASIC, 65 nm CMOS technology. Laser testing targeting the SRAM standard cells showed the high effectiveness of the FT features in correcting all SEUs.

Wilson and Wirthlin (2021) evaluated triplicated versions of four RISC-V soft processors (PicoRV32, Kronos, Taiga, and VexRiscv) in a Xilinx Artix-7 FPGA under CRAM fault injection and more than 80 times reliability improvement. Under neutron testing, a triplicated Taiga soft processor lowered 33 times the cross section (WILSON;

WIRTHLIN, 2019), and a triplicated VexRiscv performed ten times better Mean Fluence to Failure (MFTF) (WILSON et al., 2021), with the drawback of 5.6 and 4 times more area, respectively. In a continuation work, the authors characterized the RISC-V Taiga, VexRisc, PicoRV32, and NOEL-V soft processors in a Xilinx Kintex UltraScale FPGA under neutron beam (WILSON et al., 2023). The authors explored the triplication of the entire SoC, including DSPs and BRAMs, combined with external CRAM scrubbing. Results demonstrated a cross section reduction of up to 75 times, at the expense of 4.8 times more resource usage and 12.4% performance reduction.

The PULP platform's RISC-V cores can also be enhanced for fault tolerance (ROGENMOSER; TORTORELLA, 2022). The authors propose an on-demand redundancy to the processor cores that, when enabled, works as a usual TMR system. However, when disabled, the three cores execute instructions independently toward high performance. Additionally, ECC and scrubbing are employed on the user memories. No experimental data on the soft error susceptibility has been provided in that reference.

This thesis fills many gaps not covered in past works. We investigate the use of fault tolerance techniques on the RISC-V Rocket, NOEL-V, and NOEL-VFT soft processors implemented in SRAM-based FPGAs – Zynq-7000, Zynq UltraScale+, and Kintex UltraScale, respectively– under different scenarios and deeply analyze the trade-offs between the reliability improvements and the system drawbacks. The impact of accumulated faults on FPGA configuration memory implementing the Rocket and NOEL-V is investigated to understand the processors' susceptibility under multiple upsets. The literature works have primarily investigated the RISC-V processors under single-bit faults. We study applying protection to specific elements of soft processors, such as the processor core, the integer unit and cache controller, and the L1 cache, and we explore the FPGA configuration memory susceptibility. We individually analyze the impact of each protection on the overall processor susceptibility. Additionally, an investigation is carried out on the L1 cache of the Rocket processor to characterize different cache topologies under faults in the user memories.

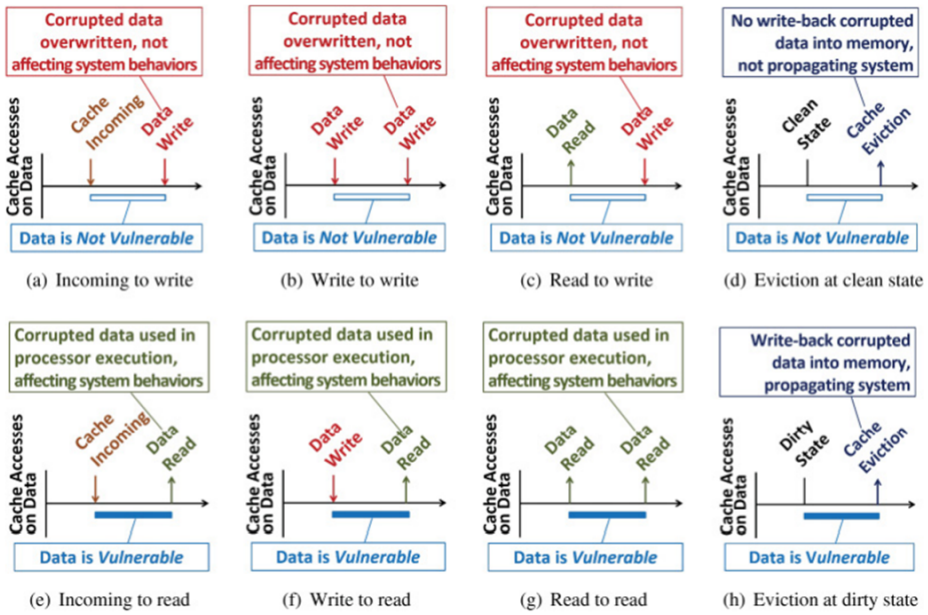
To our knowledge, this thesis is the first work to publish the SEE characterization of the commercial fault tolerant NOEL-VFT processor. Further enhancement of the NOEL-VFT is also investigated by adding an external CRAM scrubbing combined with distributed TMR. An FPGA supervisor is developed to program and scrub the CRAM externally through the SelectMap interface. We perform accelerated irradiation testing using heavy ion and proton beams, which are the most relevant particles present in-orbit,

such as Low Earth Orbit (LEO) and Geostationary Orbit (GEO). The amount of generated data with the various investigations makes this thesis a valuable guide for future implementations of RISC-V soft processors in SRAM-based FPGAs. The results of this thesis compared to the state-of-the-art works are presented in chapter 9.

4.2.2 L1 cache vulnerability

Using the cache memories can significantly impact the soft processor vulnerability (TAMBARA et al., 2015). However, the fact that the data is stored in the cache not necessarily implies that it is susceptible to soft errors. The vulnerability time window of the data is related to its usage by the processor. Figure 4.5 shows some examples of data vulnerability related to different cache accesses, such as data read, data write, and cache eviction (KO et al., 2017). An upset in the cache will only propagate through the system if the processor reads the erroneous data or writes back to the main memory in a dirty state.

Figure 4.5 – Cache data vulnerability related to different accesses: data read, data write, and cache eviction.



Source: (KO et al., 2017).

Ozturk, Topcuoglu and Kandemir (2022) studied the error propagation on cache memory through fault injection using the GEMFI simulator. The authors found a dependency between the error propagation and the type of benchmark computations, order of

executions, and input vectors. This result is expected due to the different masking effects of different applications.

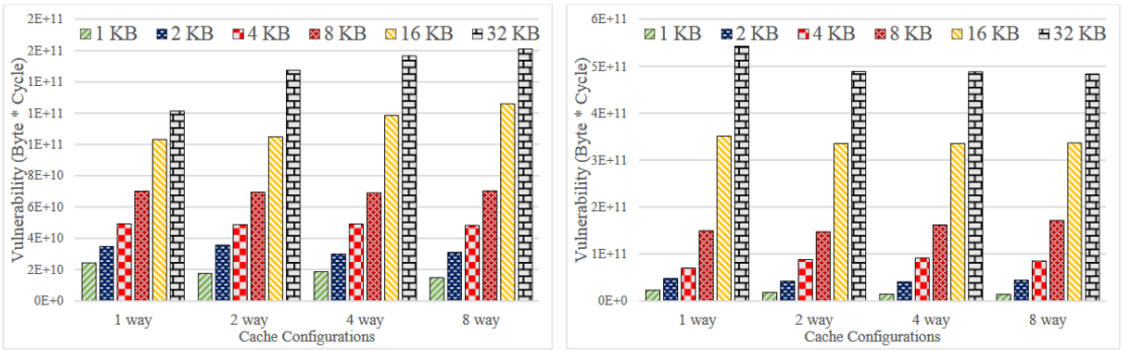
Tselonis et al. (2016) performed fault injection simulation using a MARSSx86 microarchitectural x86-64 simulator to investigate transient faults in the processor caches. The susceptibility results regarding the cache associativity may change between data and instruction caches. An overall lower vulnerability is found in a 2-way data cache and 8-way instruction cache. Results shown that increasing the cache size increases the error masking effect. However, this may be a test artifact related to a non-normalized fault injection.

As shown by Yuanwen Huang and Mishra (2016), the vulnerability is expected to increase with the cache size since the miss rate decreases. The authors computed the data vulnerability based on the Architectural Vulnerability Factor (AVF) and the summation of the vulnerable time of bytes in the cache (in byte cycles). Larger caches present lower miss rate. A lower miss rate means that the valid data stays longer time in the cache, leading to a higher vulnerability.

Livany, Salehi and Kargar (2020) investigated, via simulation on the GEM5, the vulnerability of different cache topologies, varying the size from 1 to 32 KB with 1 to 8-way set associativity. The work evaluates the cache susceptibility by computing the Cache Vulnerability Factor (CVF). Figure 4.6 depicts the relation between vulnerability and cache topology. Similar to the Yuanwen Huang and Mishra (2016) work, results from Livany, Salehi and Kargar (2020) show that increasing the cache size increases the vulnerability. Results also demonstrate that the data cache presents higher vulnerability than the instruction cache. As expected, the application execution time decreases by increasing the cache size. The authors noticed that the execution time saturates on 8 KB cache size, and increasing more the cache did not improve performance. Similarly, Eckert et al. (2017) demonstrated using an ARM soft processor embedded in an Artix-7 FPGA that increasing the cache associativity improves performance, but the gain is less expressive at larger cache size (i.e., above 16 KB). Those results might also be related to the execution flow characteristics of the evaluation benchmarks.

This thesis aims to confirm if those simulated results are also applicable to soft processors embedded in FPGAs. Additionally, the cache size and performance tradeoff against the overall vulnerability still needs to be determined. Although a larger cache implies a higher data vulnerability, a reduced execution time also means a reduced exposure time for the entire system. The boost in performance with larger cache sizes may lead to

Figure 4.6 – Cache vulnerability for different topologies: (a) varying instruction cache configuration with constant 16 KB data cache; (b) varying data cache configuration with constant 16 KB instruction cache.



(a) (b)

Source: (LIVANY; SALEHI; KARGAR, 2020).

more application workload correctly computed between failures, which can be attractive depending on the mission requirements. This thesis performs emulation fault injection on the FPGA's user memories (BRAMs) to investigate the different cache topologies and susceptibilities on the Rocket soft processor. Different cache sizes are also assessed on the NOEL-V soft processor under proton testing.

5 RADIATION CHARACTERIZATION METHODOLOGY

This chapter gives an overview of the SEE ground testing, evaluation metrics, and estimation of orbit error rate. System validation is a mandatory step for a successful mission. Qualification methodology can be used to verify the system susceptibility, assess the feasibility and reliability of the implemented fault tolerance methods, and prepare for the expected error rate during the mission lifetime.

5.1 Single Event Effects testing

5.1.1 Accelerated ground testing

Accelerated ground testing can be used to simulate the conditions of a radiation environment, either space or terrestrial. The simulated artificial condition is limited to a single particle source with computed accelerated fluxes. The particle beam commonly targets a specific component, defined as Device Under Test (DUT). On the other hand, multiple particles are present in a natural radiation environment, with real particle rates, and the entire system is affected, which might lead to combined fail effects. Ground testing might not accurately replicate an existing radiation environment but is a reliable way to assess the component behavior under radiation-induced effects.

There are plenty of standards that guide radiation testing in integrated circuits (LAUENSTEIN, 2016). Examples of well spread SEE standards are JEDEC JESD57 (JEDEC, 1996) and ESA ESCC25100 (ESA, 2014). These standards establish instructions and requirements for preparation, execution, and post SEE testing.

Heavy ions, protons, and neutrons are common particles used for SEE testing. During test preparation, one should simulate the particle range in the die layers to ensure the penetration through the Sensitive Volume (SV). The SRIM software (ZIEGLER, 2013) can be used for Transport of Ions in Matter (TRIM) calculations.

The neutron beam presents a significant depth penetration and usually does not require any DUT sample preparation. Proton beam also gives a high projected range that can penetrate hundreds of millimeters depending on the energy. Heavy ion testing, on the other hand, may require sample preparation to guarantee the ions reach the sensitive region. Different approaches can be used to reduce the device thickness based on the chip packing, such as removing the lid, grinding, polishing, or chemically etching part of the

substrate or other package layers.

The particle Effective LET (LETeff) shall be considered as the LET value at the active surface of the SV. In case the DUT is tilted during the heavy ions irradiation, the LETeff is computed as equation 5.1, where θ is the DUT inclination angle (ESA, 2014).

$$LETeff = \frac{LET(normal\_incidence)}{\cos\theta} \tag{5.1}$$

The SEE test planning shall consider LETs and energies that will trigger different effects in the device. The selected particle fluxes and fluences should be enough to gather significant events. The flux is the number of particles strikes per unit area per unit time, and the fluence is the total amount of particles strikes per unit area during the exposure time. In summary, the relation fluence (ϕ) and flux can be evaluated by equation 5.2.

$$\phi = flux \times t \tag{5.2}$$

The minimal number of failure events recommended in the JESD-57 standard is 100 events to ensure meaningful statistical calculations (JEDEC, 1996). However, gathering many events is not always feasible. The ESCC25100 standard advises accumulating a minimum fluence of 10^7 and 10^{11} p/cm^2 for heavy ions and protons, respectively (ESA, 2014).

The DUT radiation sensitivity can be assessed by performing static and dynamic tests. Ramos et al. (2018b) presents a relevant example of evaluating the static and dynamic radiation sensitivity of processors.

A static test is used to evaluate the intrinsic susceptibility of the component. The static tests aim at the technology characterization of the DUT elements, such as memory cells, FFs, combinatorial cells, PLL, etc. Usually, the memory elements testing is performed by writing a predefined pattern to the memory, starting the irradiation, and periodically checking the values for upsets. The other elements can be more complex to evaluate. For a soft processor implemented in an FPGA, the static sensitivity is related to the FPGA device resources. Usually, the FPGA vendor provides a radiation report describing the typical sensitivity for the configuration memory and BRAMs.

A dynamic test aims to assess a functional system under irradiation, focusing on the DUT's availability and responsiveness. For instance, the dynamic test of a soft processor consists of executing a software application and verifying its outputs. Quinn et al. (2015) propose a set of benchmarks for reliability analysis of dynamic radiation testing

on FPGAs and processors. The software benchmark suite includes algorithms such as CoreMark, Fast Fourier Transform (FFT), Matrix Multiplication (MxM), Advanced Encryption Standard (AES), and Quicksort (Qsort). The system's dynamic response depends on the masking effects of the design and application, which can avoid error build-up. In other words, not every upset will lead to a failure event due to the dynamic characteristics of the system. For components with the SEE characterization data available (i.e., the static sensitivity), one can estimate the expected error rate per application execution and adjust the testing accordingly.

5.1.2 Emulation Fault injection

Fault Injection (FI) is a method used to reproduce the radiation-induced effects in devices. It can be employed to characterize points of failure that require being hardened and assessing mitigation techniques implementations. Faults can be injected at a transistor level, gate and register transfer level, or system level (ANGHEL et al., 2007). In a soft processor implemented in an SRAM-based FPGA, bit-flips injected in the FPGA configuration memory will affect the processor's Register Transfer Level (RTL) design, and faults injected in the BRAMs will affect the application execution at the user level.

Injectors can be based on hardware or software implementations. Hardware engines are composed of an external module or component that usually does not interfere with the dynamic system functionality for injecting faults. In a software-based injection, the system functionality may be affected by the injection engine itself. For instance, in processors, the execution of the application is usually paused, the fault is injected, and then the application is released and checked for errors. The injector is desired to be the least intrusive to the evaluated system. Velazco, Rezgui and Ecoffet (2000) developed an algorithm using interruption mechanisms to inject faults with low intrusion in processors.

Upsets can be deterministic or random injected. In a deterministic approach, the bit-flip location is predefined. Otherwise, the target bit can be selected randomly. Injections in memory elements are said to be exhaustive when all cells are exercised during a deterministic bit-by-bit injection. For single-bit injections, one bit is upset, the system functionality is verified, and the upset is cleaned before a new injection. The exhaustive single-bit injection targeting the configuration memory of an SRAM-based FPGA is helpful for identifying all critical bits in the design. On the other hand, in accumulative injections, the bit-flips accumulate over time. In this case, the upsets are only cleaned

when the system experiences a failure. Accumulative injections are commonly random and focus on evaluating the system's reliability under multiple faults. Benevenuti and Kastensmidt (2019) showed a qualitative comparison between single-bit exhaustive and accumulative random injections.

Usually, injection approaches focus on the susceptible design area to inject faults, not covering the time domain. The operational timing of the injected fault can widely impact the fault effects (QUINN et al., 2013). For instance, injecting SETs requires that the faults are distributed uniformly over the clock cycles since the timing affects the SET latch. In processors, faults are also dependent on the software execution. Faults in architecture point that is no longer exercised will not trigger any error, or upsets in memory elements are masked if these elements are overwritten in sequence. For the fault injection to better mimic the radiation-induced effects, the faults should statistically cover the timing within a clock cycle and application execution (QUINN et al., 2013).

Fault injection can be performed by simulation using modeling tools or emulation in a device. For deeper reading on the topic, surveys in the literature presents techniques for both emulation and simulation fault injection in general (ZIADE; AYOUBI; VELAZCO, 2004; QUINN et al., 2013). Sterpone and Violante (2007) developed a methodology for static and dynamic SEU sensitivity on SRAM-based FPGAs. Moreover, Quinn and Wirthlin (2015) demonstrate several emulation methods to replicate the SEU and SET effects on FPGAs. For the proposes of this thesis, emulation fault injection is covered focusing on soft processors embedded into SRAM-based FPGAs.

There are plenty of fault injection tools available in the literature. In SRAM-based FPGAs, faults are usually inserted by emulating faulty component models or using partial reconfiguration to flip bits in the configuration memory. In the case of soft processors, emulating SEE effects using faulty models in FPGAs is valuable for prototyping the hardware before developing the final ASIC. For instance, the SET sensitivity of a LEON2 soft processor targeting an ASIC technology can be instrumented through FF faulty models at RTL and gate level in a Virtex-6 FPGA (ENTRENA et al., 2012). Using a similar approach, Mansour and Velazco (2012) performed an SEU fault injection targeting the FFs of a LEON3 soft processor in a Virtex-5 FPGA using a modified register model.

The fault injection should focus on the FPGA resources for soft processors whose final implementation targets an FPGA. Carlo et al. (2014) demonstrated a methodology to inject upsets in the essential bits of a LEON3 soft processor in a Virtex-6 FPGA. Harward et al. (2015) assessed the SEU sensitivity of five different soft processor architectures via

fault injection performed in the Virtex5 FPGA. Sari and Psarakis (2016) presented a fault injection framework targeting the IU, Memory Management Unit (MMU), and arithmetic units of a LEON3 soft processor implemented in a Virtex-5 FPGA. Ramos, Maestro and Reviriego (2017) emulated upsets in the configuration memory of an Artix-7 FPGA to characterize a Rocket soft processor. Cho (2018) performed an FF-level fault injection in two RISC-V-based soft processors embedded into Zynq-7000 FPGA. Wilson and Wirthlin (2021) implemented automatic test equipment using the Internal Configuration Access Port (ICAP) to perform fault injection experiments on five soft processors embedded into Artix-7 FPGA. These works are further discussed in Chapter 4.2.

5.2 Evaluation metrics

Many evaluation metrics are available to assess the system susceptibility to radiation-induced errors and compare different design approaches. This section presents some of the most used metrics.

5.2.1 Cross section

The cross section (σ) is a standard metric used to evaluate the DUT radiation-sensitive area. It is computed by the relation between the number of errors and the exposed fluence of particles (ϕ), as defined in equation 5.3. In case the DUT is tilted during the irradiation, the fluence must be adjusted according to the inclination angle (ESA, 2014). Most irradiation facilities already compute the final fluence based on the DUT tilting.

$$\sigma = \frac{\#errors}{\phi} \tag{5.3}$$

The cross section unit can be expressed as $cm^2/device$, for failures considering the entire device. For errors per memory bit, the obtained cross section should be divided by the total number of bits, and it is represented as cm^2/bit.

The static cross section is obtained from the component static testing and defines the intrinsic susceptibility per area unit. On the other hand, the dynamic cross section is related to the number of functional failures per area unit.

5.2.2 Fault injection error rate

For emulated system failures, the fault injection error rate (or failure rate) can be used as an alternative metric for the cross section. Instead of the fluence, this error rate is based on the number of injected bit-flips. Therefore, the fault injection error rate (τ_{inj}) is the relation of the number of errors divided by the number of injected bit-flips, as defined in equation 5.4.

$$\tau_{inj} = \frac{\#errors}{\#injected\_bit\text{-}flips} \tag{5.4}$$

5.2.3 SEU error rate estimation

In case of the unfeasibility of exposing the DUT under a beam of particles, the dynamic SEU error rate can be estimated based on the static cross section for the specific technology and fault injection results, as per equation 5.5 (REZGUI et al., 2001) (VELAZCO; FAURE, 2007).

$$\tau_{SEU} = \sigma_{SEU} \times \tau_{inj} \tag{5.5}$$

Although this standard formula is valid and easily applied to simple systems, finding an accurate estimation based only on fault injection for complex systems is significantly challenging. For example, the fault injection analysis of soft processors embedded in SRAM-based FPGAs has a limited evaluation scope. As described in section 5.1.2, faults are usually injected in the FPGA CRAM and BRAMs. Other resources are not accessible such as DSP slices, clock tree, and other internal FPGA elements.

5.2.4 Mean time to failure and mean time between failures

Mean Time to Failure (MTTF) is the average time to a failure occurs in the system and is usually attributed to non-repairable failures. On the other hand, the Mean Time Between Failures (MTBF) measures the frequency of recoverable failures. Both concepts can be easily mixed when testing components. Krasich (2009) gives an overview on how to apply both MTTF and MTBF as reliability measurement to systems.

In summary, equation 5.6 expresses the mean distribution of MTTF as the reliability at a time ($R(t)$), which gives the inverse of the failure rate (λ) (SIEWIOREK; SWARZ, 1998). The failure rate is defined per equation 5.7 as the total number of failures divided by the test time.

$$MTTF = \int_0^\infty R(t)dt = \int_0^\infty e^{-\lambda t}dt = \frac{1}{\lambda} \tag{5.6}$$

$$\lambda = \frac{\#failures}{t} \tag{5.7}$$

Equation 5.6 also applies to MTBF when the failure rate is related to recoverable fails of a single component or system. Moreover, MTBF is usually presented as a function of the cross section and the particle flux, as shown in equation 5.8, which is derived from equations 5.2, 5.3, 5.6 and 5.7.

$$MTBF = \frac{1}{\sigma \times flux} \tag{5.8}$$

5.2.5 Mean fluence to failure

Berg et al. (2017) proposed a reliability analysis focused on radiation-induced errors that is a function of fluence instead of using it from the time domain directly as defined in the MTTF. The MFTF is the average fluence (ϕ) to experience a failure and is expressed as the inverse of the cross section (σ), as per equation 5.9. The reliability function, in this case, is defined as equation 5.10.

$$MFTF = \frac{1}{\sigma} \tag{5.9}$$

$$R(\phi) = e^{\frac{-\phi}{MFTF}} \tag{5.10}$$

5.2.6 Mean executions and workload between failures

The Mean Executions Between Failures (MEBF) and Mean Workload Between Failures (MWBF) are metrics generally used to evaluate the system's reliability considering its performance. Higher MEBF and MWBF means higher functional reliability.

MEBF method considers the relation of the number of successful application executions until a failure occurs (RECH et al., 2014). MEBF relates the error rate and system performance by the ratio between the MTBF and application execution time (in seconds), as defined in equation 5.11.

$$MEBF = \frac{MTBF}{exec.\,time} \tag{5.11}$$

On the other hand, the MWBF analyzes the trade-off between the error rate and performance by the amount of data correctly computed until an output error occurs (RECH et al., 2014) (SANTINI et al., 2014). Equation (5.12) relates the data workload with cross section, beam flux of the accelerated ground testing, and the application execution time (in seconds).

$$MWBF = \frac{workload}{(\sigma \times flux \times exec.\,time)} \tag{5.12}$$

5.2.7 Empiric reliability

The empiric reliability is defined as the capability of the design to provide the correct result in time. This thesis uses a similar approach as defined by Berg et al. (2017) for the MFTF, but the reliability is represented using the experiments' empirical data results (i.e., number of failure events). Equation 5.13 represents the reliability function $R(\phi)$, where $F(\phi)$ is the cumulative distribution function abstracted from the number of accumulated faults, for emulation fault injection, or particle fluence, for accelerated ground testing.

$$R(\phi) = 1 - F(\phi) \tag{5.13}$$

5.3 SEE error prediction

The SEE error rate in-orbit is a failure prediction for a specific radiation environment based on accelerated ground testing. Standard models such as the CREME96 for cosmic rays and APB8 for trapped protons are used to calculate the particles fluxes per LET or energy in a defined orbit. Typically, the predictions target LEO and GEO orbits.

The device or system failure rate is estimated based on the SEE cross section and the particles fluxes for the defined environment. Firstly, a statistical distribution of the cross section data is computed, then the failure rate is estimated by combining the particles fluxes for the defined environment and the distribution data. Additionally to the cross section data, the rectangular parallelepiped (RPP) model of the SV directly impacts the error rate due to the charge collection in the cell area (PETERSEN et al., 2005).

Several works in the literature have investigated effective ways of performing the statistical calculation for bounding SEE rates (SRINIVASAN; TANG; MURLEY, 1994)(PETERSEN et al., 1992)(PETERSEN et al., 2005)(LADBURY, 2007). The Weibull distribution (WEIBULL, 1951) is commonly used for fitting the cross section data. The 4-parameter Weibull fit consists of:

- $LETth$ or LET_0: minimal LET to trigger a failure event (LET threshold) in heavy ion testing; for proton testing, the energy threshold shall be used;
- σ_{lim} or σ_{sat}: limiting or saturation cross section;
- W: Weibull width;
- S: Weibull shape.

The uncertainty in the rate calculation is reduced if many events are presented for several LETs or energies data points. The lower the uncertainty in defining the Weibull parameters, the better the resulting data fitting. Moreover, heavy ions and protons data shall be input to the calculations to improve the SEE error rate accuracy. The Weibull distribution can be used for fitting both heavy ion and proton data. The Bendel distribution is also a well-known method for fitting proton data (BENDEL; PETERSEN, 1983). Models, such as SIMPA (DOUCIN et al., 1995), PROFIT (CALVEL et al., 1996), and METIS (WEULERSSE et al., 2015), can be used to estimate the proton fit curve based on the heavy ion cross section data in case of non-availability of the proton data.

SPENVIS (ESA, 2018) and OMERE (TRAD, 2022) are free software tools used to compute the SEE error rate. The SEE results are given in events per day detailed for cosmic rays, trapped and solar protons. The total error rate considers the total estimation for heavy ions and protons based on the selected models and standards. The events are defined per device or bit, depending on the specified input data. One should check the correct number of SV cells and the RPP model being used in the calculations. Pessimistic or conservative SEE predictions can lead to costly over-design protection. However, under error rate estimation might result in mission failure.

6 EXPLORING THE COTS RISC-V ROCKET SOFT PROCESSOR UNDER RADIATION EFFECTS

This chapter presents the SEE characterization of the COTS RISC-V Rocket soft processor implemented in a Xilinx Zynq-7000 FPGA. The first section explores the processor under a fault injection methodology emulating bit-flips in the FPGA configuration memory and heavy ion accelerated ground testing. The fault injection targets only the CRAM bits, while the heavy ion irradiation affects the whole FPGA device. The second section investigates the influence of the L1 cache topology on processor susceptibility by emulating faults in user memories.

6.1 SEE characterization of the COTS Rocket soft processor

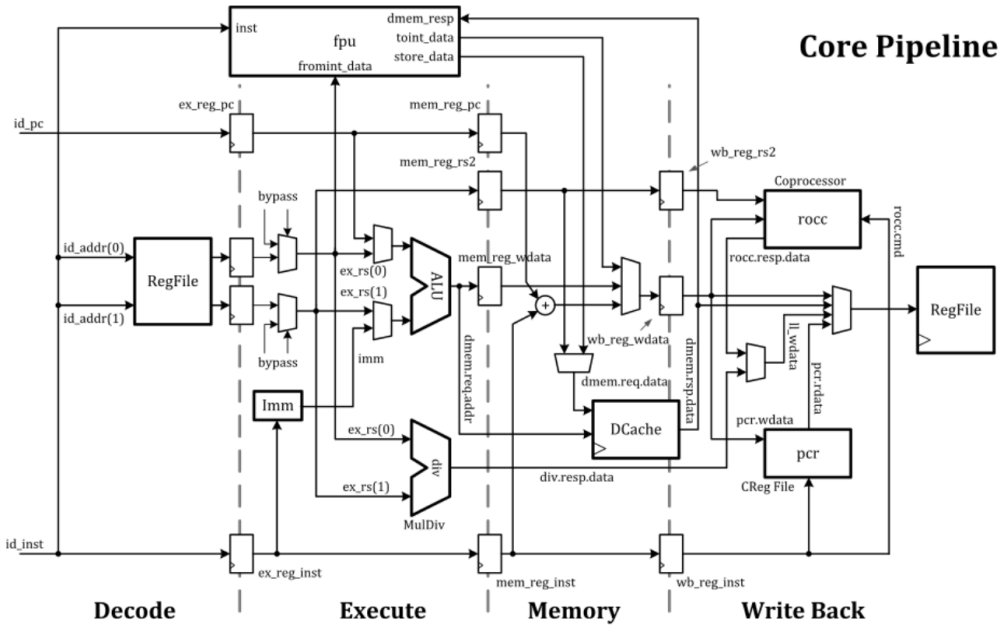
6.1.1 COTS RISC-V Rocket soft processor

The Rocket processor is a customized general-purpose RISC-V soft processor generated using the open source Rocket Chip SoC generator (ASANOVIĆ et al., 2016). It can be implemented in the RISC-V 32-bit (RV32G) or 64-bit (RV64G) ISAs. Rocket has a 5-stage single-issue in-order scalar pipeline, L1 data and instruction caches, integer ALU, and optional FPU. Figure 6.1 depicts the internal architecture of the core pipeline (SONG, 2015), showing the internal four stages: decode, execute, memory, and white back. The instruction fetch stage is not described in the figure.

Rocket Chip SoC generator builds the RISC-V based platform using the Chisel language. The Chisel is a Scala hardware description language that generates synthesizable Verilog (BACHRACH et al., 2012). The generated Verilog SoC can be implemented in ASICs or FPGAs.

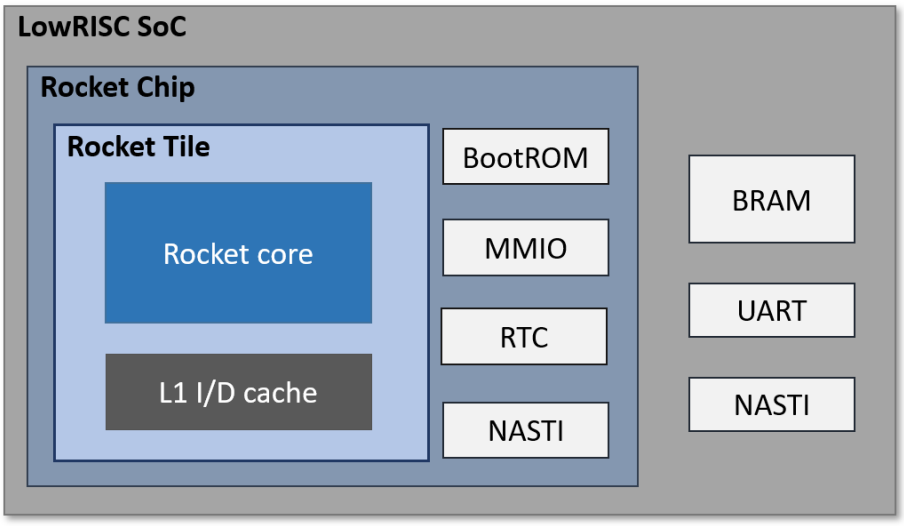
This thesis is based on the lowRISC SoC implementation (SONG, 2015), which contains the Rocket Chip generator; peripherals, such as Universal Asynchronous Receiver-Transmitter (UART) and Memory Management Controller (MMC); and an Advanced Extensible Interface (AXI) compatible with the Xilinx DDR memory controller.

Figure 6.1 – Architecture description of the Rocket processor pipeline.



Source: (SONG, 2015).

Figure 6.2 – LowRISC SoC description.



Source: From the author.

A simplification of the lowRISC SoC design implemented is described in figure 6.2. The Rocket Tile is the block that encloses the RV64G Rocket core and the L1 caches. The system configuration used in this thesis consists of a single-core processor with 8 KB L1 instruction and data caches, FPU disabled, and no L2 cache. The L1 caches are implemented in BRAMs. The top-level module of the Rocket Chip contains the Memory-Mapped I/O (MMIO) (MMIO) peripherals, BootROM, Real Time Clock (RTC), and NASTI interface. The BootROM includes the first stage bootloader with the first instructions to be executed by the processor. The NASTI interface is an adaption of the AXI-Lite to access all IO devices. The lowRISC SoC also includes on-chip BRAM memory, used as boot and user memory, and can be configured to access external memories.

6.1.2 Investigation methodology

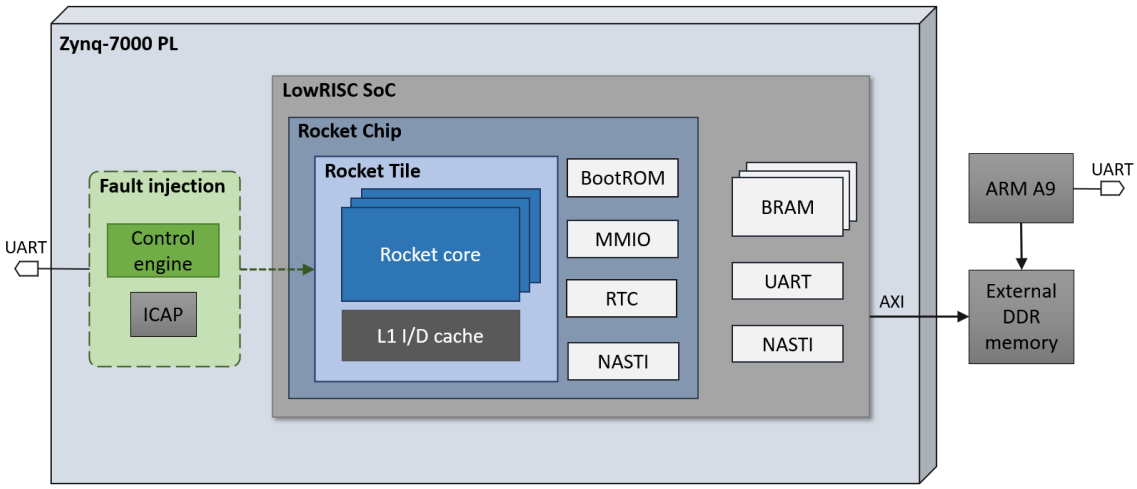
The investigation methodology aims to characterize the SEE susceptibility of the Rocket soft processor, with and without fault tolerance, implemented in an SRAM-based FPGA. The processor is embedded in a Xilinx Zynq-7000 APSoC, and the reliability analysis is performed via emulation fault injection and heavy ion testing. The fault injection experiments target the FPGA CRAM, while the entire FPGA device is irradiated during the heavy ion testing. The efficiency of well-known fault tolerance methods is explored to improve the overall system reliability.

6.1.2.1 Platform setup

The case study device adopted for the proposed analysis is the COTS Xilinx Zynq-7000 (XC7Z020-CLG484 part) APSoC. The device is fabricated in a 28 nm CMOS process. The Zynq-7000 is composed of two main hardware blocks: the PL, which consists of an SRAM-based FPGA layer, and the PS, which is formed around an ARM Cortex-A9 processor (XILINX, 2016b).

Figure 6.3 presents the system evaluation setup based on a commercially available ZedBoard Development Board. The lowRISC SoC test design is implemented into the Zynq PL. The Rocket soft processor runs a benchmark and saves the resulting outputs in the external DDR memory. The diagnosing is performed by the ARM Cortex-A9 processor that computes the golden output values, compares both results, and reports via UART.

Figure 6.3 – Evaluation setup of LowRISC SoC implementation into Zynq-7000 APSoC.



Source: From the author.

The ARM Cortex-A9 processor is only used for monitoring and is not under evaluation.

The system assessment is performed using unhardened (unprotected) and protected Rocket core versions. Different fault tolerance techniques are tested, such as scrubbing, TMR, and watchdog. All designs run at 20 MHz, which is the maximum frequency achieved in the triplicated implementations. An investigation was also performed using the unhardened core at higher frequency (50 MHz) to understand the impact on susceptibility.

To prevent the processor from booting with corrupted instructions, the boot memory is triplicated in all designs, including the unhardened. Moreover, a processor reset is performed in all designs to return the system to a known state after each benchmark execution. The fault injection system, described later in this section, is only used during the emulation experiments and is not present in the designs under heavy ion testing.

6.1.2.2 Software benchmarks

Three bare-metal benchmark applications are used for the evaluation:

- 32×32 Matrix Multiplication (MxM);
- Advanced Encryption Standard (AES) with 32 elements array; and
- Quicksort (Qsort) with 255 elements array.

Table 6.1 summarizes the characteristics of the benchmarks running on the Rocket soft processor. These applications are standard benchmarks extensively used for testing processors and FPGA designs (QUINN et al., 2015). All applications use 32-bit data encoded in fixed-point (Q16). The applications are written in C programming language and are originally from the MiBench benchmark suite (GUTHAUS et al., 2001). The applications' codes have been updated for a higher exercise of the data range, so the benchmarks are initialized with data randomly generated through a Pseudo-Random Number Generator (PRNG) at every execution.

The PRNG computation is executed in software and is expected to have low interference in the execution of the benchmarks. Table 6.1 also describes the execution time for the random numbers generation. A 16-bit checksum is computed for the resulting output of MxM and AES applications to minimize the verification time, and the entire array is compared in the Qsort. The total application execution time is computed for a 20 MHz clock frequency, and it is based on an average of multiple executions since the timing measurement can vary a few milliseconds between executions.

Table 6.1 – Benchmark applications characteristics running on the Rocket soft processor.

Benchmark	**Array length (32-bit data)**	**Workload (bits)**	**Total app. exec. (ms)**	**PRNG exec. (ms)**	**Verification (data check)**
MxM	32×32	$32,768$	745.87	17.77	16-bit checksum
AES	32	$1,024$	61.58	1.81	16-bit checksum
Qsort	255	$8,160$	555.48	12.40	full array

6.1.2.3 Failure definition

As discussed in chapter 3, bit-flips in the FPGA configuration memory may change the soft processor's circuitry functionality, leading to functional failures. For instance, faults in the control flow may provoke processor exceptions or losses of sequence. Bit-flips affecting the instruction cache or cache controller can also lead to hangs. Moreover, timeouts may occur when the clock tree or the system reset is affected by a soft error. Faults affecting data memory elements more likely cause the SDCs, and the processor register file is implemented in LUTs, increasing susceptibility. The Rocket soft processor is defined as fully functional when the resulting outputs from the application benchmarks are correct and there is no interruption in its execution. Otherwise, a failure is counted.

For this investigation, the Rocket design events are defined as below:

- Unnecessary for Architecturally Correct Execution (UNACE): the application finishes its execution as expected with the correct output. It is not a failure.
- SDC: refers to wrong computations in the benchmark. The application finishes its execution with an erroneous output, and the processor is responsive. It is a failure.
- Timeout: refers to the absence of benchmark outputs after a given time. The application execution does not finish correctly, which can occur due to SEFIs or processor hangs. It is a failure.

6.1.2.4 Emulation fault injection

Figure 6.3 also shows the CRAM fault injection system implemented in the Zynq PL. The control engine is based on previous work performed on Xilinx 7 Series FPGAs (TONFAT et al., 2016). The Fault Injection (FI) engine explores the ICAP component to read and write CRAM frames while XOR'ing the value of specific bits. Software running on a host computer determines the position of each injected bit-flip. The software control is performed via UART. The FI engine injects random bit-flips in a target area of the FPGA in order to mimic SEUs caused by ionizing radiation.

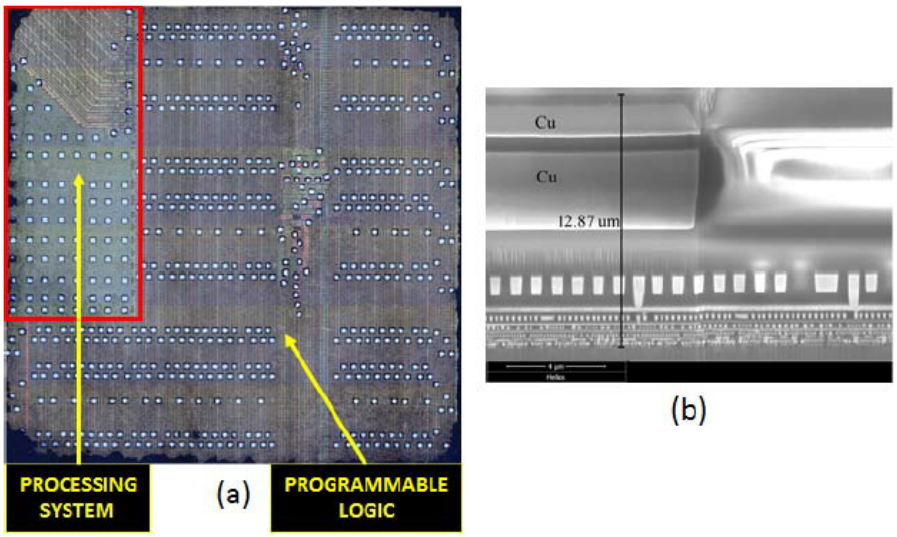
Aiming to evaluate only the Rocket processor core under faults, the FI target area is restricted to the Rocket Tile. Although it is feasible to define a restricted floorplanning area to individual FPGA design blocks, isolating the Rocket core from the L1 cache memory is not synthesizable. Therefore, the entire Rocket Tile area is affected by the emulated bit-flips. Only the CRAM bits are upset via the performed fault injection, and the BRAMs are not affected. The defined target area is the same for unhardened and protected processors to normalize the FI among all designs.

One bit-flip is injected at a time. After each injection, the Rocket processor is logically reset, and the application is re-executed. All three benchmarks (MxM, AES, and Qsort) were evaluated during FI. Faults accumulate in the design until a functional failure (i.e., SDC or timeout) is detected. The number of faults accumulated until the failure is recorded, and the FPGA is reprogrammed with the fault-free design. This procedure is repeated until $1,000$ functional failures are collected.

6.1.2.5 Heavy ion testing

The sensitive region of the Xilinx Zynq-7000 (XC7Z020-CLG484 part) can be reached with heavy ions thanks to a sample preparation of the package. Figure 6.4 presents the microscopic view of the device, showing the (a) top surface and (b) transversal section (TAMBARA et al., 2015). The total thickness of the Zynq PL passive layers is $12.87\ \mu$m.

Figure 6.4 – Xilinx Zynq-7000 (XC7Z020-CLG484 part) microscopic view: (a) top surface, and (b) transversal section.

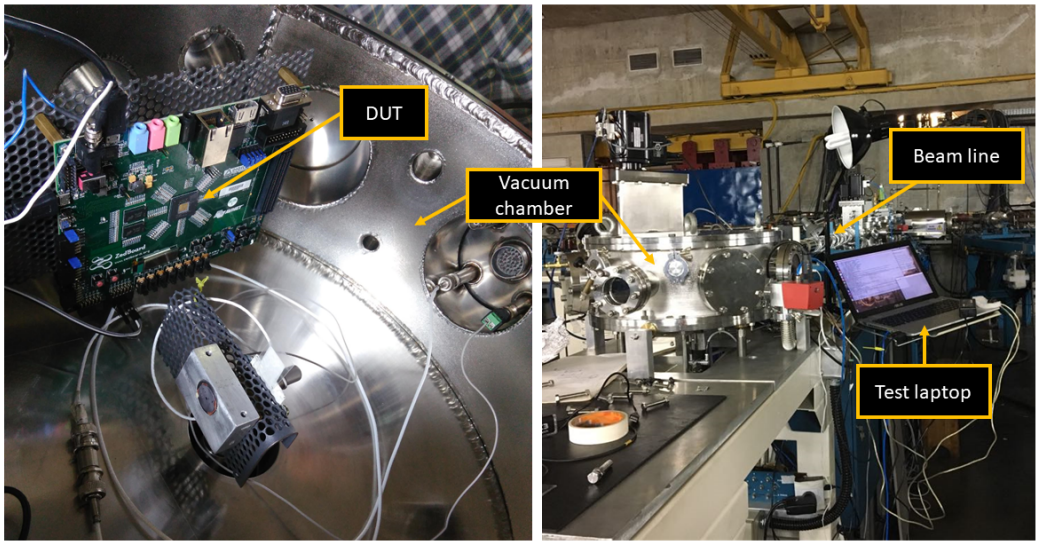


Source: (TAMBARA et al., 2015).

The irradiation tests were conducted through accelerated heavy ions in the 8UD Pelletron accelerator facility at *Laboratório Aberto de Física Nuclear of the Universidade de São Paulo* (LAFN-USP), Brazil (AGUIAR et al., 2014). The beam line complies with the ESA requirements for radiation tests in electronic devices. The heavy ion beam has high uniformity, low-intensity flux, and large area, produced by scattering in a gold foil and defocusing technique (AGUIAR et al., 2014). Figure 6.5 shows the irradiation setup. Figure 6.5(a) is the view inside the vacuum chamber, where the ZedBoard is placed, and the decapsulated Zynq-7000 (DUT) is aligned to the beam. Figure 6.5(b) shows the irradiation room, where one can notice the closed chamber, beam line, and test laptop.

The DUT was irradiated in-vacuum using Oxygen (^{16}O) and Carbon (^{12}C) ions at normal incidence and ambient temperature. The LET and the penetration of the particles are described in table 6.2. The regular beam fluxes ranged from 2.85×10^2 up to 5.28×10^2 $p/cm^2/s$, and the fluence was in the order of 10^6 p/cm^2 per test run. The selected ion fluxes lead to an average upset rate of 5 bit-flips/s in the Zynq PL. This estimation was

Figure 6.5 – Heavy ion testing setup: (a) ZedBoard inside the vacuum chamber, and (b) view of the irradiation room.



Source: From the author.

Table 6.2 – Ions description.

Ion	**LET** ($MeV/mg/cm^2$)	**Penetration** (μm)
^{16}O	6.7	23
^{12}C	3.7	36

performed using static readback tests at the beginning of the experiment. Due to the limited beam time, only the MxM benchmark was executed during the heavy ion testing.

6.1.3 Rocket soft processor under faults

This section assesses the Rocket soft processor under emulation fault injection and heavy ion-induced faults and verifies different fault tolerance methods. A deep discussion is presented based on the analysis and comparison of the results with other relevant soft processors.

6.1.3.1 Unhardened Rocket soft processor

Table 6.3 details the dynamic power and resource usage with the utilization percentage of the unhardened Rocket soft processor implemented in the Zynq PL. The table

Table 6.3 – Dynamic power and resource usage with the utilization percentage of the unhardened Rocket soft processor core and lowRISC SoC implemented in the Zynq-7000 PL.[1,2]

Design		FF	LUT	Carry	BRAM	DSP	Power (mW)[3]
Unhard	Core	$1,701$ (2%)	$4,457$ (8%)	175	0	4 (2%)	7
	SoC	$12,164$ (11%)	$17,329$ (32%)	544	68 (49%)	9 (4%)	37

1. FPGA vendor's tool: Xilinx Vivado 2016.4; Default synthesis and implementation strategies with resource sharing disabled;
2. Target FPGA: Xilinx Zynq-7000 (xc7z020clg484-1) APSoC; Estimated static power ($45°$C junction temperature): 167 mW.
3. Estimated dynamic power of lowRISC SoC and Rocket core: 8 KB L1 cache; no FPU; 20 MHz.

separately describes the required power and area for the processor core and lowRISC SoC.

The following sections present the evaluation results for the emulation fault injection and heavy ion testing.

6.1.3.1.1 *Fault injection results*

Table 6.4 describes the emulation fault injection results for the unhardened Rocket soft processor running each benchmark at 20 MHz. Faults were injected in the FPGA CRAM until accumulating a total of $1,000$ failures. On average, the functional failure results are 21% SDCs and 79% timeouts. The overall error rate is about 0.16 failures per injected bit-flip. The MxM benchmark presents fewer SDCs than AES and Qsort and has the lowest error rate (0.14). The masking effects of each application contribute to different susceptibility results. For example, MxM exercises more the arithmetic units, and the dynamic response of the data can vary by changing the matrices size; the AES can exercise the encryption instructions in the processor; and the Qsort is a type of algorithm that makes control flow decisions based on the input data.

The mean faults to failure is above 6, which means that, on average, a failure occurs at every 6 injected bit-flips. The target area of the injection is normalized by the most oversized test design (triplicated versions presented in the following sections). Therefore, many faults affect unused bits of the unhardened design, not leading to effects in the system. However, the unhardened Rocket soft processor has an increased percentage of failures due to a single bit-flip. About 14% of the failures are due to the first injected fault, demonstrating the core's high susceptibility to single-bit upsets.

Figure 6.6 presents the empiric reliability curves obtained for the unhardened Rocket soft processor running each benchmark. The maximum reliability achieved by the processor is lower than 88% for single bit-flips. Therefore, the application executions

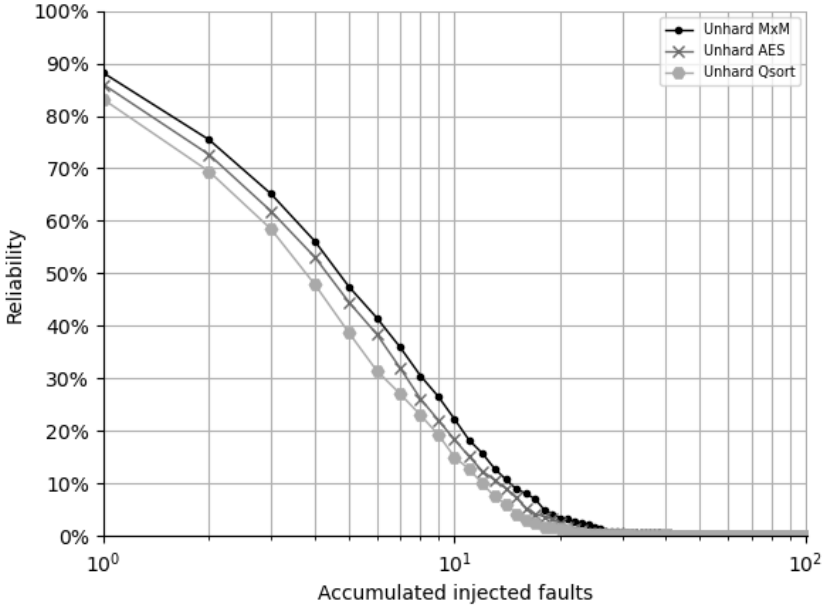
Table 6.4 – Emulation fault injection results for the unhardened Rocket soft processor per application benchmark at 20 MHz.

Benchmark	Total inj. faults	Failures			Error rate[1]	Mean faults to failure[2]	One fault failure[3]
		#SDC	#Timeouts	#Total			
MxM	7,039	159	841	1,000	0.14	7.04	12%
AES	6,381	259	741	1,000	0.16	6.38	14%
Qsort	5,698	223	777	1,000	0.18	5.70	17%
Total	19,118	641	2,359	3,000	0.16	6.37	14%

1. $FI\ error\ rate = total\ failures / total\ inj.\ faults$, as defined in section 5.2.2.
2. $Mean\ faults\ to\ failure = total\ inj.\ faults\ /\ total\ failures$.
3. Percentage of failures that occurred due to the first injected fault (only one single bit-flip in the design).

are up to 88% correct under faults. This result is correlated to *One fault failure* column in table 6.4. The reliability drops at more injected faults. The higher the number of bit-flips accumulated in the design, the higher the probability of failure.

Figure 6.6 – Empiric reliability obtained by emulation fault injection for the unhardened Rocket soft processor running MxM, AES, and Qsort benchmarks at 20 MHz.



Source: From the author.

Single-bit fault injection performed by Ramos, Maestro and Reviriego (2017) in the lowRISC SoC showed that an average of 94.6% of results are correct. The hang rate is around 2.6% among the benchmarks. Depending on the application execution time, the

SDCs can be as lower as 0.19% up to 3.07%. In another work, the Rocket processor presented an average of 3.5% of SDCs, 3.4% of hangs, and 8% of unexpected terminations under single injected faults Cho (2018). The structural differences in the fault injection methods preclude straightforward comparison. The results in this thesis align with the literature by showing that the Rocket soft processor is more prone to faults that crash or lock the core but also presents a high SDC rate. Because the fault injection methodology adopted in this thesis affects only the Rocket Tile area, the results are considered pessimistic, estimating the worst-case scenario in which a single fault is likely to provoke a failure in the execution flow.

6.1.3.1.2 *Heavy ion testing results*

Table 6.5 describes the dynamic cross section and MWBF results from the heavy ion testing of the unhardened Rocket soft processor running MxM benchmark at 20 MHz. The cross section shows the processor susceptibility to SEE-induced faults, and the MWBF demonstrates the amount of application data computed correctly between failures.

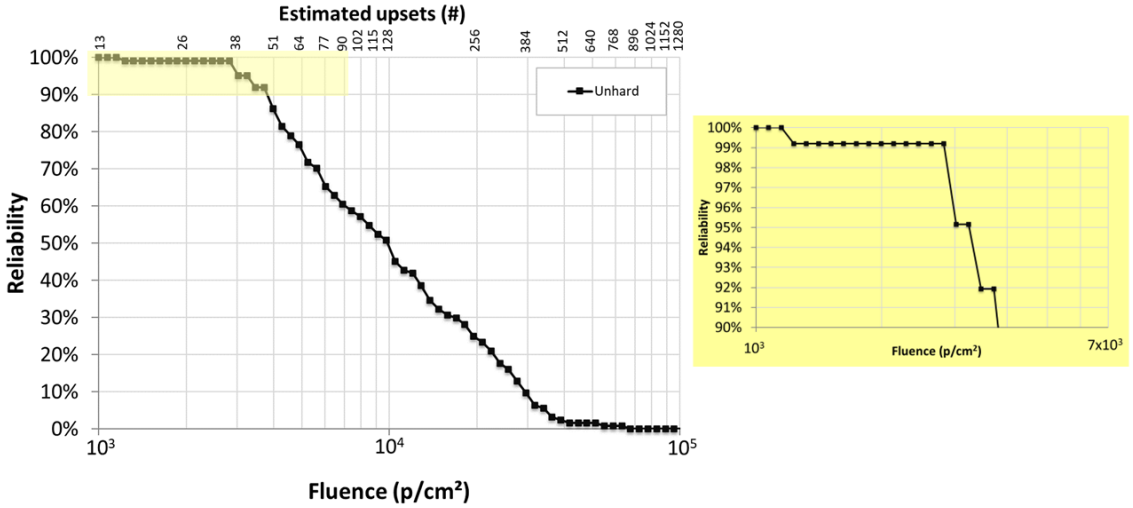
Table 6.5 – Dynamic cross section and MWBF results from heavy ion testing for the unhardened Rocket soft processor running MxM benchmark at 20 MHz.

Beam	**Fluence**	**Cross section** (cm^2)			**MWBF**
	(p/cm^2)	σ**SDC**	σ**Timeout**	σ**Total**	$(data)$
^{16}O	1.76×10^6	3.46×10^{-5}	3.57×10^{-5}	7.03×10^{-5}	1.24×10^6
^{12}C	1.31×10^6	3.50×10^{-5}	2.13×10^{-5}	5.63×10^{-5}	2.30×10^6

The SDC and timeout cross sections are similar for the Oxygen testing, and the Carbon testing presented more SDCs than timeouts. These results diverge from the CRAM fault injection, which shows the processor is more prone to timeouts. The fault injection has the limitation of not affecting all circuitry, and only the Rocket Tile area is the target of faults. Because the entire FPGA is irradiated, SEUs can occur in different elements, as discussed in chapter 3. For instance, soft errors can happen in the entire CRAM, BRAMs, DSPs, and SETs can occur in the logic. All these effects can affect the data flow and lead to SDC failures.

Figure 6.7 presents the empiric reliability curve related to fluence for the unhardened Rocket soft processor irradiated with Oxygen ions. The yellow rectangle highlights the reliability percentage from 100% down to 90%. The static CRAM cross section per device of the Zynq PL is $1.28 \times 10^{-2}\ cm^2$ (TAMBARA et al., 2015). From the cross

Figure 6.7 – Empiric reliability obtained from the Oxygen irradiation of the unhardened Rocket soft processor running MxM benchmark at 20 MHz.



Source: From the author.

section definition (section 5.2.1), the estimated number of upsets in the CRAM can be obtained from the static cross section multiplied by the fluence. The top axis of figure 6.7 shows the estimated number of upsets in the FPGA CRAM for the defined fluence.

The unhardened Rocket soft processor maintained maximum reliability until fluence of around 1.15×10^3 p/cm^2, which is estimated to lead to about 15 bit-flips accumulated in the configuration memory. The reliability drops to 90% with a fluence of 3.98×10^3 p/cm^2 (50 upsets estimated). The unhardened Rocket soft processor has not reached maximum reliability in the FI tests. As previously discussed, the FI targeted a more restricted processor area, leading to a more pessimistic analysis. During heavy ion testing, many upsets can occur in unused elements, not causing any effects. Although the irradiation testing presented better results than the FI, the unhardened Rocket soft processor ensures the correct benchmark execution until low fluence only. The processor implemented in the SRAM-based FPGA is highly susceptible to soft errors and must be protected with fault tolerance techniques to increase the overall system reliability.

6.1.3.2 Rocket soft processor protected by scrubbing

As described in section 4.1.1, configuration memory scrubbing is essential in SRAM-based FPGAs. This thesis uses the SEM-IP (XILINX, 2022b), a built-in scrubber

present in Xilinx FPGAs. The SEM-IP is an internal scrubber implemented in the FPGA fabric and is used to clean bit-flips in the CRAM of the Zynq PL. Its usage resource is not added to the utilization report of the implemented design. The Xilinx user manual states that SEM-IP requires 838 LUTs, 682 FFs, 56 I/Os, and 6 BRAMs for Zynq-7000 XC7Z020 (XILINX, 2022b). The SEM-IP is set to run at 50 MHz. The estimated time for scrubbing the entire CRAM is 16 ms, and the mitigation latency for a single upset is about 19 ms (XILINX, 2022b). The design version with scrubbing enabled is labeled with "*_S*". Since no protection is added to the Rocket core, the design is defined as an unhardened core with scrubbing (*Unhard_S*).

The SEM-IP uses the ICAP configuration interface to monitor the CRAM frames. The emulation fault injection also requires the ICAP interface to flip bits in the CRAM. Because of the ICAP usage concurrence, it is not possible to simultaneously perform fault injection and enable the SEM-IP. Therefore, this thesis does not perform FI in designs with scrubbing enabled.

6.1.3.2.1 *Heavy ion testing results*

Table 6.6 presents the results for dynamic cross section and MWBF, and figure 6.8 shows the empiric reliability for the Rocket soft processor design without and with scrubbing. Results from the unhardened core (*Unhard*) are repeated from figure 6.7 to facilitate comparison.

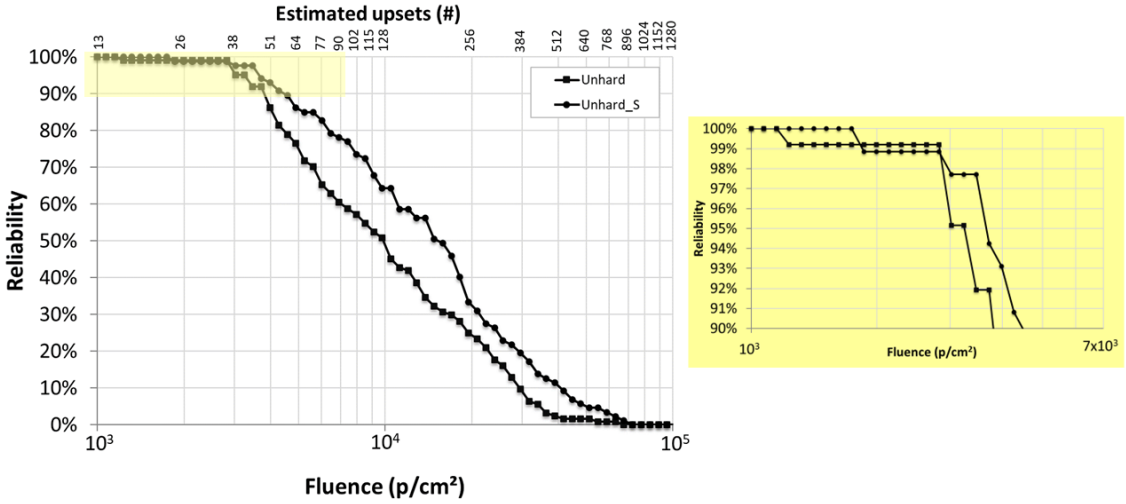
One can notice that enabling the scrubbing reduces the cross section, increases the MWBF, and improves the reliability. The correctness is ensured until a fluence of $1.74 \times 10^3\ p/cm^2$, which is about 22 bit-flips in the CRAM. However, only enabling the CRAM scrubbing does not significantly reduce the overall Rocket soft processor susceptibility.

For these results, the SDC cross section has a more significant reduction than the timeout cross section when adding scrubbing. The scrubbing technique can cope with the accumulation of bit-flips in the FPGA configuration memory, but SETs can still propagate in the logic, and upsets in critical bits can still lead to failures. The error build-up occurs when the upset effect propagates through the system, either when the affected circuitry is used or by its reflection on dynamic elements. If the scrubber controller happens to inspect the erroneous CRAM frame fast enough before the propagation, the error build-up may be avoided. Data errors may be related to the temporal characteristics of the CRAM bit-flip. For instance, if the scrubbing fixes the circuitry before the need to perform computations or transfer data through the data path, the upset does not reflect in SDC. On the other

Table 6.6 – Dynamic cross section and MWBF results from heavy ion testing for the unhardened Rocket soft processor without and with scrubbing running MxM benchmark at 20 MHz.

Beam	**Design**	**Fluence**	**Cross section** (cm^2)			**MWBF**
		(p/cm^2)	σ**SDC**	σ**Timeout**	σ**Total**	$(data)$
^{16}O	Unhard	1.76×10^6	3.46×10^{-5}	3.57×10^{-5}	7.03×10^{-5}	1.24×10^6
	Unhard_S	1.71×10^6	2.34×10^{-5}	2.75×10^{-5}	5.09×10^{-5}	1.80×10^6
^{12}C	Unhard	1.31×10^6	3.50×10^{-5}	2.13×10^{-5}	5.63×10^{-5}	2.30×10^6
	Unhard_S	1.38×10^6	1.09×10^{-5}	1.09×10^{-5}	2.17×10^{-5}	5.27×10^6

Figure 6.8 – Empiric reliability obtained from the Oxygen irradiation of the unhardened Rocket soft processor without and with scrubbing running MxM benchmark at 20 MHz.



Source: From the author.

hand, if the error propagates and is caught by a flip-flop of the control state machine, a SEFI may occur due to the loss of the control flow. Error masking techniques can be applied in order to reduce the number of critical bits in the design and reduce the error build-up. The following sections present the results of triplication strategies.

6.1.3.3 Rocket soft processor protected by TMR

Two TMR granularity strategies are tested on the Rocket soft processor: Coarse Grain TMR (CGTMR) and Fine Grain Distributed TMR (FDTMR). The CGTMR involves triplicating the Rocket core as a block and voting the outputs bit by bit. In the FDTMR, the submodules Rocket core are triplicated. The hardened netlists are gener-

ated by an automated tool based on Cadence's EDA flow (BENITES; KASTENSMIDT, 2018). The automatic tool is used to implement the TMR design of the Rocket core because of the complexity of its Verilog code, which is automatically generated from the Chisel Scala language. A manual triplication would be time-consuming and prone to human implementation errors. The automated tool, however, presents the limitation of not triplicating carry logic and DSPs. All arithmetic operations are implemented in triplicated logic. Therefore, this can increase the LUT usage more than expected for TMR designs.

Only the Rocket core is triplicated due to the limited size of the Zynq PL, and voters are also triplicated to avoid single points of failure. Xilinx recommends the implemented design should use about 75% of theFPGA resources since a higher area increases the complexity of placement and timing closure (XILINX, 2021b). The designs were built targeting that recommendation. Table 6.7 describes the resource utilization and estimated dynamic power of the CGTMR and FDTMR versions of the Rocket soft processor and shows the unhardened version for comparison.

Table 6.7 – Dynamic power and resource usage with the utilization percentage of the unhardened, CGTMR, and FDTMR Rocket soft processor core and lowRISC SoC implemented in the Zynq-7000 PL.[1,2]

Design		**FF**	**LUT**	**Carry**	**BRAM**	**DSP**	**Power** $(mW)^3$
Unhard	Core	$1,701$ (2%)	$4,457$ (8%)	175	0	4 (2%)	7
	SoC	$12,164$ (11%)	$17,329$ (32%)	544	68 (49%)	9 (4%)	37
CGTMR	Core	$11,036$ (10%)	$26,556$ (49%)	0	0	0	51
	SoC	$21,495$ (20%)	$39,232$ (74%)	369	68 (49%)	5 (2%)	82
FDTMR	Core	$9,756$ (9%)	$29,706$ (56%)	0	0	0	78
	SoC	$20,215$ (19%)	$42,382$ (80%)	369	68 (49%)	5 (2%)	108

1. FPGA vendor's tool: Xilinx Vivado 2016.4; Default synthesis and implementation strategies with resource sharing disabled;
2. Target FPGA: Xilinx Zynq-7000 (xc7z020clg484-1) APSoC; Estimated static power ($45°$C junction temperature): 167 mW.
3. Estimated dynamic power of lowRISC SoC and Rocket core: 8 KB L1 cache; no FPU; 20 MHz.

The lowRISC SoC featuring the original FDTMR applied to the Rocket core required more slice LUT resources (i.e., $55,041$) than are available in the Zynq PL (i.e., $53,200$). As a result, design optimization was enabled in the Vivado tool during the implementation phase, leading to sharing of resources. The area of the FDTMR design presented in table 6.7 is after the optimization. As described above, all the arithmetic and logic operations are implemented in LUTs for the triplicated designs. Therefore, versions

with a TMR Rocket core do not use any DSP or carry chain.

One can notice in table 6.7 that the triplicated Rocket core requires about $6\times$ (CGTMR) and $6.7\times$ (FDTMR) more LUTs than the unhardened core. The needed resources are more than three times due to the automatic tool only inferring LUTs for arithmetic and logic operations instead of carry and DSPs. Moreover, all voters are also triplicated. The FDTMR without optimization would require much more resources.

Regarding dynamic power consumption, applying a TMR leads to an overhead of $7\times$ and $11\times$ for the CGTMR and FDTMR core, respectively. However, it is worth noticing that the power values are an estimation from the Xilinx Vivado tool, which might not reflect the actual consumption figures. For a more appropriate power analysis, voltage and current monitoring circuitry should be added to the test setup to measure consumption accurately.

The best solution for a more robust system enhancement would be to triplicate the entire lowRISC SoC, not using optimizations, and add synthesis primitives to avoid shared resources. However, this would require a high-density FPGA device.

6.1.3.3.1 *Fault injection results*

The CGTMR Rocket core is tested under emulation fault injection running the MxM, AES, and Qsort benchmarks at 20 MHz. The FDTMR Rocket soft processor was not evaluated under fault injection because its version and the FI engine combined in the design required more LUT resources than those available in the Zynq PL.

Table 6.8 describes the FI results per benchmark of the CGTMR Rocket soft processor. The resulting failures are distributed at 92% of timeouts and 8% of SDCs in general. SDCs are less frequent than in the unhardened core because of the masking effects of the triplicated design. The CGTMR has a error rate of about 0.18 failures per upset, requiring less than 6 faults to experience a failure. This failure performance is worst than the unhardened Rocket core.

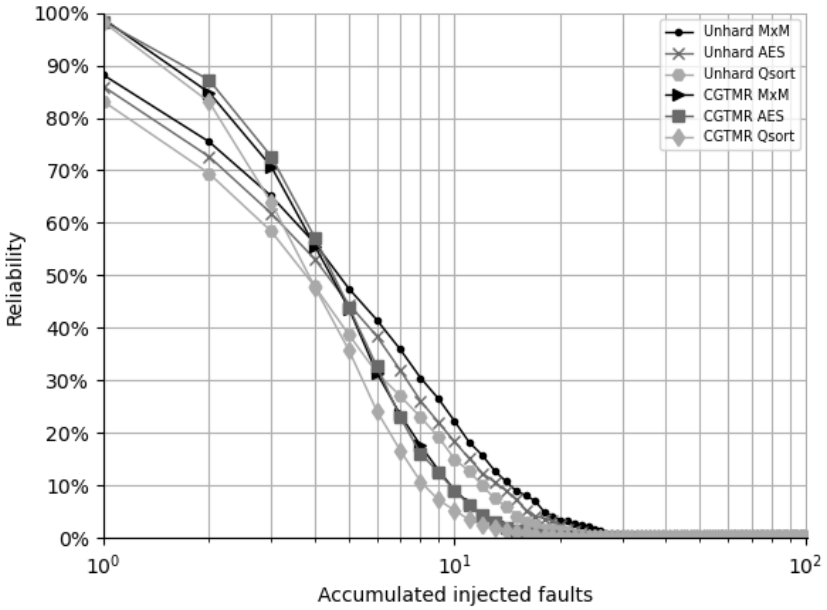
As demonstrated in section 4.1.2.1, the MTTF of a non-redundant system is better than a TMR without repair. A triplicated design is more susceptible to upsets due to the larger exposed area. Since upsets accumulate over time, the triplication masking effects are overcome, and the TMR becomes more vulnerable. There needs to be more than the triplication alone to cope with faults on a long time basis, and a repair method should be applied, such as scrubbing.

Table 6.8 – Emulation fault injection results for the CGTMR Rocket soft processor per application benchmark at 20 MHz.

Benchmark	Total inj. faults	Failures			Error rate[1]	Mean faults to failure[2]	One fault failure[3]
		#SDC	#Timeouts	#Total			
MxM	5,691	49	951	1,000	0.18	5.69	1.2%
AES	5,722	86	914	1,000	0.17	5.72	1.6%
Qsort	5,071	112	888	1,000	0.20	5.07	1.8%
Total	16,484	247	2,753	3,000	0.18	5.49	1.5%

1. $FI\ error\ rate = total\ failures / total\ inj.\ faults$, as defined in section 5.2.2.
2. $Mean\ faults\ to\ failure = total\ inj.\ faults\ /\ total\ failures$.
3. Percentage of failures that occurred due to the first injected fault (only one single bit-flip in the design).

Figure 6.9 – Empiric reliability obtained by emulation fault injection for the unhardened and CGTMR Rocket soft processor running MxM, AES, and Qsort benchmarks at 20 MHz.



Source: From the author.

Analyzing the *One fault failure* column in table 6.8, one can see that the CGTMR is highly efficient in reducing the soft processor susceptibility to single faults. The reduction is about 10% compared to the unhardened Rocket core. This improvement is demonstrated in the empiric reliability curve in figure 6.9, where the maximum reliability is 98.8%. The CGTMR does not achieve maximum correctness for single-bit faults because the emulation fault injection targets the entire Rocket Tile area and not only the

triplicated core. Moreover, cross-domain errors can also occur in the clocking and routing system, for instance.

6.1.3.3.2 *Heavy ion testing results*

Table 6.9 describes the dynamic cross section and MWBF results from the heavy ion testing of the CGTMR and FDTMR Rocket soft processor running MxM benchmark at 20 MHz. The unhardened core results are added for comparison. A modest improvement is noticed for the FDTMR processor core. However, the CGTMR presents worst results than the unhardened core, as indicated in the FI analysis, since the design is without scrubbing and faults accumulate over time.

Table 6.9 – Dynamic cross section and MWBF results from the heavy ion testing for the unhardened, CGTMR, and FDTMR Rocket soft processor running MxM benchmark at 20 MHz.

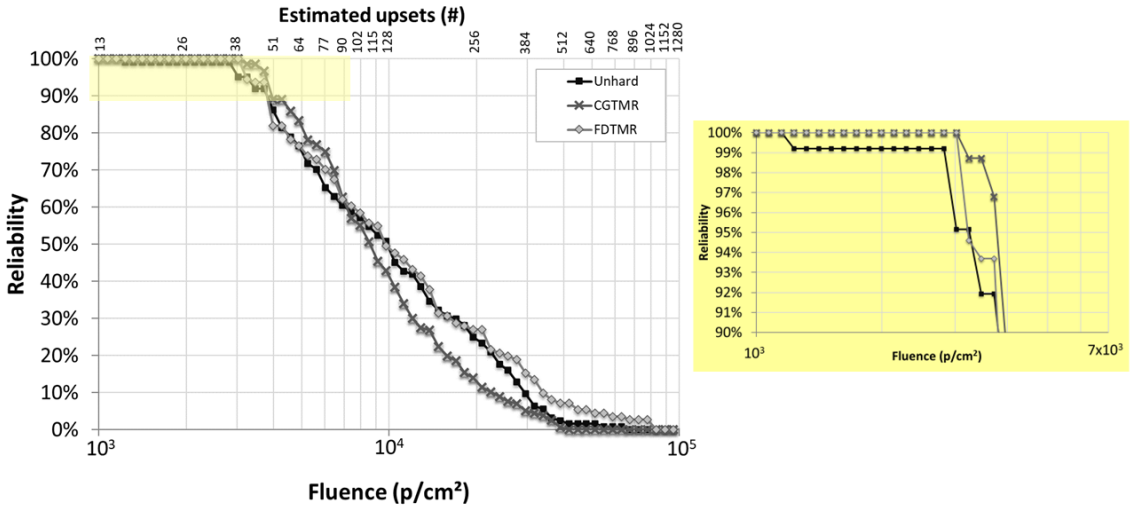
Beam	**Design**	**Fluence**	**Cross section** (cm^2)			**MWBF**
		(p/cm^2)	σ**SDC**	σ**Timeout**	σ**Total**	$(data)$
^{16}O	Unhard	1.76×10^6	3.46×10^{-5}	3.57×10^{-5}	7.03×10^{-5}	1.24×10^6
	CGTMR	1.84×10^6	2.88×10^{-5}	5.59×10^{-5}	8.47×10^{-5}	9.83×10^5
	FDTMR	1.83×10^6	2.73×10^{-5}	3.33×10^{-5}	6.05×10^{-5}	1.40×10^6
^{12}C	Unhard	1.31×10^6	3.50×10^{-5}	2.13×10^{-5}	5.63×10^{-5}	2.30×10^6
	FDTMR	1.20×10^6	1.58×10^{-5}	1.91×10^{-5}	3.49×10^{-5}	3.77×10^6

Figure 6.10 presents the empiric reliability curves for the designs tested using Oxygen ions. The CGTMR and FDTMR Rocket cores improve about 3 times the reliability compared to the unhardened version. The designs are able to maintain the correctness for more bit-flips than in the FI tests due to the target area of faults. The FI focuses the injections in an active area of the design, whereas the full FPGA is affected by the heavy ions, and most of the upsets might occur in an unused area. Nonetheless, the higher the fluence (i.e., the more accumulated faults), the least reliable the TMR.

6.1.3.4 Rocket soft processor protected by combined fault tolerance techniques

As shown in the previous sections, the scrubbing and TMR fault tolerance techniques can improve the system reliability but, individually, are not highly efficient in reducing the susceptibility in designs implemented in SRAM-based FPGAs. A combined approach is expected to enhance the soft processor further.

Figure 6.10 – Empiric reliability obtained from the heavy ion test campaign with the unhardened, CGTMR, and FDTMR Rocket soft processor running MxM benchmark at 20 MHz.



Source: From the author.

This section brings the results of the CGTMR and FDTMR Rocket soft processor with the CRAM scrubbing enabled. A watchdog timer is also evaluated for timeout detection. The watchdog is controlled by the ARM Cortex-A9 in the Zynq PS and reprograms the PL whenever a timeout occurs. Rocket soft processor with watchdog enabled is labeled with "*_T*". Designs with scrubbing and watchdog enabled are labeled with "*_ST*".

The resource usage of the designs is the same as in table 6.7. As previously described, the FI is limited to designs without scrubbing. Therefore, the combined fault tolerance techniques are evaluated under heavy ions only.

6.1.3.4.1 *Heavy ion testing results*

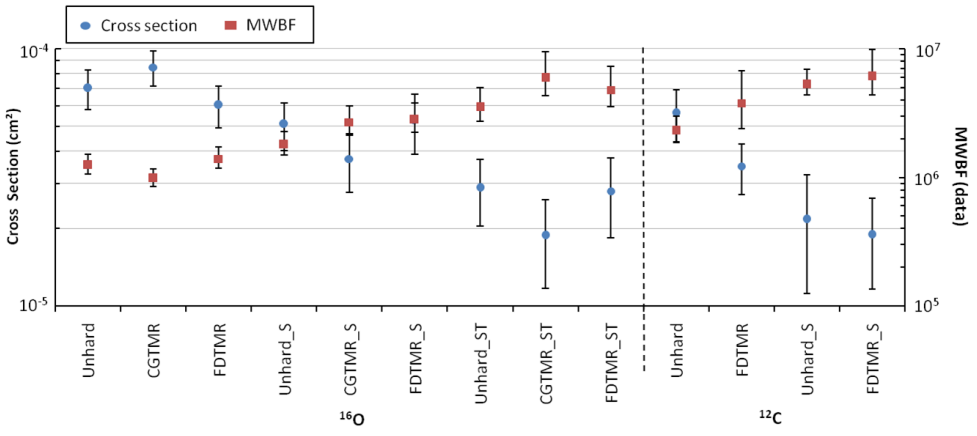
Table 6.10 describes the dynamic cross section and MWBF results for combined fault tolerance techniques applied to the Rocket soft processor. Results from previous sections are repeated for comparison. Figure 6.11 graphically shows the results in terms of total dynamic cross section (left axis) and MWBF (right axis) with a 95% confidence interval, as defined in the ESCC25100 (ESA, 2014). A reliable design should present a lower cross section, and a higher MWBF.

The cross section can be reduced up to three times, and the MWBF is improved more than two times when TMR and scrubbing are combined in a design. Surprisingly, the FDTMR results are relatively less expressive than expected. That might be a con-

Table 6.10 – Dynamic cross section and MWBF results from the heavy ion testing for the unhardened and protected Rocket soft processor running MxM benchmark at 20 MHz.

Beam	**Design**	**Fluence**	**Cross section** (cm^2)			**MWBF**
		(p/cm^2)	σ**SDC**	σ**Timeout**	σ**Total**	$(data)$
^{16}O	Unhard	1.76×10^6	3.46×10^{-5}	3.57×10^{-5}	7.03×10^{-5}	1.24×10^6
	CGTMR	1.84×10^6	2.88×10^{-5}	5.59×10^{-5}	8.47×10^{-5}	9.83×10^6
	FDTMR	1.83×10^6	2.73×10^{-5}	3.33×10^{-5}	6.05×10^{-5}	1.40×10^6
	Unhard_S	1.71×10^6	2.34×10^{-5}	2.75×10^{-5}	5.09×10^{-5}	1.80×10^6
	CGTMR_S	1.56×10^6	1.41×10^{-5}	2.31×10^{-5}	3.73×10^{-5}	2.68×10^6
	FDTMR_S	1.06×10^6	2.27×10^{-5}	3.02×10^{-5}	5.29×10^{-5}	2.81×10^6
	Unhard_ST	1.56×10^6	2.69×10^{-5}	1.92×10^{-6}	2.88×10^{-5}	3.52×10^6
	CGTMR_ST	1.44×10^6	1.81×10^{-5}	6.96×10^{-7}	1.88×10^{-5}	5.93×10^6
	FDTMR_ST	1.18×10^6	2.63×10^{-5}	1.70×10^{-6}	2.80×10^{-5}	4.79×10^6
^{12}C	Unhard	1.31×10^6	3.50×10^{-5}	2.13×10^{-5}	5.63×10^{-5}	2.30×10^6
	FDTMR	1.20×10^6	1.58×10^{-5}	1.91×10^{-5}	3.49×10^{-5}	3.77×10^6
	Unhard_S	1.38×10^6	1.09×10^{-5}	1.09×10^{-5}	2.17×10^{-5}	5.27×10^6
	FDTMR_S	1.37×10^6	1.09×10^{-5}	8.00×10^{-6}	1.89×10^{-5}	6.09×10^6

Figure 6.11 – Total dynamic cross section and MWBF results from heavy ion testing for unhardened and protected Rocket soft processor running MxM benchmark at 20 MHz.



Source: From the author.

sequence of the design oversimplification during the synthesis optimization phase. As aforementioned, some of the optimization attributes were not disabled due to the limited resources of the Zynq PL.

The improvement of scrubbing combined with TMR may be low due to the accelerated flux of particles used during heavy ion testing. Results are considered pessimistic since the expected particle rate is much lower in a natural radiation environment, which would give more time for the scrubbing to clean single-bit faults. In the case of multiple-bit errors in the same CRAM frame, the SEM-IP halts due to an uncorrectable error, and the system is left unprotected until FPGA reprogramming. Another characteristic of the SEM-IP is resuming the scrubbing cycle from the top CRAM frame when any single-bit upset is detected and corrected. A high upset rate may lead to the bottom frames of the CRAM being less frequently scrubbed. Additionally, the scrubber engine is internal to the FPGA and susceptible to soft errors. Therefore, in a natural environment, the improvement from the combination of scrubbing and triplication is expected to be much higher than the presented results.

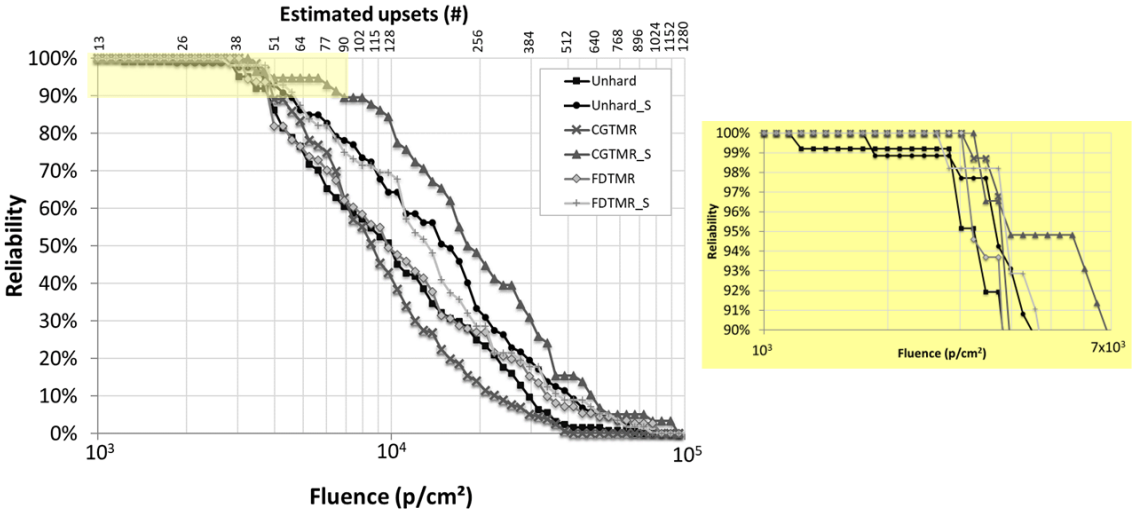
The designs with scrubbing and watchdog presented a reduction of more than one order of magnitude in the timeout cross section. The CGTMR_ST has a timeout reduction of more than 51 times compared to the unhardened design, and the MWBF is improved almost five times. The timeout monitoring is performed by the ARM Cortex-A9 for validation purposes only. Instead of using a processor, a more suitable approach would be implementing a low-resource monitor in a dedicated triplicated RTL module.

Figure 6.12 shows the empiric reliability curves for the unhardened and protected Rocket soft processor. Designs with TMR and scrubbing presented a reliability three times higher than the unhardened design. The *CGTMR_S* sustained a high level of reliability until $3.24 \times 10^3\ p/cm^2$, which represents 41 estimated bit-flips in the configuration memory. Nonetheless, these upsets are not accumulated since the scrubbing is enabled.

The *CGTMR_S* performs about the same as the *CGTMR* (no scrub version) under single-bit faults. As previously discussed, scrubbing does not avoid the error build-up for upsets on critical bits. In triplicated designs, the critical bits are those cross-domain bits related, for instance, to clocking, routing, or shared resources due to optimizations. The essential and most effective scrubbing impact is cleaning faults in time, avoiding accumulation. One can notice the designs with scrubbing enabled are the most reliable at high fluence.

It is essential to mention the penalty of the unhardened design for the MWBF results. To execute the tests under the same conditions, the unhardened Rocket runs at 20 MHz frequency only because of the performance limitation of the triplicated designs. The unhardened Rocket running at a higher frequency is expected to improve the MWBF.

Figure 6.12 – Empiric reliability obtained from the heavy ion testing for the unhardened and protected Rocket soft processor running MxM benchmark at 20 MHz.



Source: From the author.

The following section investigates that hypothesis.

6.1.3.5 Influence of the processor frequency in the soft error susceptibility

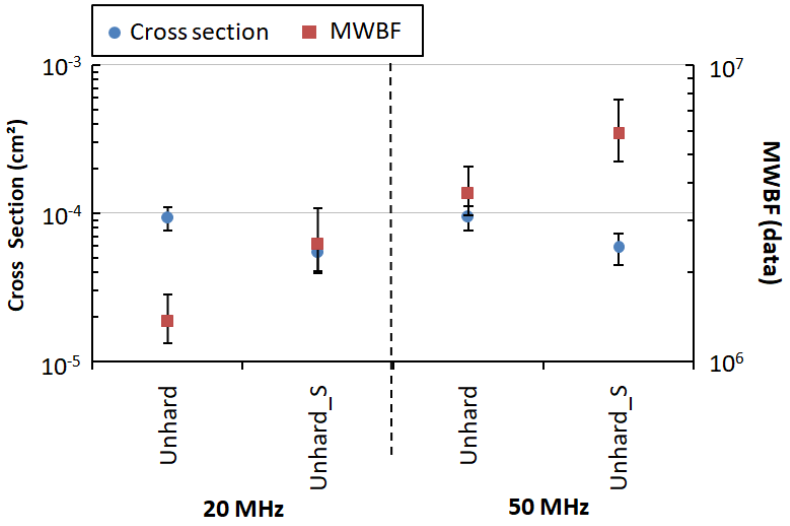
This section assesses two different design frequencies under heavy ions aiming to verify the impact of processor speed on the susceptibility. Results from the Rocket processor running at 20 and 50 MHz are analyzed for unhardened Rocket core without and with scrubbing. This analysis is not performed using triplicated designs because their maximum frequency limit is 20 MHz.

6.1.3.5.1 *Heavy ion testing results*

Figure 6.13 shows the total dynamic cross section and MWBF for the unhardened Rocket processor without and with scrubbing running MxM benchmark at 20 and 50 MHz. Results from 20 MHz are repeated from previous sections for comparison.

Increasing the clock frequency improves the MWBF up to 2.7 times while slightly affecting the system cross section. The higher the processor speed, the faster the application is executed, and consequently, more data is computed correctly before experiencing a failure. However, higher clock frequencies might affect the cross section significantly. Therefore, the trade-off cross section and performance must be considered.

Figure 6.13 – Total dynamic cross section and MWBF results from heavy-ions experiments for the unhardened Rocket processor without and with scrubbing running MxM benchmark at 20 and 50 MHz.



Source: From the author.

One can notice that the gain in MWBF is higher than the frequency gain (i.e., 2.7× vs. 2.5×). This difference can occur due to some factors. Although a pre-defined flux is set per run, fluctuations are common in particle accelerators, and the average flux of particles is expected to differ between runs, as mentioned in section 6.1.2.5. As shown in equations 5.3 and 5.12 (chapter 5), the cross section computation uses the fluence, which masks this effect, but the MWBF uses the average flux directly. The application execution time is also computed as an average of multiple executions since the values can vary some milliseconds during the time measurement. The MxM execution time for the 20 MHz is 745.87 ms (described in table 6.1), and the computed time for the 50 MHz is 283.43 ms. Round numbers can also add noise during calculations. All that justify the more than expected increase in the MWBF for higher frequencies, and error bars are a guide to cover the data uncertainty.

6.1.3.6 Discussion

As expected, fault injection and heavy ions results show that scrubbing is essential for SRAM-based FPGA designs. Scrubbing prevents the accumulation of upsets in the CRAM by recovering the correct bitstream configuration. Additionally, the periodic reset performed in all designs restores the system to a known state. Nonetheless, results demonstrated that the unhardened Rocket soft processor core is highly susceptible to single bit-flips, and additional user-level mitigation techniques are required to mask or

correct faults.

Although combining TMR and scrubbing was expected to substantially increase the system's fault tolerance, only modest improvements were observed in the results. One reason is the accelerated testing which presents a much higher upset rate than natural irradiation environments, leading to pessimistic results since the scrubbing fails to cope with all CRAM bit-flips. Another reason may be that only the processor core was triplicated, and essential parts of the SoC were still vulnerable to soft errors, such as the caches and peripherals.

Cache memories are susceptible elements in processors. TAMBARA et al. (2015) demonstrated that errors in unprotected L1 caches highly impact the processor cross section. To increase reliability, caches can be protected with EDAC approaches. The peripherals and buses can also be triplicated to avoid single points of failure. Errors in the unprotected caches and peripherals may have masked the real improvement of the triplicated Rocket core. Moreover, cross-domain single points of failures may have been introduced due to the design simplification in the FPGA synthesis tool. One should check the final design to ensure no simplification on the critical path.

Table 6.11 shows SEE testing results for soft processors embedded into SRAM-based FPGAs. Despite the differences in testing methodology and processors' configurations and architectures, one can observe the cross section improvements in adding mitigation methods to the system. The heavy ions results show the Rocket with a lower cross section than the Cortex-M0, which is the opposite expected from the sensitive results presented on the Harward et al. (2015) work. However, the benefits from TMR and scrubbing are more visible in the Cortex-M0. The Cortex-M0 design has better improvement because the core and all the external SoC elements are triplicated, which reinforces the importance of protecting all resources (BENITES et al., 2019). The LEON3 has the highest cross section reduction, in which the entire SoC is triplicated (KELLER; WIRTHLIN, 2017). Additionally, an improvement of $50\times$ is achieved when a BRAM memory scrubbing is used in the LEON3 soft processor. Due to the lower error rate of Neutrons experiments, the scrubber can clean the configuration memory effectively while avoiding many accumulated upsets simultaneously. When combined with the masking factor of TMR, the scrubber leads to a high impact in the cross section reduction.

Table 6.11 – Cross section comparison of soft processors implemented in SRAM-based FPGAs from this thesis and literature.

Soft processor	**SEE testing**	**Cross section** (cm^2)	
		Unhardened	**TMRbest + Scrubbing (Improvement)**
Rocket (lowRISC)[1]	Heavy Ions - Zynq-7000	7.03×10^{-5}	1.89×10^{-5} (3×)
Cortex-M0[2]	Heavy Ions - Zynq-7000	1.14×10^{-4}	9.19×10^{-5} (4.5×)
LEON3[3]	Neutrons - Kintex-7	2.61×10^{-9}	9.68×10^{-11} (27×)

1. Results from this thesis.
2. (BENITES et al., 2019).
3. (KELLER; WIRTHLIN, 2017).

6.2 L1 cache susceptibility investigation

Based on the observed results from the irradiation experiments of the Rocket soft processor presented in the previous section, there was a need for further investigation of the L1 cache influence on the processor error rate. This thesis investigates the SEU sensitiveness of the processor based on the L1 cache topology, evaluating the impacts related to memory size and associativity. The system assessment is performed on the Rocket soft processor in the Zynq-7000 APSoC under emulation fault injection on the BRAM memories. The analysis covers the failure distribution per L1 data and instruction cache. Additionally, we evaluate the reliability improvement by the protection of periodic scrubbing of the user memories.

6.2.1 Investigation methodology

The platform setup is the same as presented in section 6.1.2.1, except that the injection engine was adapted to inject upsets in the BRAMs of the L1 instruction and data cache. The investigation uses the unhardened Rocket core, and faults only affect the L1 cache memory.

The 32×32 MxM benchmark, described in section 6.1.2.2, is executed at 20 MHz. The benchmark program instruction size is about 60 KB, and the computed data is roughly 12 KB (three 32×32 matrices of 32-bit words). During fault injection, the Rocket events are defined as in section 6.1.2.3: UNACE, SDC, and timeout.

6.2.1.1 L1 cache configurations

This thesis investigates different set-associative cache topologies on the Rocket soft processor implemented in the Zynq-7000. The lowRISC SoC presented in section 6.1 has a 8 KB cache, with 4-way associativity and 32 sets of 64 bytes per block cache. For this investigation, the L1 cache size varies from 1 KB to 16 KB, and the cache topology is exercised through different sets and ways. The L1 cache size is computed per equation 6.1.

$$L1\ cache\ size = \#bytes\_per\_block \times \#sets \times \#ways \tag{6.1}$$

Where:

- $\#bytes\_per\_block$: length of each cache block in bytes;
- $\#sets$: number of locations the cache is divided into, and in which blocks can be mapped and stored;
- $\#ways$: number of blocks each set can store.

The Rocket processor uses 64 bytes per block cache. The new configurations were generated by updating the RocketChip Scala code and then re-generating the verilog of the Rocket processor. Table 6.12 describes the architecture of the various L1 cache configurations tested. The architecture of the L1 instruction and data (I+D) memories is divided into memories that store data (e.g., application data or program instructions) and memories that are part of the cache control (e.g., tag information).

To demonstrate the L1 cache architecture described in table 6.12, we can look further at the 16k_s32w8 configuration. For this configuration, 16 KB of instruction cache and 16 KB of data cache are instantiated using 32 sets and 8 ways each. The data memory of the instruction and data caches (I+D data memory) is implemented in 8 BRAMs of 256 addresses, and each address includes a 64-bit word. Therefore, in total the I+D data memory requires: $(8 \times 256 \times 64) + (8 \times 256 \times 64) = 262,144$ bits. A similar logic applies to the I+D control memory, which uses $19,456$ bits.

6.2.1.2 Emulation fault injection

The emulation fault injection engine is a hardware module similar to the FI previously described. The fault injection targets the BRAMs implementing the Rocket L1 cache. The BRAM elements were updated from the default single-port to true dual-port

Table 6.12 – Architecture of different Rocket L1 cache configurations implemented in the Zynq-7000.

L1 config.	I+D size (KB)	#Sets	#Ways	I+D data memory		I+D control memory	
				Architecture	Size (bits)	Architecture	Size (bits)
16k_s32w8	$16 + 16$	32	8	8×256 64-bit words + 8×256 64-bit words	$262,144$	8×256 4-bit words + 1×32 168-bit words + 1×32 184-bit words	$19,456$
16k_s64w4	$16 + 16$	64	4	4×512 64-bit words + 4×512 64-bit words	$262,144$	4×512 4-bit words + 1×64 80-bit words + 1×64 88-bit words	$18,944$
8k_s32w4	$8 + 8$	32	4	4×256 64-bit words + 4×256 64-bit words	$131,072$	4×256 4-bit words + 1×32 84-bit words + 1×32 92-bit words	$9,600$
8k_s64w2	$8 + 8$	64	2	2×512 64-bit words + 2×512 64-bit words	$131,072$	2×512 4-bit words + 1×64 40-bit words + 1×64 44-bit words	$9,472$
4k_s16w4	$4 + 4$	16	4	4×128 64-bit words + 4×128 64-bit words	$65,536$	4×128 4-bit words + 1×16 88-bit words + 1×16 96-bit words	$4,992$
4k_s32w2	$4 + 4$	32	2	2×256 64-bit words + 2×256 64-bit words	$65,536$	2×256 4-bit words + 1×32 42-bit words + 1×32 46-bit words	$4,864$
2k_s8w4	$2 + 2$	8	4	4×64 64-bit words + 4×64 64-bit words	$32,768$	4×64 4-bit words + 1×8 92-bit words + 1×8 100-bit words	$2,560$
2k_s16w2	$2 + 2$	16	2	2×128 64-bit words + 2×128 64-bit words	$32,768$	2×128 4-bit words + 1×16 44-bit words + 1×16 48-bit words	$2,496$
1k_s4w4	$1 + 1$	4	4	4×32 64-bit words + 4×32 64-bit words	$16,384$	4×32 4-bit words + 1×4 96-bit words + 1×4 104-bit words	$1,312$
1k_s8w2	$1 + 1$	8	2	2×64 64-bit words + 2×64 64-bit words	$16,384$	2×64 4-bit words + 1×8 46-bit words + 1×8 50-bit words	$1,280$

memories (i.e., two independent access ports). The processor accesses one port, and the injector engine accesses another. This strategy allows non-intrusive fault injection in the BRAMs, without interfering with the application execution flow.

Bit-flips are randomly inserted during the application execution, considering the number of clock cycles required to finish the application. The execution time is shorter or longer depending on the L1 cache topology. The execution time per configuration is reported later in the results section. To inject a fault at a random application clock cycle, the FI engine reads a random memory word, flips the random bit, and writes back the word with the inverted bit. A single-bit fault is injected per application execution, and faults do

not accumulate over time. The Rocket soft processor is reset between executions.

The injections are normalized by area. Table 6.13 shows the target BRAM area used for all L1 cache configurations. The target area was inferred considering the highest number of sets and ways between the topologies. One million faults were injected per L1 cache configuration. Due to the temporal characteristics of the user memories, in which the application naturally refreshes the data, a large number of upsets need to be injected to gather a significant number of failures. The injections were divided by affecting only the I+D data memories or the I+D control memories. Therefore, it is possible to assess their impact on reliability individually.

Table 6.13 – Normalized memory area used for fault injection targeting all L1 cache configurations.

#Sets	**#Ways**	**I+D data memory**		**I+D control memory**	
		Architecture	**Size (bits)**	**Architecture**	**Size (bits)**
64	8	8×512 64-bit words + 8×512 64-bit words	$524,288$	8×512 4-bit words + 2×64 184-bit words	$39,936$

6.2.2 Results

A preliminary fault injection targeting the FPGA configuration memory was performed targeting the Rocket processor tile, similar to the one executed previously but varying the L1 cache topology. The goal was to assess the susceptibility of the cache controller depending on the configuration. It was observed that changing the L1 cache topology has no or low impact on the design error rate, which was about 0.16. The CRAM upsets have a high probability of leading to errors. The influence of the cache topology is negligible since the size of the cache controller does not vary much. For those reasons, the emulation fault injection presented in this analysis only targets the BRAMs.

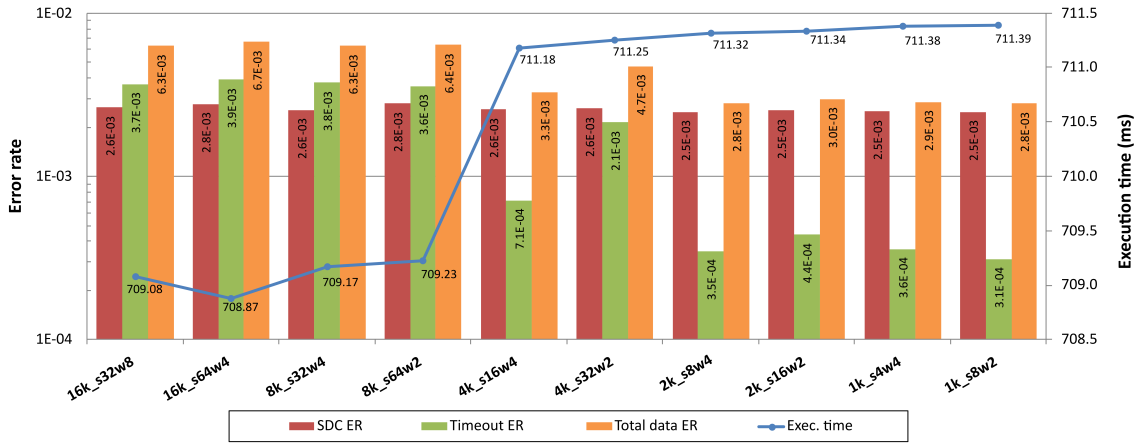
6.2.2.1 Data memory error rate

Table 6.14 shows the results for the emulation fault injection targeting the BRAMs of the L1 I+D data memories. This table describes the observed failures and the total error rate for the data memories. Figure 6.14 details the SDC, timeout, and total error rate, and shows the MxM application execution time per cache configuration.

Table 6.14 – Emulation fault injection on the Rocket L1 instruction and data cache: targeting I+D data memories.

L1 config.	Total inj. faults	Failures			Total data error rate
		#SDC	#Timeout	#Total	
16k_s32w8	1×10^6	2,631	3,681	6,312	6.31×10^{-3}
16k_s64w4	1×10^6	2,766	3,902	6,668	6.67×10^{-3}
8k_s32w4	1×10^6	2,557	3,752	6,309	6.31×10^{-3}
8k_s64w2	1×10^6	2,785	3,578	6,363	6.36×10^{-3}
4k_s16w4	1×10^6	2,565	712	3,277	3.28×10^{-3}
4k_s32w2	1×10^6	2,600	2,138	4,738	4.74×10^{-3}
2k_s8w4	1×10^6	2,472	348	2,820	2.82×10^{-3}
2k_s16w2	1×10^6	2,526	440	2,966	2.97×10^{-3}
1k_s4w4	1×10^6	2,493	359	2,852	2.85×10^{-3}
1k_s8w2	1×10^6	2,472	312	2,784	2.78×10^{-3}

Figure 6.14 – I+D data memory error rate (bars related to the left axis) and application execution time (line related to the right axis).



Source: From the author.

As expected, a smaller L1 cache size leads to longer execution time, and larger sizes improve performance. A more significant step is observed between 4 KB and 8 KB. These performance results are related to the MxM benchmark running on the processor, and different applications may vary the exercise of the L1 cache. Applications that process more significant amounts of data are expected to lead to longer execution time differences between cache sizes.

The methodology of the emulation fault injection leads to pessimistic results for large cache sizes since it always injects one fault per application execution. The injection methodology does not consider that larger caches execute the application faster, which should reduce the probability of faults in time. The injections are normalized in the area but not in time, which penalizes more resource-consuming configurations.

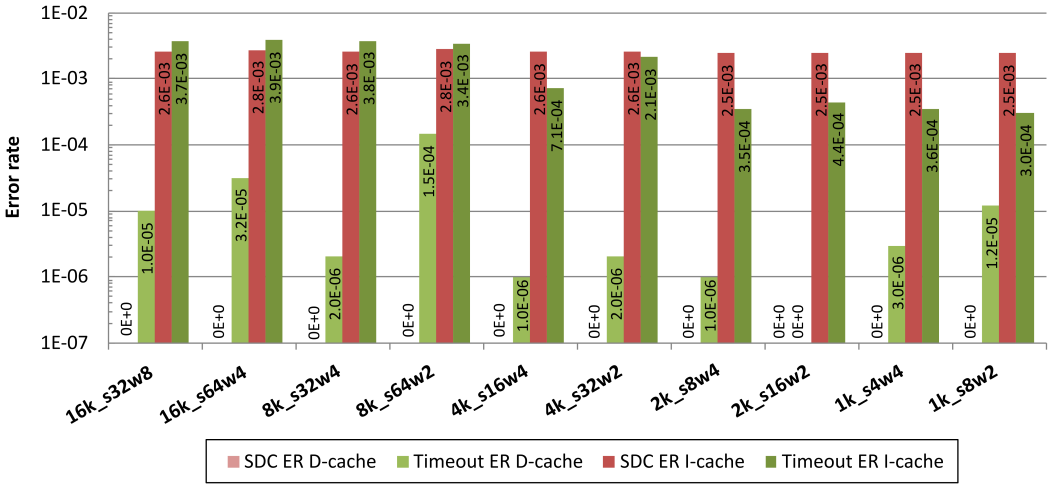
The SDC error rate is similar over the tested L1 cache configurations, which demonstrates the high probability of application outputs errors caused by upsets in the user memories. The timeout error rate is higher for larger caches. The 16k_s64w4 configuration is 12.5 times worst than the 1k_s8w2. Since small caches have a high refresh rate (high miss rate), the time window of a valid data/instruction is short, and the memory is more often re-written. On large caches, more memory positions are available and the data/instruction can be re-used multiple times, which increases the valid time window and, therefore, increases the vulnerability. Ko et al. (2017) demonstrated the data vulnerability over the different cache accesses. Corrupted data can affect the processor execution only at a reading operation or if the is written back to the main memory.

An unexpected timeout error rate is observed for 4k_s32w2 compared to a similar configuration. The fault injection experiment was re-executed with similar results. This outcome might be due to the 4 KB being an intermediate size using a high number of sets.

Figure 6.15 presents the error rate per instruction and data cache over the different configurations. This error rate is computed by the number of failures (i.e., SDCs or timeouts) originating from the instruction or data cache divided by the total injected bit-flips (i.e., one million). The majority of the failures are due to upsets on the instruction cache. Few timeouts and no SDCs were observed from data cache bit-flips, showing the higher vulnerability of the instruction cache. As discussed in section 3.1.3, soft errors in the instruction cache may lead to data or control flow errors. If a corrupted instruction is validated, the processor can execute an incorrect instruction or generate a trap.

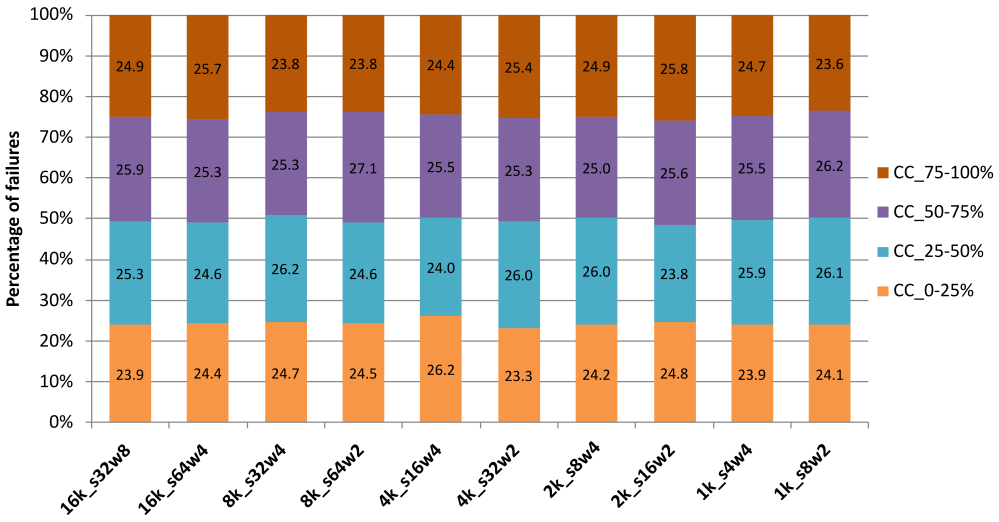
Figure 6.16 shows the distribution of failures related to the time window in which the fault was injected during the application execution. The graph gives the percentage of failures by injections on the application's range of clock cycles (c.c.). Failures are observed evenly throughout the application in all configurations, and there is no distinguished critical time window. The well-distributed failures in the application indicate no dependency between faults.

Figure 6.15 – Error rate for the data memories per instruction and data cache over the different configurations.



Source: From the author.

Figure 6.16 – Percentage of failures related to the application time window (in c.c.).



Source: From the author.

Similarly to the simulated results from the related works (Yuanwen Huang; Mishra, 2016; LIVANY; SALEHI; KARGAR, 2020), the emulation fault injection in BRAMs shows that increasing the cache size of the Rocket soft processor increases the vulnerability. However, contrary of the (LIVANY; SALEHI; KARGAR, 2020) study, the Rocket's instruction cache is more prone to errors. The differences between the vulnerability computation and the actual error build-up during fault injection might lead to these different results. The application benchmark characteristics also influence the way the cache is exercised. Although about 12 KB of data is computed in total for the MxM, the data is loaded on demand and processed in lines, so not all data is vulnerable at the same time.

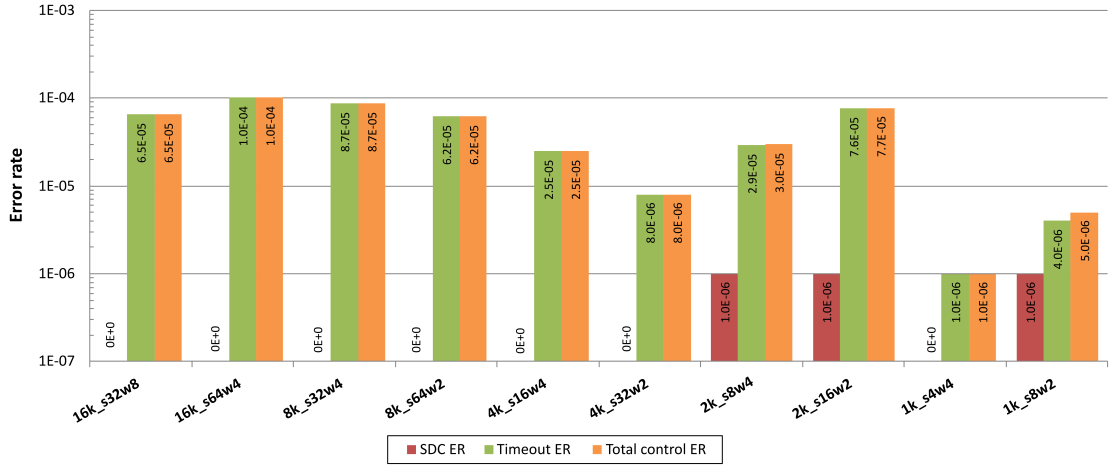
6.2.2.2 Control memory error rate

Table 6.15 describes the failure events for the emulation fault injection targeting the BRAMs of the L1 I+D control memories, and figure 6.17 presents the error rates per cache configuration. Results show a low probability of failure due to upsets in the control memories. Only a few events were observed, primarily timeouts. Most bit-flips were masked due to the inherent characteristics of the control memories of defining data location and status. As discussed in section 3.1.3, faults in the tag array can invalidate an instruction/data, leading to a cache miss, or validate the erroneous instruction/data, which could cause an error.

Table 6.15 – Emulation fault injection on the Rocket L1 instruction and data cache: targeting I+D control memories.

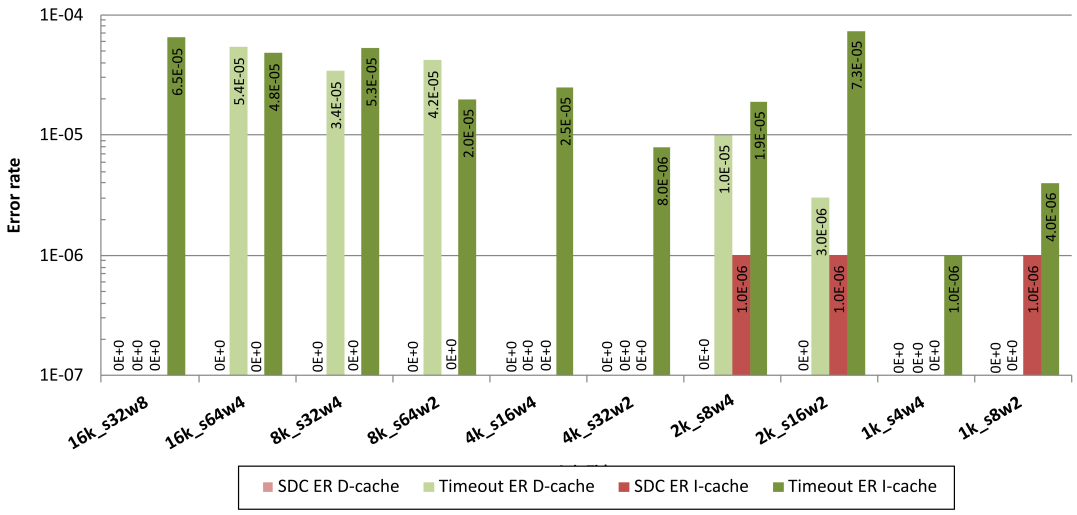
L1 config.	**Total**	**Failures**			**Total control**
	inj. faults	**#SDC**	**#Timeout**	**#Total**	**error rate**
16k_s32w8	1×10^6	0	65	65	6.50×10^{-5}
16k_s64w4	1×10^6	0	102	102	1.02×10^{-4}
8k_s32w4	1×10^6	0	87	87	8.70×10^{-5}
8k_s64w2	1×10^6	0	62	62	6.20×10^{-5}
4k_s16w4	1×10^6	0	25	25	2.50×10^{-5}
4k_s32w2	1×10^6	0	8	8	8.00×10^{-6}
2k_s8w4	1×10^6	1	29	30	3.00×10^{-5}
2k_s16w2	1×10^6	1	76	77	7.70×10^{-5}
1k_s4w4	1×10^6	0	1	1	1.00×10^{-6}
1k_s8w2	1×10^6	1	4	5	5.00×10^{-6}

Figure 6.17 – I+D control memory error rate per cache configuration.



Source: From the author.

Figure 6.18 – Error rate for the control memories per instruction and data cache over the different configurations.



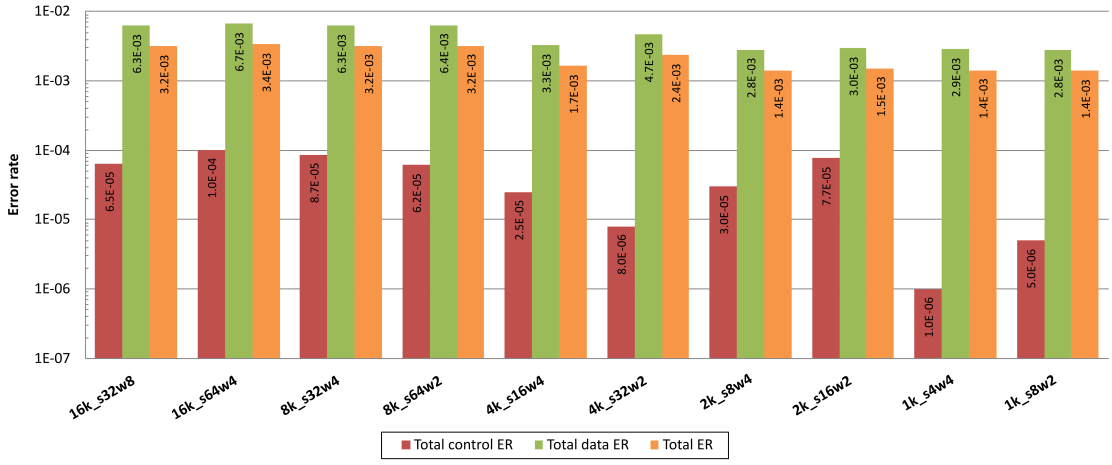
Source: From the author.

Similar to the data memory results from the previous section, the 1 KB cache also presents fewer failures. Figure 6.18 shows the error rates per instruction and data cache. Upsets in the instruction cache lead to more timeouts over the configurations, but bit-flips in the data cache also present a similar timeout rate for some topologies.

6.2.2.3 Total error rate

Figure 6.19 shows the total error rate considering data and control memories of the L1 instruction and data cache. The error rate for the data and control memories are from the previous sections. The total error rate (Total ER) is computed by summing the

Figure 6.19 – Total error rate for I+D data and control memories per cache configuration.



Source: From the author.

number of failures on both tests and dividing by the total injected faults (i.e., two million faults). The total error rate of the 16k_s64w4 is about 2.4 worst than the 1k_s8w2. The difference in error rate between cache topologies of the same size is negligible for larger caches. For smaller caches, it can be noticed that higher associativity (i.e., more ways) presents a slightly better error rate.

6.2.2.4 Instruction cache protected by periodic flush

As described in previous sections, upsets in the instruction cache are more likely to lead to failures, and data memories are more susceptible than control memories. This thesis, therefore, investigates the use of periodic flush on data memories of the L1 instruction cache to reduce the soft error vulnerability. Different flush periods are tested, ranging from 100 clock cycles delay between refreshes to $100,000$ clock cycles.

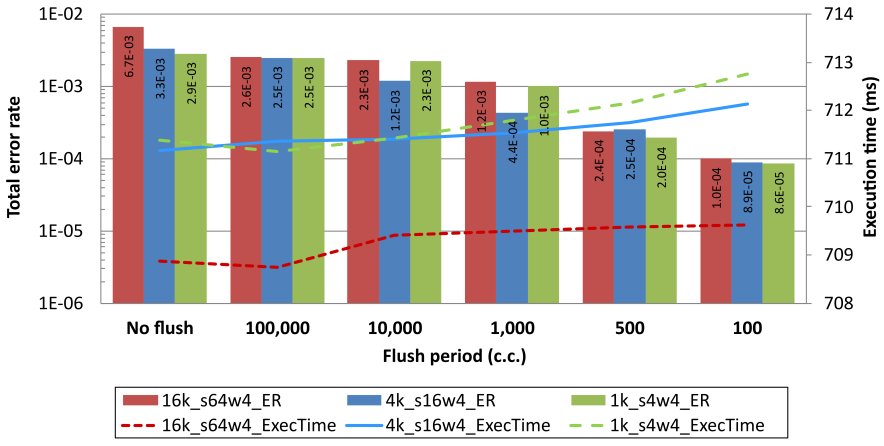
Table 6.16 describes the emulation fault injection results per cache configuration. Due to the number of combinations, only three cache topologies are tested: 16k_s64w4, 4k_s16w4, and 1k_s4w4. The selected topologies exercise the corners and intermediate sizes tested using 4-way associativity.

Figure 6.20 shows the total error rate per cache topology and flush period. The execution time is also described in this figure. For comparison, results from the configurations with no flush are repeated from section 6.2.2.1. As expected, increasing the flush rate impacts the application performance in all topologies, and the more frequent the refresh, the lower the error rate.

Table 6.16 – Emulation fault injection on the Rocket L1 instruction and data cache: target data memories protected with periodic flush.

L1 config.	Flush Period (c.c.)	Total inj. faults	Failures			Total error rate
			#SDC	#Timeout	#Total	
16k_s64w4	100,000	1×10^6	2,481	76	2,557	2.56×10^{-3}
	10,000	1×10^6	2,223	69	2,292	2.29×10^{-3}
	1,000	1×10^6	1,137	39	1,176	1.18×10^{-3}
	500	1×10^6	232	10	242	2.42×10^{-4}
	100	1×10^6	70	30	100	1.00×10^{-4}
4k_s16w4	100,000	1×10^6	2,429	58	2,487	2.49×10^{-3}
	10,000	1×10^6	1,159	39	1,198	1.20×10^{-3}
	1,000	1×10^6	406	32	438	4.38×10^{-4}
	500	1×10^6	231	22	253	2.53×10^{-4}
	100	1×10^6	57	32	89	8.90×10^{-5}
1k_s4w4	100,000	1×10^6	2,430	55	2,485	2.49×10^{-3}
	10,000	1×10^6	2,222	54	2,276	2.28×10^{-3}
	1,000	1×10^6	991	23	1,014	1.01×10^{-3}
	500	1×10^6	181	14	195	1.95×10^{-4}
	100	1×10^6	47	39	86	8.60×10^{-5}

Figure 6.20 – Total error rate (bars related to the left axis) and execution time (lines related to the right axis) per cache topology and flush period.



Source: From the author.

The 1 KB and 4 KB cache topologies with the flush period of 100 clock cycles present a reduction in the error rate of more than 30 times compared to their versions without refresh. The most significant improvement comes with the 16 KB L1 cache for a flush period of 100 clock cycles that reaches almost 67 times of error rate reduction. The

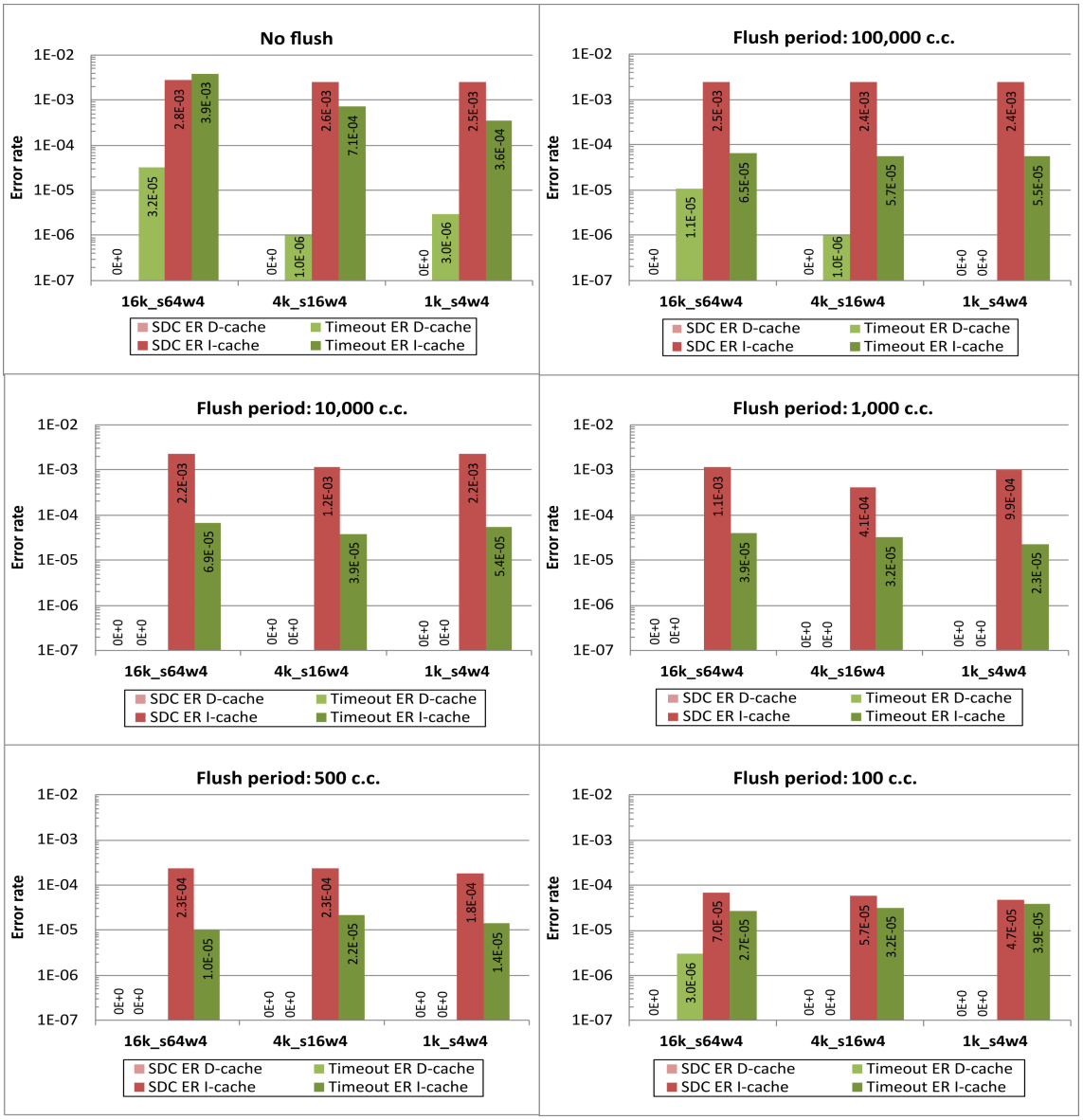
16k_s64w4 presents the combined advantage of low vulnerability with faster application execution.

Figure 6.21 presents the error rate per L1 instruction and data cache over the different configurations and flush periods. Despite the L1 data cache not being refreshed, the periodic flush of the instruction cache impacts the timing of the input vectors that exercise the data cache, leading to few or no failures originating from these memories.

The timeout error rate due to instruction cache upsets is similar for flushed variants. Additionally, SDCs are the most frequent failures. A hypothesis is the error latency for SDCs is of few clock cycles and many cycles for timeouts. The refresh to be efficient must occur during the error latency period (time from flipping the bit until a cache read, for instance).

The error rate improvement of adding periodic flush is observed in all cache sizes and shows an opportunity for using larger caches. The error rate of the 16 KB cache is similar to 1KB at a more frequent refresh. The impact on performance is also lower for larger caches. Therefore, one can take advantage of the faster application execution of a larger cache with a similar error rate of a small memory. The next chapter investigates that hypothesis in the NOEL-V soft processor under proton irradiation.

Figure 6.21 – Error rate for the data memories per instruction and data cache over the different configurations and flush periods.



Source: From the author.

6.3 Discussion

Understanding the error distribution and how it affects the processor execution is essential to implement the most suitable mitigation techniques. As demonstrated by Cho (2018), the most critical faults are the ones affecting the register file, CSR, and the integer pipeline of the Rocket processor. Therefore, the designer can apply mitigation focusing on those elements, such as ECC on the register file and triplicating the pipeline. Alternatively, a lockstep method can be implemented for the processor architecture enhancement, similar to the lockstep mode of ARM Cortex R5 (ARM, 2011).

Nonetheless, as discussed in section 3.1.3 and demonstrated by experiments in section 6.1.3, the core is not the only SEE-sensitive part in a soft processor. The Rocket soft processor with only the core triplicated shows a limited improvement in fault tolerance. Results from section 6.2 show that the L1 cache is also vulnerable to errors, and the instruction cache is highly susceptible. Increasing the L1 cache size also increases the vulnerability, but adding periodic flush can be an alternative to reduce the error rate.

Fault mitigation and correction methods should be extensively applied to all SoC elements, and the more heterogeneous they are, the better. Therefore, a heterogeneous hybrid solution that mixes hardware and software protection may be a better approach for SoCs with complex soft processors. At the hardware level, a dedicated RTL watchdog module can be used to monitor timeouts and trigger an FPGA reprogramming when required. The processor core enhancement may rely on TMR or lockstep, and the peripherals should also be triplicated. An ECC or parity method may protect the register file and on-chip memories. CRAM scrubbing must be performed to avoid the accumulation of faults, and periodic reset can reestablish the system state. A SIHFT technique can be applied at the software level to deal with SDCs. Moreover, increasing the processor speed is an alternative to achieve better MWBF without affecting the cross section significantly. Commercial soft processors with built-in fault tolerance are also available in the market and can be considered, such as the fault tolerant NOEL-VFT (FRONTGRADE GAISLER, 2022b).

Real scenarios may have restrictions on the available resources, such as constraints in the design area, power consumption, or a limited financial budget. For instance, the required FPGA might not allow full triplication of the design. As shown in section 6.1.3.3, the possibility of applying a full TMR to the lowRISC SoC was excluded due to the limited area of the Zynq-7000 PL. As a solution, only the Rocket core was triplicated.

Moreover, optimizations were enabled to the FDTMR design due to area restrictions. Although the strategy made a level of protection possible, it came with the cost of a less expressive improvement in reliability.

The following chapter intends to answer some open questions from the previous experiments. It characterizes a RISC-V NOEL-V soft processor employing a distributed TMR with feedback voters, using another automated tool for the netlist triplication to avoid the over-inferred logic. We also investigate the different cache sizes and protection under proton testing.

7 EXPLORING THE COTS RISC-V NOEL-V SOFT PROCESSOR UNDER RADIATION EFFECTS

This chapter demonstrates the mitigation process using different fault tolerance techniques on the COTS open source RISC-V NOEL-V processor synthesized into the SRAM-based Zynq UltraScale+ FPGA. The SEE characterization is performed under proton testing, and further investigation is done through emulation fault injection. We assess the effect of the cache size on the reliability of the processor and the use of scrubbing combined with triplication, duplication with comparison, and cache refreshing.

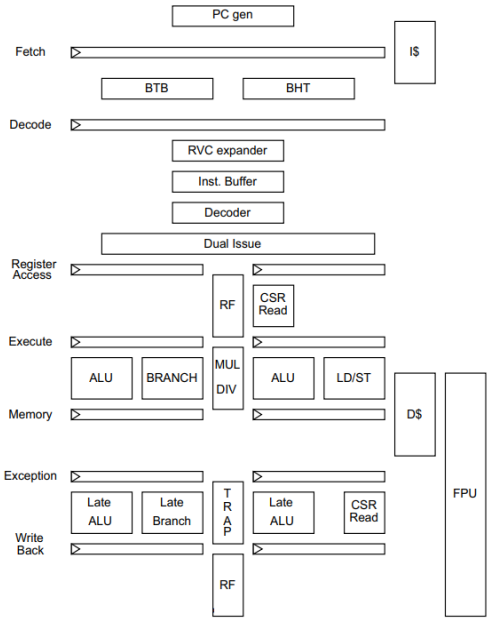
7.1 COTS RISC-V NOEL-V soft processor

The NOEL-V is a state-of-the-art soft processor based on the RISC-V architecture (FRONTGRADE GAISLER, 2022b). The NOEL-V soft processor can be synthesized as a 32-bit (RV32) or a 64-bit (RV64) model, and its IP subsystem allows configurations ranging from a lite controller to high performance. The in-order pipeline can be implemented with either a single- or dual-issue. Figure 7.1 shows the implementation of the 7-stage dual-issue integer pipeline. The NOEL-V features advanced branch prediction capabilities, a cache controller supporting high throughput, an optional MMU integrated into the cache controller, and an Advanced Microcontroller Bus Architecture (AMBA) 2.0 AMBA High-performance Bus (AHB) bus interface.

In this chapter, we use the NOEL-V soft processor as well as the IP subsystem from the COTS open source non-protected version of the GRLIB IP Library[1] (FRONTGRADE GAISLER, 2022b). As a case study, a lite configuration of the NOEL-V was selected due to its reduced size. The selected NOEL-V is a 32-bit in-order single-issue processor implementing an RV32IM RISC-V ISA. The NOEL-V core contains the IU, ALU, multiplier, divider, dynamic branch prediction, branch target buffer, register file, L1 instruction and data cache, and cache controller. By default, this configuration has a 1 KB L1 cache. The L1 cache usage and size vary depending on the test design, as further describer in section 7.2.2. The NOEL-V SoC features an AMBA interface with AHB and Advanced Peripheral Bus (APB), AHB memory bridge to AXI, debug unit, interrupt controller, peripherals such as UART, and GPIO.

[1] GNU General Public License (GPL) - reference grlib-gpl-2022.1-b4272.

Figure 7.1 – NOEL-V processor 7-stage dual-issue integer pipeline.



Source: (FRONTGRADE GAISLER, 2022b).

7.2 Investigation methodology

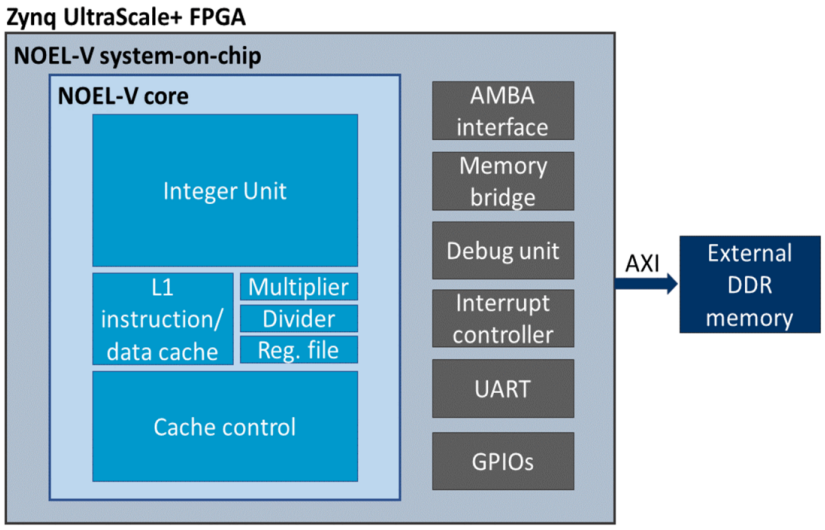
7.2.1 Platform setup

The DUT is the COTS Xilinx Zynq UltraScale+ Multi-Processor System-on-Chip (MPSoC), which is built on the 16 nm FinFET technology (XILINX, 2022). Similar to the Zynq-7000 device, whose tests are described in chapter 6, the Zynq UltraScale+ comprises a PS and PL. The PS is an SoC around hard core ARM Cortex-A53 and Cortex-R5F processors, and the PL is the FPGA fabric.

The test board is the Avnet Ultra96-V2 development board (AES-ULTRA96-V2-G) (AVNET, 2021) that features a Zynq UltraScale+ XCZU3EG-1SBVA484E chip (ZU3EG). The ZU3EG chip is a bare-die flip-chip Ball Grid Array (BGA), with a thickness of around 800 μm to the sensitive region of Silicon, and the die area is about 16.29×11.04 mm (Xilinx, 2022). Because of the substrate thickness and the non-availability of sample preparation, the irradiation testing is performed using a proton beam since its penetration can reach many microns in the device.

Unlike the test setup presented in chapter 6, we decided not to use any ARM processor controlling the NOEL-V testing. Despite the ASIC PS being more robust to radiation than the FPGA fabric, soft errors can still occur. Therefore, we avoid PS errors being misinterpreted as FPGA errors. Instead, the control of the experiments is performed

Figure 7.2 – NOEL-V SoC implemented into the Zynq UltraScale+ FPGA.



Source: From the author.

via Python scripts, and the monitoring of the NOEL-V design is via the GRMON3 debug tool[2] (FRONTGRADE GAISLER, 2022c) through an UART link.

Another improvement of this test setup compared to the one presented in chapter 6 is the external DDR memory as the boot memory. Previously, the boot memory was implemented in BRAMs that, although triplicated, were also exposed to radiation. Figure 7.2 shows the NOEL-V SoC implemented in the Xilinx Zynq UltraScale+ FPGA. The NOEL-V SoC characteristics are presented in section 7.1. The processor's register file and L1 cache are implemented in BRAMs.

7.2.2 Mitigating the COTS NOEL-V soft processor

This thesis studies different combinations of mitigation solutions. The investigation focuses on the effect of the L1 cache size on the processor reliability and the use of scrubbing combined with TMR, DWC, and cache refreshing. The characteristics of the used techniques are described below.

7.2.2.1 Scrubbing

The Xilinx SEM-IP is used for the FPGA CRAM scrubbing. As discussed in the previous chapters, scrubbing is mandatory in SRAM-based FPGAs due to its partic-

[2]Evaluation/academic version (reference grmon-eval-64-3.2.15.1).

ular configuration memory vulnerability. Therefore, the SEM-IP is enabled in all tested designs to correct bit-flips periodically. The SEM-IP scrubbing frequency is 50 MHz.

7.2.2.2 TMR for the central processing unit

A distributed TMR is applied to the NOEL-V SoC using the open source SpyDr-Net TMR tool (BYU CONFIGURABLE COMPUTING LAB., 2020). The design netlist is triplicated with feedback voters for error correction in the flip-flop's feedback logic. The BRAMs are not triplicated in the design. The TMR protection had to be restricted to the NOEL-V processor core due to the limited resources available in the Zynq UltraScale+ ZU3EG FPGA.

Two versions of fine grain distributed TMR designs are implemented:

- TMRcore: the TMR is applied to the entire NOEL-V core, including the IU, multiplier and divider elements, and cache controller. All the logic and flip-flops around the register file and L1 cache are triplicated.
- TMRiucctrl: the TMR is applied to the IU and cache controller only, which are the most resource-consuming elements of the NOEL-V core. The resources not included in the IU and cache controller components are not triplicated. This version is an alternative workaround related to the few resources available in the Zynq UltraScale+ ZU3EG FPGA.

7.2.2.3 L1 cache protection

As previously discussed, cache memories affect the processor's soft error susceptibility. The impact of the L1 cache configuration and associativity on the Rocket soft processor has been assessed in chapter 6.2. To progress the investigation on the NOEL-V processor, we propose an analysis of the tradeoff between performance gain and susceptibility by using the L1 cache. Different configurations are used for this analysis:

- Cache usage: The NOEL-V processor is tested with L1 cache enabled and disabled.
- Cache size: the impact of the cache size is evaluated for the NOEL-V processor featuring a 1 KB or 8 KB L1 instruction and data cache. Increasing the cache size also increases the resource usage of the cache controller.
- Cache fault tolerance (cFT): L1 cache with DWC implementation. All the BRAMs of cache memory are duplicated. A checker verifies the instruction and data mem-

ory consistency at every access. The checker is a combinational logic and does not affect the processor execution in a faulty-free execution. In case of data mismatch, the entire L1 cache is flushed.

- Periodic flush: a periodic flush is performed every 100 ms to refresh the L1 cache contents.

The NOEL-V L1 cache is implemented in unprotected BRAMs for all designs.

7.2.2.4 Designs description overview

Table 7.1 describes all design versions of the NOEL-V SoC synthesized into the Zynq UltraScale+ FPGA. The configurations vary the L1 cache enabled and disabled, L1 cache size in 1 KB and 8 KB, and implemented protection. The L1 cache can be enhanced using DWC with a flush in case of errors or a periodic flush at every 100 ms. The distributed TMR is applied to the entire NOEL-V core or to the IU and cache controller only. All designs are with scrubbing enabled. Note that versions with the L1 cache disabled are still with the 1 KB L1 cache memory instantiated in the design since the NOEL-V core does not provide support for removing the cache.

Table 7.1 – NOEL-V test designs description.

NOEL-V SoC design	**L1 cache**			**Distributed TMR**
	Size	**Status**	**Protection**	
NV_1KBdis	1 KB	DIS	NO	NO
NV_TMRcore_1KBdis	1 KB	DIS	NO	NOEL-V core
NV_1KB	1 KB	EN	NO	NO
NV_TMRcore_1KB	1 KB	EN	NO	NOEL-V core
NV_cFT1KB	1 KB	EN	DWC + flush (if mismach)	NO
NV_TMRcore_cFT1KB	1 KB	EN	DWC + flush (if mismach)	NOEL-V core
NV_TMRiucctrl_cFT1KB	1 KB	EN	DWC + flush (if mismach)	IU/cache controller
NV_Flush1KB	1 KB	EN	Periodic flush (every 100 ms)	NO
NV_TMRcore_Flush1KB	1 KB	EN	Periodic flush (every 100 ms)	NOEL-V core
NV_8KB	8 KB	EN	NO	NO
NV_TMRiucctrl_8KB	8 KB	EN	NO	IU/cache controller
NV_cFT8KB	8 KB	EN	DWC + flush (if mismach)	NO
NV_TMRiucctrl_cFT8KB	8 KB	EN	DWC + flush (if mismach)	IU/cache controller
NV_Flush8KB	8 KB	EN	Periodic flush (every 100 ms)	NO
NV_TMRiucctrl_Flush8KB	8 KB	EN	Periodic flush (every 100 ms)	IU/cache controller

7.2.3 Software benchmark

A bare-metal 32-bit data 100×100 Matrix Multiplication (MxM) application is used to evaluate the NOEL-V SoC designs. The MxM benchmark is one of the standard applications extensively used in SEE testing processors and FPGA designs (QUINN et al., 2015). Large matrix multiplication is used for highly exercising the L1 cache under evaluation. The application is written in C programming language and is originally from the MiBench benchmark suite (GUTHAUS et al., 2001). The application's code has been updated for a higher exercise of the data range, so the matrices are initialized with data randomly generated through a PRNG at every execution. The PRNG computation is expected to have low interference in the execution of the benchmarks and not affect the susceptibility results. A 16-bit size checksum is generated from the resulting matrix for data verification.

The application reporting is performed via the GRMON3 debug tool. At every execution, the processor boots from the DDR memory, loads the L1 cache (if enabled), initializes the 100×100 matrices, runs the MxM, computes the checksum, and reports the result. After the application result is checked, a new execution starts. No processor reset or FPGA reprogramming is performed between executions. Due to the limited proton beam time and the number of different design versions, only the MxM benchmark application is tested.

7.2.4 Failure definition

For this investigation, the NOEL-V design events are defined as below:

- UNACE: The application finishes its execution as expected with the correct output. It is not a failure.
- SDC: The application finishes its execution as expected but reports an incorrect checksum from the resulting matrix. It is a failure.
- SEFI: The application does not finish its execution, and the design is not responsive, including application timeout or system crash. It is a failure.

The failures are divided into transient and permanent. Transient failures do not require any recovery, and the application execution in the sequence is UNACE. On the other hand, permanent failures persist over application executions and are only recovered

via processor reset, FPGA reprogramming, or board power cycle.

7.2.5 Emulation fault injection

The fault injection engine is based on the one used in chapter 6. The ICAP interface is used to access the CRAM frames and emulate bit-flips by XOR'ing the value of specific bits. The FI engine's VHDL code has been updated to target the Zynq UltraScale+ FPGA. The injection control is performed via Python scripts through a UART interface.

Faults are injected in the entire CRAM, except for the area of the injector engine. The target area is the same for all NOEL-V designs to perform a normalized fault injection. Bit-flips are randomly injected in order to mimic the radiation-induced upsets. Accumulative injections are used to evaluate the designs' reliability under multiple faults. One bit-flip is injected at a time, but the upsets are only cleaned when the system experiences a failure. A total of $10,000$ faults are injected per NOEL-V design. Since the SEM-IP also uses the ICAP to scrub the FPGA, scrubbing is disabled in all designs for the emulation fault injection experiments.

7.2.6 Proton testing

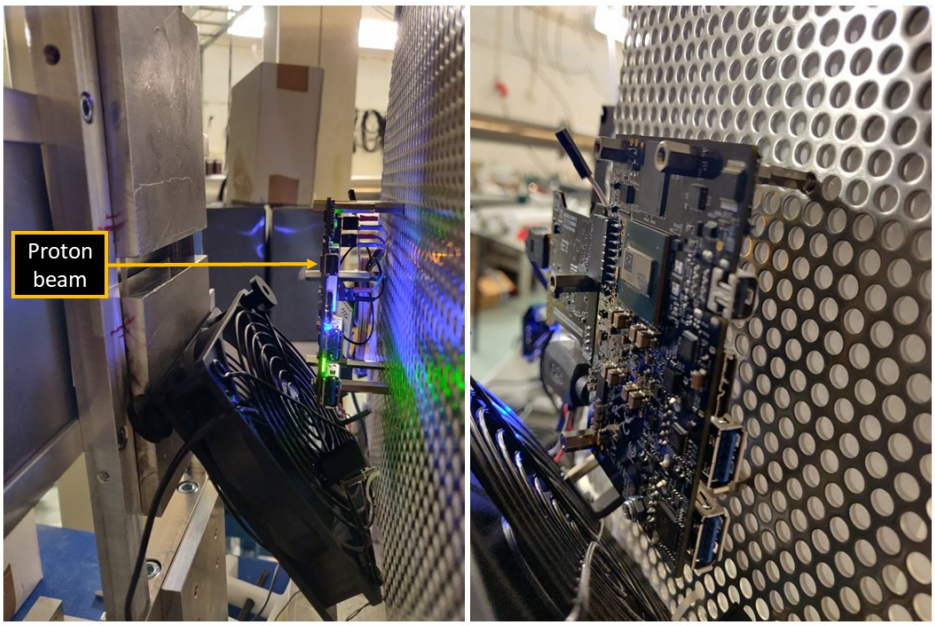
The Zynq UltraScale+ chip was irradiated using a proton beam at the Radiation Effects Facility (RADEF) at the University of Jyväskylä, Finland (UNIVERSITY OF JYVÄSKYLÄ, 2022). The beam time was granted by the RADiation facility Network for the EXploration of effects for indusTry and research (RADNEXT)[3] (CERN, 2023).

Figure 7.3 shows the test setup at the facility. The heat sink of the Ultra96-V2 board was removed to expose the Zynq UltraScale+ chip. A cooling fan was used to prevent the device from overheating. The tests were performed in air using a proton energy of 53 MeV, and the target fluence per run was 5×10^{10} p/cm^2. The 53 MeV proton beam is expected to reach more than 12 mm in Silicon, based on TRAD Omere tool estimation. This penetration is more than enough to go through the substrate thickness of the Zynq UltraScale+. To avoid the high spread of particles affecting other components on the board, a collimator with opening dimensions of 20×13 mm was set the closest

[3]This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No 101008126.

possible to the device, about a 7 cm distance.

Figure 7.3 – Zynq UltraScale+ proton testing setup.



Source: From the author.

7.3 Results

7.3.1 FPGA resource usage and performance

Table 7.2 presents the area and performance information per test design. The FPGA resource usage is detailed per processor core and the entire SoC. The number of essential bits in the design is also provided, including the percentage of FPGA usage.

The distributed TMR of the NOEL-V core requires about five times more LUTs and three times more FFs compared to the original NV_1KB design. Applying the TMR only to the IU and cache controller requires around 4.5 and 2.5 times more LUTs and FFs, respectively. Differently from expected, designs using the L1 cache DWC method do not use twice the number of BRAMs, but less than 1.6 times are required. Possibly because not all BRAM addresses are used in the original design, and those empty addresses might be filled when more memory is needed for the DWC designs. As expected, TMR designs require about three times more essential bits.

Table 7.2 also shows the MxM application execution time (in seconds) per design. All NOEL-V designs run at 50 MHz, which is the maximum achieved frequency for the

Table 7.2 – NOEL-V test designs information: resource usage per processor core and entire SoC; number of essential bits (percentage of FPGA usage); and application execution time (in seconds).

NOEL-V SoC design	Resource usage						Essential bits	App. exec
	Comp.	LUT	FF	Carry	DSP	BRAM	(% usage)	time (sec)
NV_1KBdis	core	11,690	6,833	150	16	13	5,466,435	33.965
	SoC	15,392	9,979	186	16	16	(17.70%)	
NV_TMRcore_1KBdis	core	59,687	20,365	450	48	13	17,506,959	33.947
	SoC	63,386	23,511	486	48	16	(56.70%)	
NV_1KB	core	11,690	6,833	150	16	13	5,466,435	9.924
	SoC	15,392	9,979	186	16	16	(17.70%)	
NV_TMRcore_1KB	core	59,687	20,365	450	48	13	17,506,959	9.928
	SoC	63,386	23,511	486	48	16	(56.70%)	
NV_cFT1KB	core	11,770	6,871	153	16	17	5,849,888	9.900
	SoC	16,511	10,333	207	16	20	(18.95%)	
NV_TMRcore_cFT1KB	core	59,550	20,401	453	48	17	17,774,878	9.950
	SoC	64,296	23,863	507	48	20	(57.57%)	
NV_TMRiucctrl_cFT1KB	core	52,732	17,499	347	16	17	16,296,146	9.905
	SoC	57,474	20,961	401	16	20	(52.78%)	
NV_Flush1KB	core	11,690	6,833	150	16	13	5,466,435	9.939
	SoC	15,392	9,979	186	16	16	(17.70%)	
NV_TMRcore_Flush1KB	core	59,687	20,365	450	48	13	17,506,959	9.968
	SoC	63,386	23,511	486	48	16	(56.70%)	
NV_8KB	core	13,820	7,256	155	16	23	6,188,702	2.964
	SoC	17,522	10,406	191	16	26	(20.04%)	
NV_TMRiucctrl_8KB	core	60,703	18,724	347	16	23	17,676,131	3.013
	SoC	64,400	21,874	383	16	26	(57.25%)	
NV_cFT8KB	core	14,482	7,323	161	16	37	6,673,561	3.031
	SoC	19,125	10,785	215	16	40	(21.61%)	
NV_TMRiucctrl_cFT8KB	core	61,841	18,807	353	16	37	17,676,131	3.978
	SoC	66,584	22,269	407	16	40	(57.25%)	
NV_Flush8KB	core	13,820	7,256	155	16	23	6,188,702	3.023
	SoC	17,522	10,406	191	16	26	(20.04%)	
NV_TMRiucctrl_Flush8KB	core	60,703	18,724	347	16	23	17,676,131	3.181
	SoC	64,400	21,874	383	16	26	(57.25%)	

triplicated designs. Disabling the L1 cache impacts 3.4 times the performance compared to the design with a 1 KB cache enabled. On the other hand, increasing the L1 cache size to 8 KB improves performance by more than three times.

7.3.2 Fault injection results

Table 7.3 describes the results of the fault injection experiments per NOEL-V design, detailing the number of failures, the error rate ($\#failures/\#inj.faults$) with

Table 7.3 – Fault injection result table describing the number of failures per design, the error rate with 95% confidence interval, and the mean faults to failure.

NOEL-V SoC design	Failures			Error rate[1]			Mean faults to failure[2]
	#SDCs	#SEFIs	#Total	Total	Lower confidence	Upper confidence	
NV_1KBdis	78	90	168	1.68×10^{-2}	1.44×10^{-2}	1.95×10^{-2}	59.52
NV_TMRcore_1KBdis	12	36	48	4.80×10^{-3}	3.54×10^{-3}	6.36×10^{-3}	208.33
NV_1KB	31	134	165	1.65×10^{-2}	1.41×10^{-2}	1.92×10^{-2}	60.61
NV_TMRcore_1KB	4	35	39	3.90×10^{-3}	2.77×10^{-3}	5.33×10^{-3}	256.41
NV_cFT1KB	39	153	192	1.92×10^{-2}	1.66×10^{-2}	2.21×10^{-2}	52.08
NV_TMRcore_cFT1KB	11	66	77	7.70×10^{-3}	6.08×10^{-3}	9.62×10^{-3}	129.87
NV_TMRiucctrl_cFT1KB	16	59	75	7.50×10^{-3}	5.90×10^{-3}	9.40×10^{-3}	133.33
NV_Flush1KB	48	152	200	2.00×10^{-2}	1.73×10^{-2}	2.30×10^{-2}	50.00
NV_TMRcore_Flush1KB	11	45	56	5.60×10^{-3}	4.23×10^{-3}	7.27×10^{-3}	178.57
NV_8KB	40	148	188	1.88×10^{-2}	1.62×10^{-2}	2.17×10^{-2}	53.19
NV_TMRiucctrl_8KB	12	53	65	6.50×10^{-3}	5.02×10^{-3}	8.28×10^{-3}	153.85
NV_cFT8KB	62	173	235	2.35×10^{-2}	2.06×10^{-2}	2.67×10^{-2}	42.55
NV_TMRiucctrl_cFT8KB	23	57	80	8.00×10^{-3}	6.34×10^{-3}	9.96×10^{-3}	125.00
NV_Flush8KB	49	146	195	1.95×10^{-2}	1.69×10^{-2}	2.24×10^{-2}	51.28
NV_TMRiucctrl_Flush8KB	17	53	70	7.00×10^{-3}	5.46×10^{-3}	8.84×10^{-3}	142.86

1. $FI\ error\ rate = total\ failures / total\ inj.\ faults$, as defined in section 5.2.2.
2. $Mean\ faults\ to\ failure = total\ inj.\ faults\ /\ total\ failures.$

a 95% of confidence interval, and the mean faults to failure. As expected, triplicating the entire processor core leads to a reduced number of errors. The NV_TMRcore_1KB presents the lowest error rate, which is more than four times better than NV_1KB. The design with triplication and L1 cache disabled (NV_TMRcore_1KBdis) also gives good results. Since one bit-flip is injected in the CRAM per application execution, the design's performance is not explored, and resource usage significantly impacts susceptibility. Designs with unprotected hardware are expected to have the worst error rates, such as the non-triplicated versions with 8 KB L1 cache.

Because injections do not affect BRAMs, the L1 cache protections are not exercised. Designs with periodic flush and cFT are underused, and the extra hardware increases the susceptibility. Those designs present more SEFIs than completely unprotected designs. A reason might be due to faults in the unmitigated cache controller that could have a SEFI during the regular flushes or lead to inconsistent data in the memories, causing a constant refresh. Either case would result in application timeout.

The mean faults to failure column in table 7.3 shows the average number of accumulated bit-flips to cause an error. The higher the values, the better. Since that computation is the reverse of the error rate metric, a design with a low error rate requires more

accumulated faults until experiencing a failure.

Figure 7.4 shows the reliability curves per design related to the number of accumulated injected bit-flips until a failure occurs, and figure 7.5 presents the Weibull fit of the reliability curves, describing the Weibull parameters per design in the format of (shape; width). Due to the characteristics of the CRAM fault injection, the configurations present similar reliability curves. However, the non-triplicated and triplicated designs are clearly spread. As expected, designs with TMRed hardware are more reliable, meaning they statistically provide more correct results. Non-triplicated designs use fewer resources but are highly susceptible to errors in case of bit-flips in the design area.

7.3.3 Proton testing results

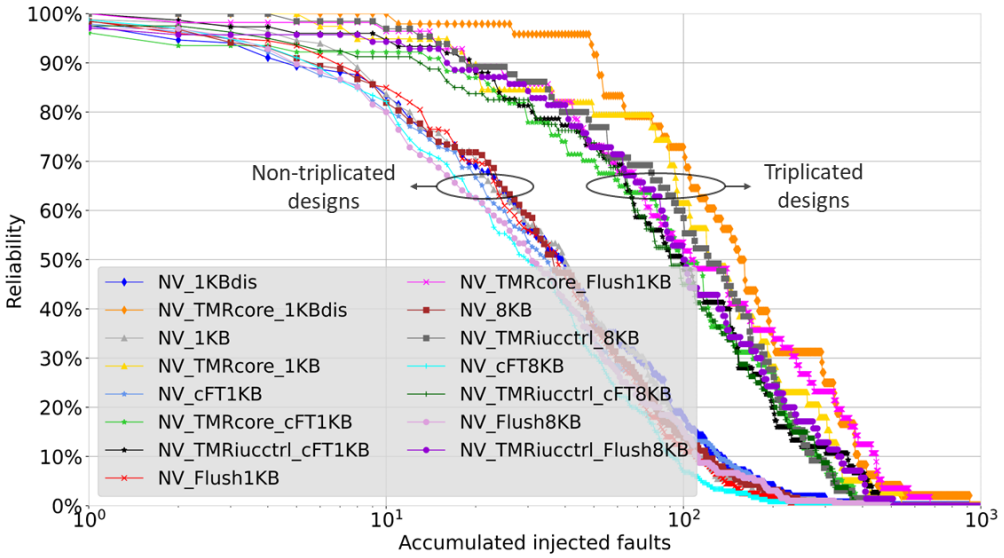
Two Ultra96-V2 boards were used in the experiments. The first board (BN1) was irradiated until accumulating a TID of about 80 Krad(Si). Then, the BN1 connection constantly failed, even with the beam disabled and after multiple power cycles. This behavior may be due to TID effects in ICs located around the DUT on the Ultra96-V2 board. As a note, the BN1 is again functional after annealing at room temperature. The second board (BN2) was irradiated until the end of the tests, with an accumulated TID of 40 Krad(Si).

The Zynq UltraScale+ FPGA fabric (ZU3EG chip) has $154,350$ system logic cells, about $54,063,047$ configuration bits. For a 53 MeV proton beam, the estimated fluence to upset in the FPGA CRAM is 1.3×10^8 p/cm^2, based on data from (KOGA et al., 2018). Table 7.4 describes the average flux and computed fluence per design and test board. The valid fluence means the calculated value after data analysis in which the unresponsive time of the design was removed.

Due to the limited beam time, a reduced number of configurations were tested. Table 7.5 presents the observed results per NOEL-V design. The total cross section considers the sum of SDCs and SEFIs and is presented with a 95% confidence interval, considering a fluence uncertainty of ±10%, as defined in the ESCC25100 (ESA, 2014). Transient and permanent failures were observed during the tests.

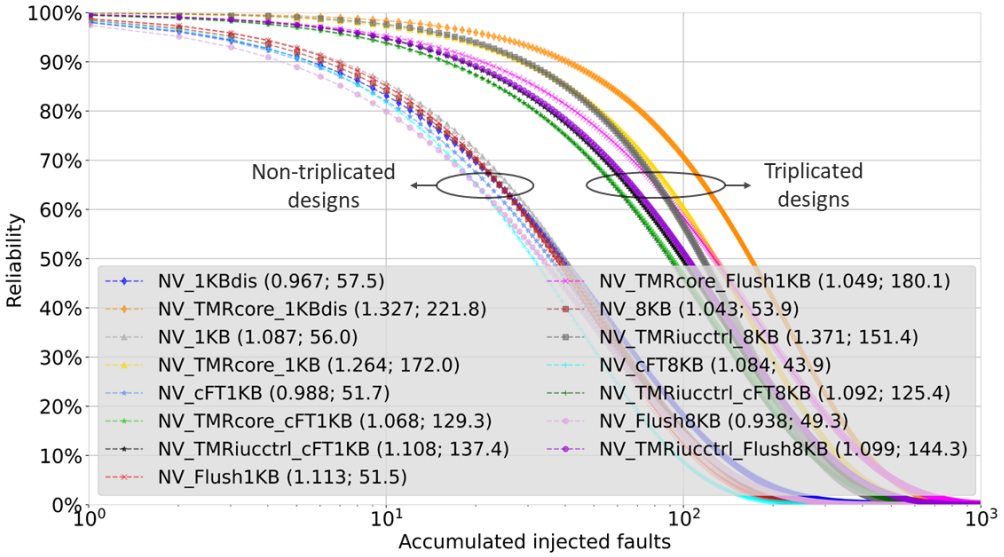
Transient failures were related to some SDCs, meaning that one MxM execution returned an incorrect result, but the subsequent executions were correct. Transient failures can occur due to soft errors in the user memories, such as the register file and L1 cache. When the memory is refreshed for the subsequent application execution, the data

Figure 7.4 – Reliability to failure as a function of accumulated injected faults.



Source: From the author.

Figure 7.5 – Weibull fit of the reliability to failure as a function of accumulated injected faults. The Weibull parameters are described per design (shape; width).



Source: From the author.

Table 7.4 – Proton testing table describing actual flux and fluence by design per test board.

NOEL-V SoC design	Test board	Proton beam	
		Flux (p/cm^2/s)	Fluence (p/cm^2)
NV_1KBdis	BN1	4.36×10^7	7.90×10^9
NV_TMRcore_1KBdis	BN1	4.22×10^7	2.14×10^{10}
NV_1KB	BN1	4.44×10^7	3.39×10^{10}
	BN2	4.46×10^7	3.76×10^{10}
NV_TMRcore_1KB	BN1	4.75×10^7	4.54×10^{10}
NV_cFT1KB	BN1	4.26×10^7	2.22×10^{10}
NV_TMRcore_cFT1KB	BN1	4.92×10^7	3.92×10^{10}
	BN2	4.44×10^7	4.26×10^{10}
NV_TMRiucctrl_cFT1KB	BN1	4.39×10^7	4.37×10^{10}
NV_8KB	BN1	4.83×10^7	4.35×10^{10}
	BN2	4.33×10^7	2.44×10^{10}
NV_TMRiucctrl_8KB	BN1	4.36×10^7	1.96×10^{10}
NV_TMRiucctrl_cFT8KB	BN1	4.68×10^7	1.73×10^{10}
	BN2	4.28×10^7	2.57×10^{10}
NV_TMRiucctrl_Flush8KB	BN2	4.14×10^7	4.58×10^{10}

Table 7.5 – 53 MeV proton testing result table describing the observed failures and total cross section with 95% confidence interval (±10% fluence uncertainty).

NOEL-V SoC design	Test board	Failures		Cross section		
		#SDCs	#SEFIs	σTotal (cm^2/device)	Lower confidence	Upper confidence
NV_1KBdis	BN1	0	2	2.53×10^{-10}	2.92×10^{-11}	9.15×10^{-10}
NV_TMRcore_1KBdis	BN1	0	1	4.67×10^{-11}	9.44×10^{-13}	2.61×10^{-10}
NV_1KB	BN1	1	3	1.18×10^{-10}	3.13×10^{-11}	3.03×10^{-10}
	BN2	2	2	1.06×10^{-10}	2.83×10^{-11}	2.73×10^{-10}
NV_TMRcore_1KB	BN1	1	2	6.61×10^{-11}	1.32×10^{-11}	1.93×10^{-10}
NV_cFT1KB	BN1	0	5	2.25×10^{-10}	7.14×10^{-11}	5.26×10^{-10}
NV_TMRcore_cFT1KB	BN1	1	3	1.02×10^{-10}	2.71×10^{-11}	2.61×10^{-10}
	BN2	0	2	4.69×10^{-11}	5.42×10^{-12}	1.70×10^{-10}
NV_TMRiucctrl_cFT1KB	BN1	0	1	2.29×10^{-11}	4.62×10^{-13}	1.27×10^{-10}
NV_8KB	BN1	4	5	2.07×10^{-10}	9.27×10^{-11}	3.94×10^{-10}
	BN2	0	6	2.46×10^{-10}	8.83×10^{-11}	5.36×10^{-10}
NV_TMRiucctrl_8KB	BN1	2	2	2.04×10^{-10}	5.42×10^{-11}	5.23×10^{-10}
NV_TMRiucctrl_cFT8KB	BN1	1	6	4.04×10^{-10}	1.59×10^{-10}	8.34×10^{-10}
	BN2	0	8	3.11×10^{-10}	1.32×10^{-10}	6.15×10^{-10}
NV_TMRiucctrl_Flush8KB	BN2	1	1	4.37×10^{-11}	5.04×10^{-12}	1.58×10^{-10}

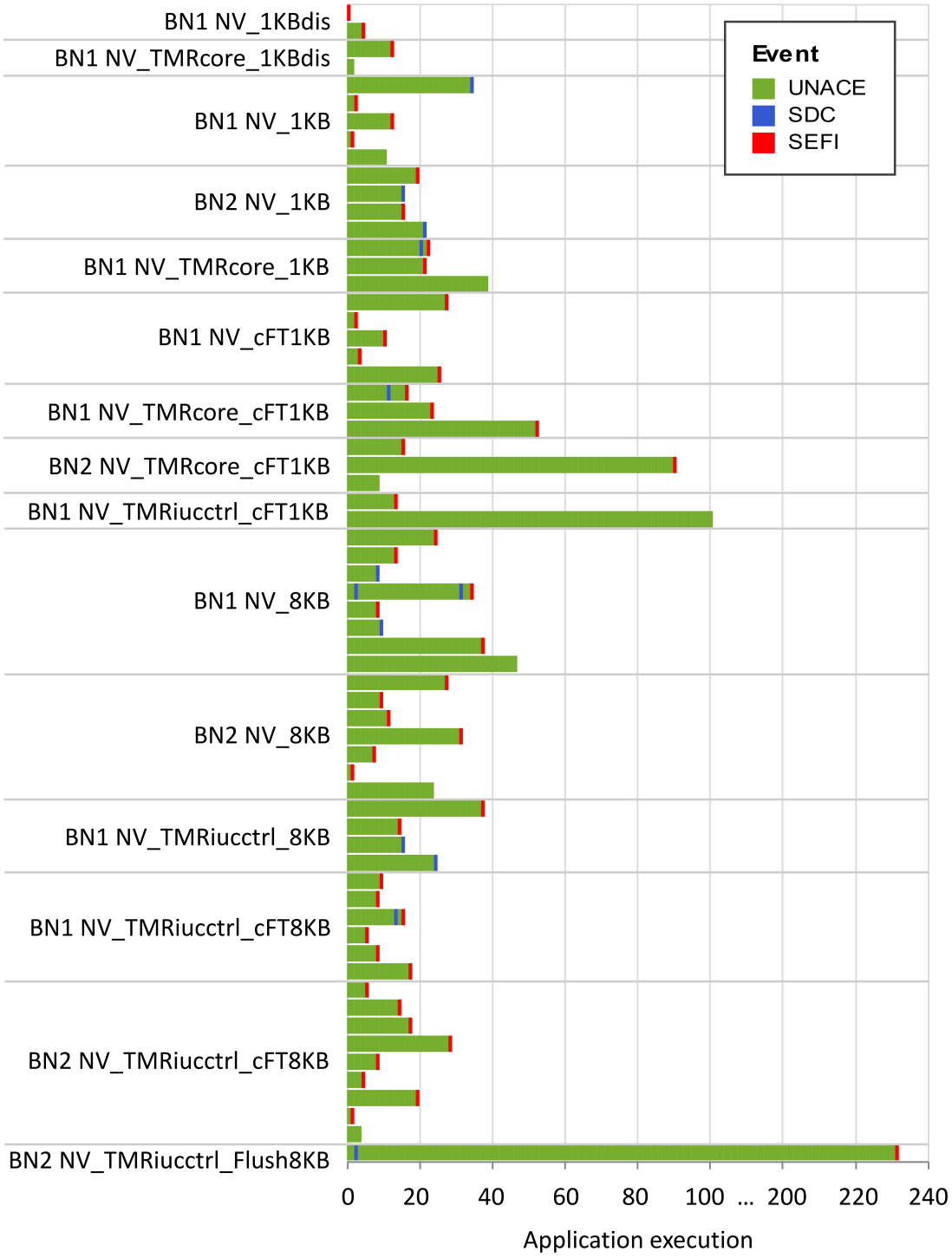
error is overwritten. All SEFI failures were permanent and repeatedly occurred in all the subsequent application executions until being recovered. Some SDCs were also detected in a burst of sequential MxM runs. Permanent failures are more related to architectural changes in the processor because of upsets in the FPGA CRAM. Additionally, the state of flip-flops is not recovered since the processor is not reset between executions. The permanent failure duration was excluded from the valid test fluence, and just the first event of the failure burst was computed as a valid event. The valid fluence and valid number of events were used to calculate the cross section results. Figure 7.6 shows the application executions events per design until a permanent failure or the end of the test run.

The FPGA CRAM soft errors can be persistent even using scrubbing mitigation. Although the detection and correction of CRAM upsets are enabled, the SEM-IP does not ensure the correction of all bit-flips. The SEM-IP scrubbing stops in the presence of uncorrectable errors, which may occur due to multiple upsets in the same CRAM frame (XILINX, 2022d). Therefore, the design is temporary without scrubbing until an FPGA reprogramming. Unfortunately, the occurrence of those events was not logged.

Some NOEL-V designs experienced application executions that were considerably shorter than expected. For instance, executions took around 2 seconds to compute a result despite an expected execution time of 10 seconds for the MxM operation. As a result, the application would either return an incorrect checksum or a repeated value from a previous execution. Such behavior can be due to control flow errors, like a branch error causing the program to jump to the wrong address. Multiples of these events happened sequentially and were only recovered with reprogramming, sometimes only with a power cycle. These events were considered SEFIs, and the multiple occurrences in sequence were counted as a single SEFI. Another unexpected observed behavior was the application executed for the expected period but did not report any result. These events commonly appeared in a permanent failure sequence. The NOEL-V designs with TMRiucctrl combined with L1 cache DWC (both 1 and 8 KB) or the periodic flush were the only designs that did not report any of those unexpected events. Interesting to note that these designs are with cache protection and have the distributed TMR applied to the IU and cache controller only.

Bursts of SDCs were observed in designs with an unprotected 8 KB L1 cache. Multiple sequential executions with wrong data results were reported in the NV_8KB and NV_TMRiucctrl_8KB designs. The burst of SDCs only stopped after a timeout or a system crash. Then the system was recovered with FPGA reprogramming.

Figure 7.6 – Events per application execution until a permanent failure or the end of the test run. Results are presented per design and test board (BN1/BN2).

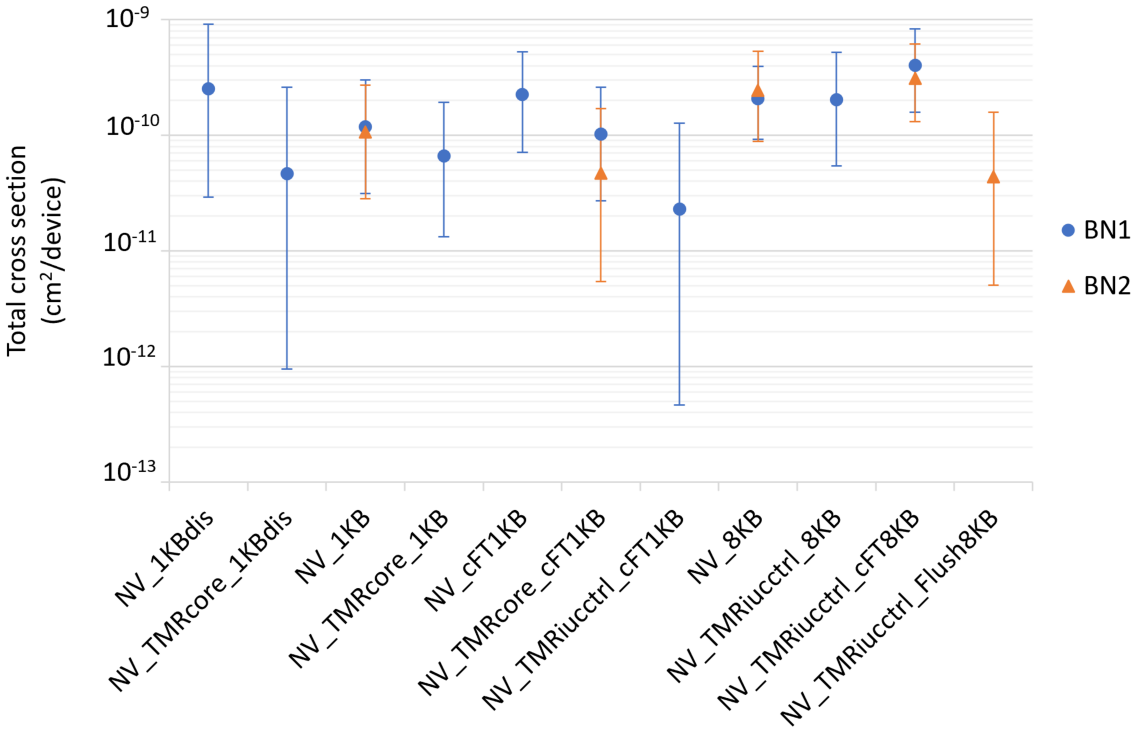


Source: From the author.

7.3.3.1 Failure cross section

Figure 7.7 presents the total failure cross section of the NOEL-V designs with a 95% confidence interval (±10% fluence uncertainty). These values are also described in table 7.5. Testing with higher proton fluence would be preferable to gather more statistics and improve the confidence interval of the results. Due to the limited beam time and the number of designs to be tested, the maximum valid fluence was 4.5×10^{10} p/cm^2 per design.

Figure 7.7 – Total cross section per device with 95% confidence interval for 53 MeV proton testing. Results are presented per design and test board (BN1/BN2).



Source: From the author.

The unprotected design with the L1 cache disabled has one of the highest cross sections due to the longer application execution time, which increases the susceptibility period during the execution. Additionally, although the L1 cache is disabled, the hardware (cache controller and BRAMs) is still implemented in the design since there is no support for removing the L1 from the processor. Upsets in the unused modules might still lead to failures.

The processor triplication has improved the system enhancement, with better results for 1 KB L1 cache (EN/DIS) designs. The TMRiucctrl also positively impacts the designs, reducing the cross section more than five times. Since the IU and the cache controller are the most critical and significant components in the NOEL-V core, the fact

that other elements are not triplicated has a low reliability impact. One should notice that results might differ in a more complex processor configuration. For instance, adding a FPU would considerably increase the core resource usage, and the susceptibility variation compared to an entire triplicated core is expected to increase. The cFT DWC also performs better with the 1 KB L1 cache and TMRiucctrl. The NV_TMRiucctrl_cFT1KB has 11 times lower cross section compared to NV_1KBdis.

Scrubbing is essential to avoid the accumulation of upsets in the FPGA CRAM and maintain the TMRed modules' integrity. However, upsets in critical bits may cause design failures. Cross-domain errors may still occur in triplicated designs and overcome the TMR masking capability, leading to failures. Additionally, the limitation of SEM-IP in treating uncorrectable bit-flips may leave the design vulnerable, as previously discussed.

As expected, increasing the L1 cache size without memory mitigation shows to be highly susceptible to soft errors due to the larger unprotected area. However, the periodic flush has efficiently mitigated memory upsets, avoiding error propagation. The NV_TMRiucctrl_Flush8KB design is about three and six times better than the NV_1KB and NV_1KBdis, respectively. A larger L1 cache with mitigation allows better performance with reduced cross section than an unprotected smaller cache.

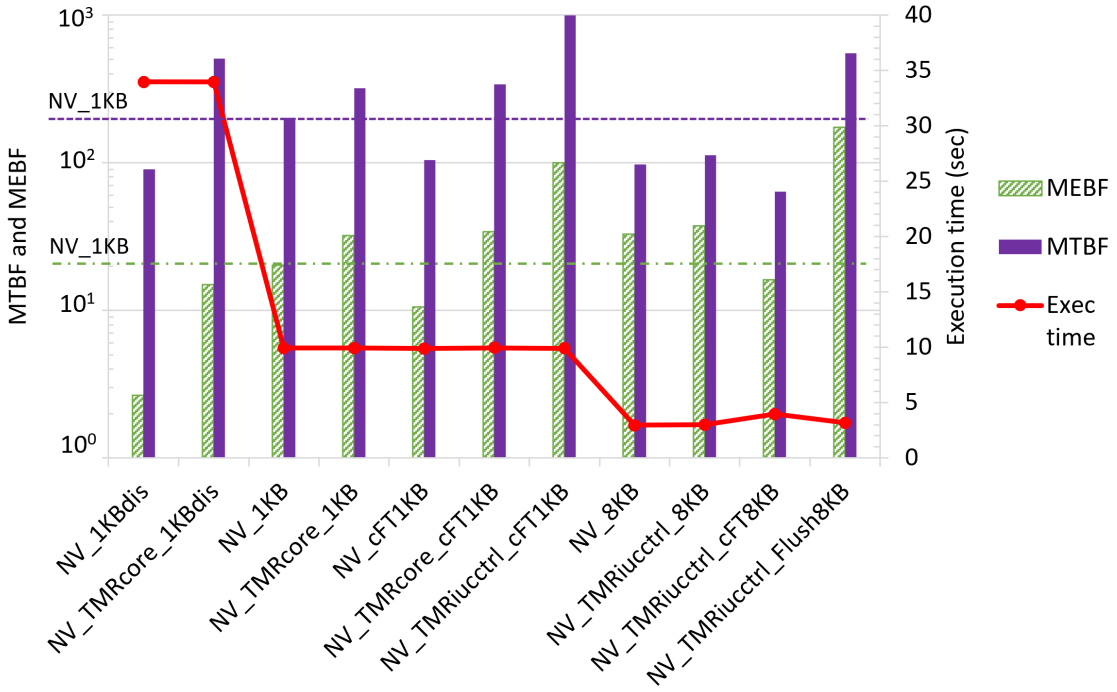
7.3.3.2 MTBF and MEBF results

Figure 7.8 shows the MTBF and MEBF results per NOEL-V design integrated from runs of both test boards (BN1 and BN2). MTBF and MEBF are presented in bars related to the left axis. The purple and green horizontal dashed lines highlight the NV_1KB design as a reference point for MTBF and MEBF, respectively. The best designs present the MTBF and MEBF higher than the NV_1KB reference. The higher the values, the longer the design availability until experiencing a failure and the higher the number of correct application executions. Figure 7.8 also shows the application execution time (in seconds), for a fault-free scenario, as the red line related to the right axis.

The NOEL-V design with the 1 KB L1 cache enabled performs better than with the cache disabled and, therefore, presents better MTBF and MEBF. The soft error susceptibility is reduced by triplicating the processor core even if the L1 cache is disabled. The NV_TMRcore_1KBdis design has higher MTBF than the NV_1KB, but with a lower MEBF as a penalty for the higher application execution time.

The highest MTBF is from the NV_TMRiucctrl_cFT1KB. Although most designs with 8 KB L1 cache present lower MTBF than the NV_1KB, the MEBFs are overall

Figure 7.8 – MTBF and MEBF are presented in bars related to the left axis, with the horizontal dashed lines highlighting the results for NV_1KB design as a reference. The red line shows the execution time, in seconds, related to the right axis. Results are presented per design integrated from both test boards.



Source: From the author.

better. Designs with a larger L1 cache tend to have higher MEBF because they present better performance (i.e., faster execution time), and the impact in the cross section is not so expressive. The best-case scenario for using the 8 KB L1 cache is reached with the periodic flush on the cache and triplicated IU and cache controller. Although the cache flush has some impact on the application execution time, the gain in error mitigation leads this design to present a high MTBF and the best MEBF.

7.3.4 Discussion

This thesis assesses the soft error vulnerability of a RISC-V NOEL-V soft processor implemented in the Xilinx Zynq UltraScale+ FPGA. Several factors have been evaluated, including the impact of L1 cache size on processor reliability, the effectiveness of periodic refreshing and DWC for cache protection, and the benefits of combining scrubbing with distributed TMR.

NOEL-V designs were evaluated under proton testing and emulation fault injection. Since the Zynq UltraScale+ FPGA is built on the 16 nm FinFET technology, it

requires a high proton fluence to upset. A higher test fluence per design configuration would be preferable for gathering a statistically higher number of failure events and reducing the data uncertainty. Nonetheless, valuable conclusions can be drawn from the results.

Combining distributed TMR and CRAM scrubbing can significantly improve processor enhancement. The TMR masking capabilities are maintained by the scrubbing ability to recover upsets, reducing the overall susceptibility to errors. The processor with the L1 cache disabled has a lower exposed area but at the expense of longer exposure time due to poor performance. The more time the design is exposed, the more prone to upsets. Better performance is achieved by enabling the L1 cache. However, mitigation techniques are mandatory for further improving reliability. The maximum improvement reached with mitigated NOEL-V designs was about eleven times.

As expected, designs with unprotected larger memories present higher susceptibility due to the increase of vulnerable resources. However, combining the distributed TMR to periodic flush of the cache leads to a reduced cross section and more application executions between failures, improving the MEBF. A design including a larger L1 cache with mitigation allows better performance with reduced error propagation than designs with unprotected smaller memories. Therefore, increasing the L1 cache size and using the correct combination of mitigation techniques can reduce the overall SEE susceptibility in RISC-V processors implemented in SRAM-based FPGAs.

8 EXPLORING THE COMMERCIAL FAULT TOLERANT RISC-V NOEL-VFT SOFT PROCESSOR COMBINED WITH EXTERNAL FPGA SUPERVISOR UNDER RADIATION EFFECTS

This thesis presents the SEE characterization under proton testing of the commercial fault tolerant RISC-V NOEL-VFT soft processor implemented in a 20 nm Xilinx Kintex UltraScale combined with the extra protection of an external FPGA supervisor and distributed TMR. The NOEL-VFT soft processor features built-in fault tolerance to protect the user memories against SEUs. It is equipped with SECDED in its internal memories, which can be adaptable per target technology or technology-agnostic, and BRAM scrubbing. The investigation aims to explore the SEE sensitiveness of the NOEL-VFT and to understand if even a fault tolerant soft processor requires additional enhancement when implemented in an SRAM-based FPGA.

The Ph.D. student developed the external FPGA supervisor while working at Frontgrade Gaisler (previously Cobham Gaisler) and, in collaboration with the Frontgrade Gaisler's team, performed the SEE characterization evaluation of the commercial NOEL-VFT soft processor.

8.1 RISC-V NOEL-VFT soft processor

The RISC-V NOEL-V processor, previously described in chapter 7, allows different configurations through the GRLIB IP Library (FRONTGRADE GAISLER, 2022b). Besides the processor features detailed in the previous chapter, the commercial version of the NOEL-V supports built-in soft error protection. The fault tolerance features of NOEL-VFT can also be complemented by GRLIB IP library designs with SEE mitigation techniques, including protection of on-chip peripherals and external memory (FRONTGRADE GAISLER, 2022b).

The fault tolerant NOEL-V (NOEL-VFT) can implement a custom RTL EDAC scheme to protect both the register file and the L1 cache. Each 32-bit word is covered by 8 check bits. The check bits can be used to correct 1-bit errors and detect 2-bit errors, similar to the conventional $32 + 7$ BCH protection. However, the top half of the check bits (4 bits) is the logical XOR of the other nibbles in the codeword. This extra check guarantees the detection of 3 and 4 adjacent bit errors, also protecting MBUs on some

technologies. The L2 cache is protected using a (39, 32, 7) BCH SECDED code.

The L1 and L2 caches include a hardware scrubber of the user memories that complements the error detection and correction schemes, preventing error accumulation without software intervention. The scrubber mechanism exercises the EDAC in all memory addresses and triggers the data correction in case of Correctable Errors (CEs). In case of Uncorrectable Errors (UEs), the cache line is invalidated, which leads to a refresh of the data.

As many modern FPGAs include BRAMs with ECC capabilities, the NOEL-VFT, like other GRLIB IPs, is also able to exercise such built-in ECC, maximizing the area efficiency of the design. In the case of the Xilinx UltraScale FPGAs (XILINX, 2022c), such error correction codes consist of 8 check bits for every 64 bits of data with SECDED capability.

In this chapter, we use the commercial fault tolerant NOEL-VFT processor. The selected NOEL-VFT is a single-issue 64-bit processor set for Controller (MC) configuration. This version was chosen due to its size to allow design triplication fitting into the Xilinx Kintex UltraScale XCKU060 FPGA. The SoC features an 8 KB L1 cache, a small FPU unit, 256 KB L2 cache, an Serial Peripheral Interface (SPI) memory controller, AMBA 2.0 AHB bus, debug unit, timers, UARTs, and error monitoring peripherals.

8.2 External FPGA supervisor

The external FPGA supervisor features programming and scrubbing capabilities on the target SRAM-based FPGA. The supervisor IP module, namely GRSCRUB, was developed by the Ph.D. student and is part of the GRLIB IP Library (FRONTGRADE GAISLER, 2022b). The advantage of an external FPGA supervisor is the reduced susceptibility to SEE since its engine is expected to be implemented in a radiation-tolerant technology device. The GRSCRUB IP can be implemented in a Rad-Hard FPGA or ASIC. The GRSCRUB will also be included in the next generation of Frontgrade Gaisler's microcontrollers, such as the GR716B (FRONTGRADE GAISLER, 2022a) and GR765 (FRONTGRADE GAISLER, 2023b).

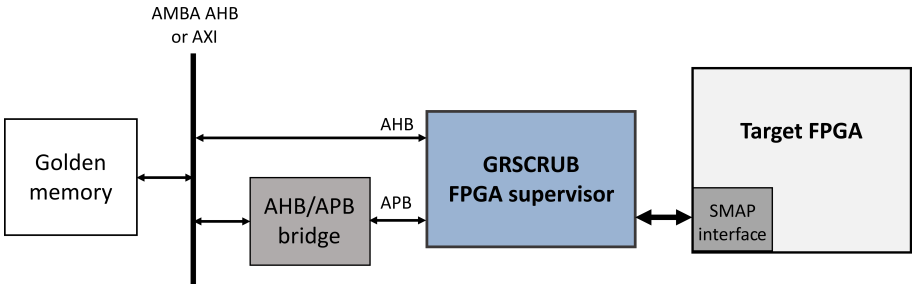
The GRSCRUB can detect and correct single and multiple errors affecting the FPGA CRAM, avoiding the accumulation of upsets. The GRSCRUB is compatible with the Xilinx Kintex UltraScale and Virtex-5 FPGA families. It accesses the FPGA configuration memory externally through the SelectMap (SMAP) interface. The SelectMap

performs better than JTAG due to its parallel data access with supported bus widths of 8-, 16-, or 32-bit data.

8.2.1 System integration

Figure 8.1 shows the block diagram of a GRSCRUB-based system integrated with the target FPGA. The target FPGA CRAM is accessed externally through the slave SelectMap configuration interface (control and data). The bus width (i.e., 8-, 16-, or 32-bit) can be configured in the GRSCRUB to meet the setup requirements. The SelectMap input clock should be provided externally by the subsystem in which the GRSCRUB is embedded. The GRSCRUB is designed in a multiple clock domain, which includes the internal system AMBA clock, and the SelectMap clock used for synchronization.

Figure 8.1 – Example of a system including GRSCRUB.



Source: From the author.

The GRSCRUB accesses through an AMBA AHB or AXI bus, a memory (Golden memory) that stores the golden reference of the configuration bitstream and mask data of the design implemented in the target FPGA. The golden bitstream is used both to configure the FPGA at start-up and to repair CRAM bit-flips. The mask data information is provided by the synthesis tool (e.g., Xilinx Vivado) and contains a description of all dynamic bits in the design. As described in section 4.1.1, the CRAM scrubbing operation does not protect the dynamic memory elements against soft errors. The GRSCRUB uses the mask data to identify all the dynamic bits in the CRAM frames during data verification.

8.2.2 Operational modes

The GRSCRUB FPGA supervisor features four operation modes:

- Programming mode: programs the configuration bitstream into the target FPGA.
- Scrubbing mode: executes a scrubbing operation. The IP can be configured to scrub the entire FPGA configuration memory or just selected CRAM frames.
- Mapping mode: identifies and maps the CRAM frame addressing of the target FPGA. The addressing map defines the positioning of the CRAM frames, required for any scrubbing operation. Only frames that refer to configuration blocks are mapped, i.e., the memory block frames are not considered. The frame addresses are saved in the Golden memory and are accessed by the IP in scrubbing mode during reading and writing operations.
- Golden CRC mode: computes the golden CRC codes for the current frame data of the target FPGA configuration memory. The CRC code can be selected as a data check in the readback scrubbing mode. A CRC code is computed to each frame of the configuration memory, and it is verified against the golden CRC copy.

8.2.2.1 Scrubbing operation

The GRSCRUB scrubbing operation mode supports both blind and readback methods. The blind scrubbing mode rewrites the configuration frames without any data verification. The blind scrubbing can be performed periodically, continually refreshing the configuration data.

In the readback scrubbing mode, the GRSCRUB reads frame-by-frame and check for inconsistencies. The error detection can be performed by Full Frame Check (FFC), which verifies all bits of the FPGA configuration frames, or using a 32-bit CRC algorithm, which validates the CRC code of each configuration frame. The error correction is performed by rewriting the entire faulty frame with its reference copy in the Golden memory. Differently from the blind scrubbing, the readback mode allows detecting errors and correcting the frame only if necessary. The readback can also be executed periodically.

8.2.2.2 Configuration interface integrity check

A novelty of the GRSCRUB is the integrity check of the FPGA configuration interface in the scrubbing mode. Soft errors affecting this interface may lead to catastrophic results during the scrubbing operation. For instance, an SEU in the FPGA Frame Address Register (FAR) may change its value to another valid address. During a blind scrubbing execution, this would wrongly overwrite all the subsequent scrubbing frames, compro-

mising the entire design. Lee, Swift and Wirthlin (2016) observed high-current events in Xilinx FPGAs due to SEUs affecting the configuration interface, which led to the blind scrubbing to write multiple frames in incorrect addresses.

The GRSCRUB was designed to decrease the probability of failures during the scrubbing operation due to a faulty interface. The GRSCRUB can be configured to verify the integrity of the configuration interface before each new scrubbing execution. The verification is performed by reading a specific frame and checking its address. If the returned address matches the expected one, the interface is considered stable, and the scrubbing cycle starts. Otherwise, an error is reported, and the scrubbing operation is suspended. With the integrity check, the responsiveness of the configuration interface is ensured before a new scrubbing. Additionally, setting up the configuration interface for each scrubbing frame could be safer than configuring all frames at once. For instance, writing one frame at a time during blind scrubbing avoids overwriting the entire memory in case of errors in the FAR register. Both blind and readback scrubbing can be configured to enable or disable such features.

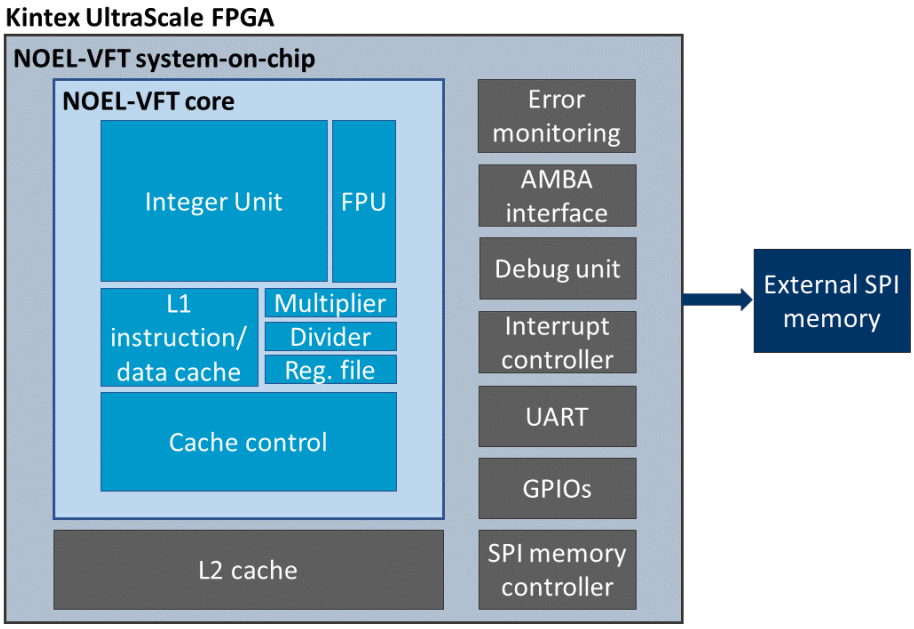
8.3 Investigation methodology

8.3.1 Platform setup

Figure 8.2 shows a block diagram of the NOEL-VFT SoC implemented in the Kintex UltraScale FPGA. The NOEL-VFT SoC characteristics are presented in section 8.1. The NOEL-VFT L1 cache makes use of the Xilinx UltraScale BRAMs built-in ECC capabilities, while the L2 cache uses the RTL EDAC. The boot memory in this setup is an external SPI memory. After boot, the L2 cache is locked and works as the processor's main memory. The hardware scrubbers of both L1 and L2 caches are enabled. In case of an uncorrectable error in the L1 cache, the cache line is invalidated, and the data is fetched from the L2 cache. In case of an uncorrectable error in the L2 cache, the behavior is different due to the setup restriction on not allowing a dynamic refresh of the L2 cache. If the scrubber detects the L2 UE, the error is reported, but no action is taken. In the case the L2 UE is detected due to a processor request, a trap is triggered, and the software is locked until a reset.

The DUT is the COTS Xilinx Kintex UltraScale XCKU060 FPGA, which is built on a High-K Metal Gate (HKMG) Taiwan Semiconductor Manufacturing Company

Figure 8.2 – NOEL-VFT SoC implemented into the Kintex UltraScale FPGA.



Source: From the author.

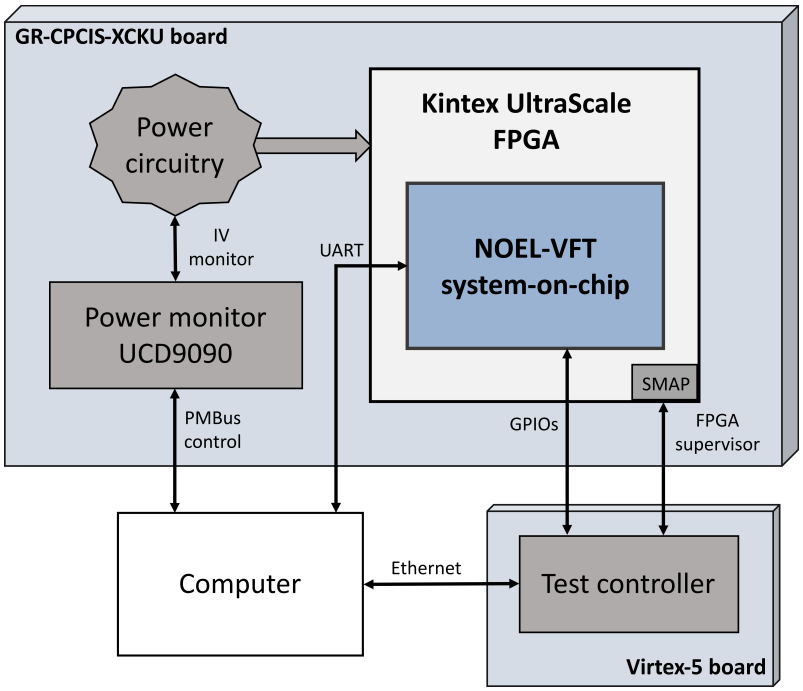
(TSMC) 20 nm planar CMOS technology (XILINX, 2022c). The test board is the GR-CPCIS-XCKU development platform (FRONTGRADE GAISLER, 2023a), which features a Kintex UltraScale XCKU060 FPGA (flip-chip BGA package). No sample preparation was required since the DUT was characterized under a proton beam.

Figure 8.3 describes the NOEL-VFT test system. The test board allows external access to the FPGA SelectMap interface. The Test Controller (TC), an additional board based on a Xilinx Virtex-5 FPGA, implements a GRSCRUB-based design that accesses the FPGA SelectMap interface. The GRSCRUB is used to program and scrub the Kintex UltraScale FPGA. A periodic readback, with both FFC and CRC methods enabled, continually checks the correctness of the entire CRAM. The XCKU060 has $147,596,064$ CRAM bits, and the SelectMap clock frequency is 25 MHz with 8-bit data bus width. The scrubbing cycle period is about 6.7 seconds. The performance is related to the GRSCRUB settings and SelectMap configuration (frequency and bus width). Moreover, the performance is affected by the size of the target FPGA.

The TC is also responsible for configuring and monitoring the NOEL-VFT SoC via GPIOs. The commercial GRMON3 debug monitor (FRONTGRADE GAISLER, 2022c) is used on the host computer to control the test system through Ethernet communication with the TC. The NOEL-VFT software reporting and memory error diagnosis are collected through Python scripts via the UART communication.

The test board embeds power monitoring circuitry of the DUT's power rails. The

Figure 8.3 – NOEL-VFT test system.



Source: From the author.

live monitoring of the DUT's current and voltage is performed through the Texas Instrument Fusion Digital Power™ software. Limits were established to power down the Kintex UltraScale FPGA in case of high current events.

8.3.2 Additional fault tolerance

As discussed in the previous chapters, the processor architecture is highly vulnerable to soft errors due to the CRAM SRAM cell characteristics. Failures can still occur even with complete protection on the user memories and with CRAM scrubbing. Since the NOEL-VFT features built-in protection for the user memories, the processor triplication is employed to investigate if TMR is also required in a commercial fault tolerant soft processor.

A distributed TMR synthesis strategy is applied to the entire NOEL-VFT SoC using Synopsys' Synplify Premier automated software tool (SYNOPSYS, 2015b). All internal submodules are triplicated and feedback voters are added to flip-flops with feedback logic for error correction. BRAMs are not triplicated since the L1 and L2 caches are already protected from SEUs by NOEL-VFT's EDAC and hardware scrubbing. Due to the complexity of the design and in order to facilitate the synthesis, the TMR is applied

individually to some elements in the NOEL-VFT subsystem, leading to single voters between those components.

8.3.3 Designs description overview

Table 8.1 describes the NOEL-VFT design versions synthesized into the Kintex UltraScale FPGA. As previously defined, the 8 KB L1 cache is enabled using the built-in Xilinx BRAM ECC, and the 256 KB L2 cache uses the RTL EDAC. Both caches are with hardware scrubbing enabled. In the NVFT_TMR, the distributed TMR is applied to the entire SoC, except for the BRAMs. Due to the timing constraints of the triplicated design, the NOEL-VFT processor runs at 30 MHz in both versions. Both designs are with GRSCRUB's CRAM scrubbing enabled. For comparison, the designs are also tested with CRAM scrubbing disabled but at low fluence since that is not the primary testing focus and due to the beam time restrictions.

Table 8.1 – NOEL-VFT test designs description.

NOEL-VFT	**L1 cache**			**L2 cache**			**Distributed**
SoC design	**Size**	**Status**	**Protection**	**Size**	**Status**	**Protection**	**TMR**
NVFT	8 KB	EN	BRAM ECC / scrubbing	256 KB	EN	RTL EDAC / scrubbing	NO
NVFT_TMR	8 KB	EN	BRAM ECC / scrubbing	256 KB	EN	RTL EDAC / scrubbing	Complete SoC, except BRAMs

8.3.4 Software benchmark

The test software is a bare-metal benchmark designed by Frontgrade Gaisler for processors' testing. The test software includes a set of test cases to evaluate different parts of the processor:

- IU (Integer Unit RAM test): to exercise the L1 cache and register file.
- Paranoia: to exercise the floating-point unit.
- Stanford: a set of benchmarks for general processor tests, such as sorting, FFT, puzzle, towers of Hanoi, permutation, and matrix multiplication.

The test software sequentially executes the test cases (i.e., IU, Paranoia, and Stan-

ford). Each test case runs in a loop for ten seconds. After the ten seconds window, the next test case starts. When the Stanford execution finishes, the IU test is re-executed, and the loop restarts. No reset is performed between test cases, and the software uninterruptedly runs until a SEFI is detected or the end of the test. The objective is to evaluate the uninterrupted software execution in the presence of correctable errors.

The test software reporting is performed via UART. A package is sent at every test case execution, informing the success of the task or data error detected. The software also reports the detected and corrected errors in memory elements, such as the L1 and L2 caches. The software execution is not affected when a memory error is corrected, except for reporting the error counters included in the design. For SEFI monitoring, the test software toggles a GPIO at every 100 ms, which works as a heartbeat to indicate that the software is working correctly. Moreover, a report package is also periodically sent via UART to sign the software availability.

8.3.5 Failure definition

For this investigation, the NOEL-VFT events are defined as below:

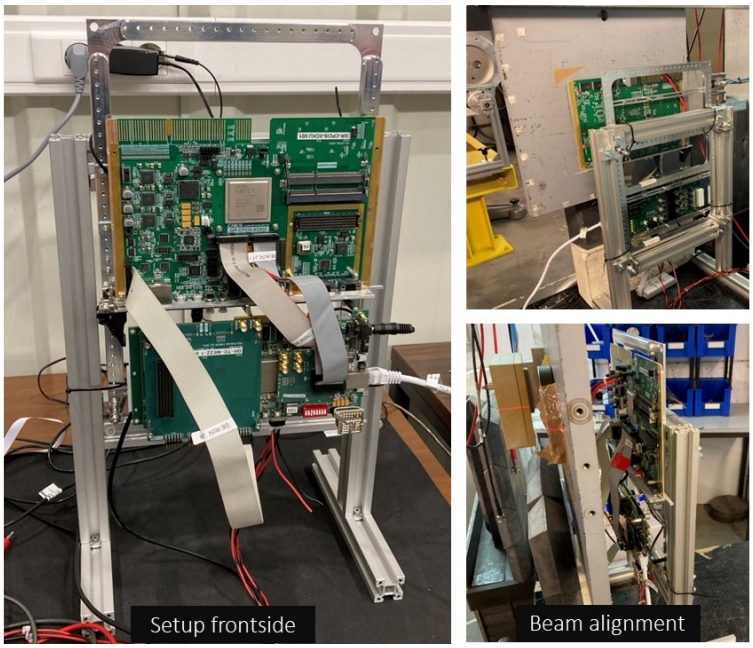
- SDC: A test case finishes its execution with a data error. It is a failure.
- SEFI: A test case does not finish its execution, and the processor is not responsive, including application timeout or system crash. SEFIs are also considered when a processor trap occurs due to L2 uncorrectable errors. It is a failure.
- L1 CE: L1 cache correctable errors are bit-flips detected and corrected by the fault tolerance features and do not interfere with the software execution. It is not a failure.
- L1 UE: L1 cache uncorrectable errors are bit-flips detected but cannot be corrected by the fault tolerance features due to multiple bit-error. This event may result in a processor trap and lead to a SEFI. It can lead to failure.
- L2 CE: L2 cache correctable errors are bit-flips detected and corrected by the fault tolerance features and do not interfere with the software execution. It is not a failure.
- L2 UE: L2 cache uncorrectable errors are bit-flips detected but cannot be corrected by the fault tolerance features due to multiple bit-error. This event may result in a processor trap and lead to a SEFI. It can lead to failure.

8.3.6 Proton testing

The proton testing was performed at the Light Ion Facility (LIF) at the Université Catholique de Louvain (UCL), Belgium (UNIVERSITÉ CATHOLIQUE DE LOUVAIN, 2023). The Kintex UltraScale XCKU060 FPGA was irradiated with the lid. It was assumed a 15 μm kovar for the lid and 100 μm to reach the sensitive silicon region. The irradiation was in the air using energies of 14.4, 25.8, 30.1, and 62 MeV at room temperature. Those proton energies are expected to reach many millimeters, ranging from 1 mm (14.4 MeV) to 16 mm (62 MeV), based on TRAD Omere tool estimation.

To avoid the high spread of particles affecting other components on the board, a collimator with opening dimensions of 40×40 mm was set the closest possible to the DUT. The test runs were divided into low and high fluence runs, with average flux ranging from 1×10^6 to 3×10^7 p/cm^2/s and fluence from 1×10^9 to 2×10^{10} p/cm^2. Figure 8.4 shows the test setup at LIF/UCL.

Figure 8.4 – Kintex UltraScale proton testing setup.



Source: From the author.

8.4 Results

8.4.1 FPGA resource usage

Table 8.2 presents the FPGA resource usage of the test designs for the entire NOEL-VFT SoC, and the number of essential bits, including the percentage of FPGA usage. The distributed TMR of the NOEL-VFT SoC requires about 4.8 more LUTs and 3.1 more FFs than the NOEL-VFT non-triplicated. More than three times of LUTs are needed due to the extra voters added after each triplicated submodule. In total, the NVFT_TMR uses 3.1 more essential bits.

Both NOEL-VFT designs run at 30 MHz, which is the maximum achieved frequency for the triplicated SoC. The performance information is omitted from table 8.2 due to the test software characteristic of endless execution, which does not lead to a fixed execution time.

Table 8.2 – NOEL-VFT test designs information: SoC resource usage and number of essential bits (percentage of FPGA usage).

NOEL-VFT SoC design	**Resource usage**					**Essential bits**
	LUT	**FF**	**Carry**	**DSP**	**BRAM**	**(% usage)**
NVFT	$44,461$	$24,271$	696	18	121.5	$15,706,771$ (10.60%)
NVFT_TMR	$211,643$	$74,842$	$2,744$	39	126	$48,826,903$ (32.95%)

8.4.2 Proton testing results

Table 8.3 describes the proton beam (energy and valid fluence), observed failures, and total cross section per NOEL-VFT SoC design. The valid fluence means the calculated value after data analysis in which the unresponsive time of the design was removed. The NVFT_TMR design was exercised over all tested proton energies. Due to beam time restrictions, the NVFT design was tested only under 30.1 and 62 MeV protons. In order to evaluate the GRSCRUB external scrubbing improvement in reliability, the NVFT and NVFT_TMR designs were also tested with scrubbing disabled (versions with *noScrub* labels) under 62 MeV protons.

The following sections detail the observed results in terms of susceptibility of the FPGA CRAM and BRAM bits, and the failure cross section.

Table 8.3 – Proton testing result table describing the proton energies and fluence, observed failures, and total cross section.

NOEL-VFT SoC design	Proton beam		Failures		Cross section
	Energy (MeV)	Fluence (p/cm^2)	#SDCs	#SEFIs	σTotal (cm^2/device)
NVFT	30.1	1.85×10^{10}	8	15	1.25×10^{-9}
	62	5.84×10^{10}	33	114	2.52×10^{-9}
NVFT_noScrub	62	9.99×10^{8}	26	1	2.70×10^{-8}
NVFT_TMR	14.4	5.00×10^{9}	0	0	0.00
	25.8	5.00×10^{9}	1	0	2.00×10^{-10}
	30.1	6.30×10^{10}	0	15	2.38×10^{-10}
	62	1.35×10^{11}	4	39	3.19×10^{-10}
NVFT_TMR_noScrub	62	1.53×10^{10}	3	13	1.05×10^{-9}

8.4.2.1 Configuration memory cross section

Figure 8.5 shows the Kintex UltraScale XCKU060 CRAM cross section per bit for upsets detected and corrected by the GRSCRUB. The CRAM cross section computation considers the integrated values of CRAM upsets results from both NOEL-VFT designs. All detected CRAM upsets were corrected. No bit-flips were detected by GRSCRUB for 14.4 MeV protons up to a fluence of 5×10^9 p/cm^2.

The resulting CRAM cross section from bit-flips reported by GRSCRUB matches the expected CRAM error rate of the Kintex UltraScale XCKU060 FPGA for proton testing (MAILLARD et al., 2019). Figure 8.5 also includes the Bendel distribution fitting for the GRSCRUB reporting results and the XCKU060 expected values.

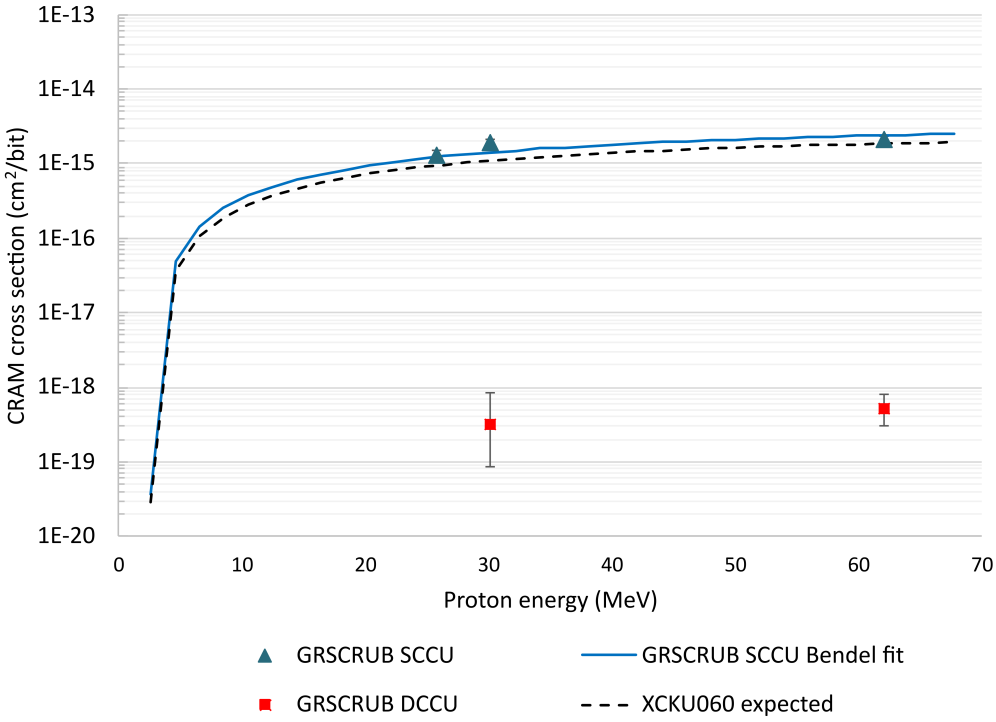
The CRAM upsets reported by the GRSCRUB are defined as Single-Cycle Corrected Upsets (SCCUs) and Double-Cycle Corrected Upsets (DCCUs). The SCCUs are upsets corrected in one scrubbing execution cycle, while DCCUs require two sequential scrubbing cycles to be repaired.

SRAM-based FPGAs present two types of CRAM errors (BERG et al., 2008):

- Single point error: an upset that affects a single function logic bit; and
- Burst of errors: an upset that directly affects the state of multiple bits.

For clarification, the burst of errors is not related to an MBU, but a single CRAM bit is flipped, and because of the updated state of the specific bit, other multiple bits have their states also updated. The burst of errors may occur due to SEUs in configuration bits responsible for controlling the status of other bits in the FPGA. When a control bit is

Figure 8.5 – CRAM cross section per bit for the Kintex UltraScale XCKU060: GRSCRUB SCCU and DCCU, Bendel fit of GRSCRUB SCCU, and Bendel fit of the XCKU060 reference data.



Source: From the author.

flipped, all other bits under its control are also affected and may only return to the correct state after correcting the original control bit. Therefore, in the worst-case scenario, two scrubbing cycles are required to recover all affected bits and restore the configuration memory to a consistent state. GRSCRUB could cope with both types of events and correct all detected errors.

8.4.2.2 User memory cross section

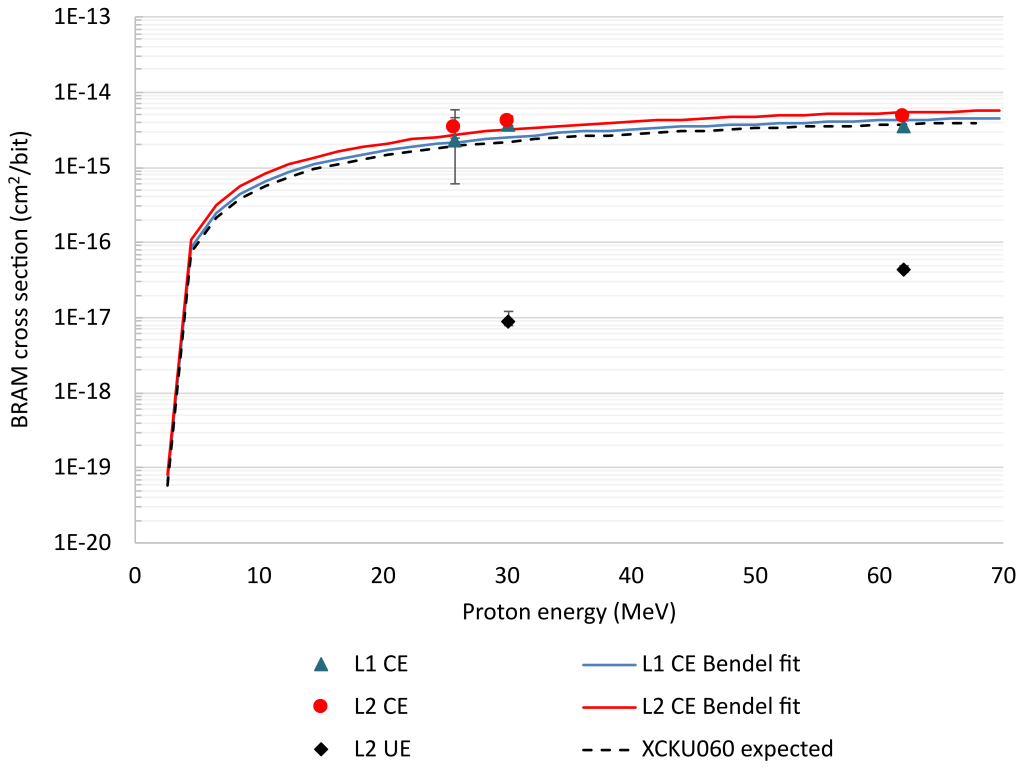
Figure 8.6 describes the BRAM cross section per bit for upsets detected and corrected by the EDAC features in the NOEL-VFT L1 and L2 caches. The BRAM cross section computation considers the integrated values of cache memories' upsets results from all NOEL-VFT designs. The observed BRAM upsets agree with the reference data for BRAM cross section of the Kintex UltraScale XCKU060 FPGA for proton testing (MAILLARD et al., 2019). Figure 8.6 also presents the Bendel distribution fitting for the reporting results of L1 and L2 correctable errors and the XCKU060 expected values.

The NOEL-VFT EDAC features successfully cope with single-bit upsets in the cache memories, avoiding the build-up of errors. No upsets were detected in the memories

for 14.4 MeV protons up to a fluence of 5×10^9 p/cm^2.

Figure 8.6 also shows the detected L2 cache uncorrectable errors. No UEs were observed in the L1 cache. An UE is due to multiple bit-flips in a cache line, which might be related to the high particle flux used during the accelerated irradiation testing. Therefore, the actual error rates in a space orbit are expected to be lower than the numbers reported here.

Figure 8.6 – BRAM cross section per bit for the Kintex UltraScale XCKU060: L1 cache CE, L2 cache CE and UE, Bendel fit of L1 and L2 caches CE, and Bendel fit of the XCKU060 reference data.



Source: From the author.

8.4.2.3 Failure cross section

Figure 8.7 presents the total failure cross section of the NOEL-VFT designs with a confidence level of 95%, considering a fluence uncertainty of ±10%, as per ESCC25100 (ESA, 2014). Large error bars are observed for some tests due to the lack of events. No failure events were observed for the NVFT_TMR design under the 14.4 MeV proton beam until a fluence of 5×10^9 p/cm^2. For this case, the cross section upper limit calculation presented in figure 8.7 considers one failure event for the 14.4 MeV tested fluence. For the 25.8 MeV protons, only one test case error was reported in the NVFT_TMR with a

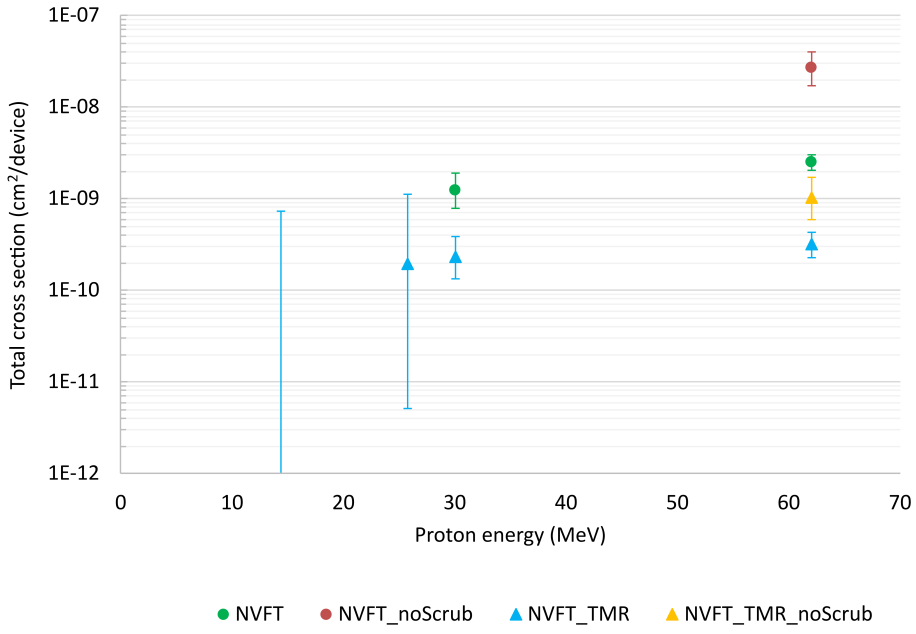
fluence of 5×10^9 p/cm^2, leading to a high error bar interval.

As expected, applying triplication to NOEL-VFT SoC shows to be highly effective along with scrubbing. Although the fault tolerance features of the NOEL-VFT protect the embedded user memories, the processor architecture in the CRAM is still highly susceptible. The NVFT_TMR leads to up to 7.9 times reduced cross section compared to NVFT. Failures can still occur due to critical bits in the triplicated design. Upsets in the design's single points of failure might overcome the TMR protection and lead to an error. All SEFIs were recoverable with a soft reset in designs with scrubbing enabled (NVFT and NVFT_TMR). In cases where scrubbing was disabled (i.e., versions with *noScrub*), reprogramming the FPGA was often required to recover the system.

The effectiveness of the GRSCRUB external scrubbing shows to improve the NVFT reliability by almost 11 times. When comparing the combined protection of scrubbing and triplication (i.e., NVFT_noScrub versus NVFT_TMR), the improvement boosts nearly 85 times.

No SEL or high current events were detected during the tests. The accumulated TID was about 57 krad(Si). No TID-induced effects were observed by the end of the campaign.

Figure 8.7 – Total cross section per device with 95% confidence interval for different proton energies.



Source: From the author.

8.4.2.4 MTTF and error rate in-orbit

The NOEL-VFT designs have been tested with different proton energies, resulting in distinct cross section points over the energy range. That data allows the computation of a fitting curve, which can be used to estimate the error rates in-orbit, such as LEO and GEO.

Assuming that the particle flux is much lower in a natural environment than in an accelerated irradiation test, efficient periodic scrubbing protection is expected to cope with all bit-flips in the CRAM, and upsets will not accumulate over time. Considering that the NOEL-VFT has built-in protection for the user memories, the SECDED features are expected to correct all errors in the used BRAMs. Based on those assumptions, failures would only be caused by SEUs happening within a CRAM scrubbing cycle. Therefore, estimating the MTTF and the error rate should only consider failures within the scrubbing cycle period.

The approach for estimating the MTTF for SEFI events follows the method proposed by Ostler et al. (2009). The authors adapted the MTTF calculation, previously presented in section 5.2.4, towards failures in a CRAM scrubbing duration. In summary, the re-designed MTTF equation is defined below (OSTLER et al., 2009).

$$MTTF = t_s \times (\Sigma_{n=0}^{\infty} P(n)P(F/n))^{-1} \tag{8.1}$$

In which:

- t_s is the scrubbing cycle duration (i.e., GRSCRUB period = 6.7 sec);
- the summation gives the probability of a failure in each scrubbing cycle;
- $P(n)$ is the probability of n upsets in a single scrubbing cycle; and
- $P(F/n)$ is the probability of a failure in a scrubbing cycle, assuming n upsets in that scrubbing cycle.

$P(n)$ follows a Poisson distribution relating the SEU rate and the scrubbing cycle, determined by

$$P(n) = e^{\mu t_s} \frac{(\mu t_s)^n}{n!} \tag{8.2}$$

where:

- μ is the Xilinx Kintex UltraScale KU060 CRAM upset rate (orbit dependent) calcu-

lated in OMERE 5.5 tool based on the reference static cross section data for heavy ion and proton testing (MAILLARD et al., 2019).

$P(F/n)$ (equation 8.3) is computed from the NOEL-VFT empirical proton data relating the GRSCRUB correctable error counters and the SEFIs in the design. A logistic curve, based on $P(F/n)$, is fitted to the data using the least square method (a and b are the parameters to be fitted).

$$P(F/n) = f(n) = \frac{1}{1 + (\frac{n}{a})^{-b}} \tag{8.3}$$

The error rate is estimated for LEO (800 km, $98°$ inclination) and GEO (36000 km) using CREME96 (Z=1-92) for heavy ions and AP8min for protons, with 1 g/cm^2 of Al shielding. The computed Kintex UltraScale KU060 CRAM upset rates (μ) are around 5 and 7.5 upsets/device/day for LEO and GEO, respectively. The SEFI rate and MTTF for the NOEL-VFT designs with scrubbing enabled are described in table 8.4.

Table 8.4 – Orbital SEFI rate and MTTF of NOEL-VFT test designs.

NOEL-VFT SoC design	**SEFI rate (events/device/day)**		**SEFI MTTF (years)**	
	LEO	**GEO**	**LEO**	**GEO**
NVFT	3.86×10^{-4}	5.78×10^{-4}	7.10	4.74
NVFT_TMR	6.00×10^{-5}	8.99×10^{-5}	45.70	30.47

9 COMPARATIVE ANALYSIS BETWEEN SOFT PROCESSORS

This chapter presents a comparison between this thesis and state-of-the-art works. Additionally, a comparison is performed between the soft processors regarding resource usage, emulation fault injection results, and proton testing cross sections.

9.1 Comparison to the state-of-the-art

Table 9.1 shows a comparative analysis between this thesis and the literature on state-of-the-art works on RISC-V soft processors. The contributions of this thesis over the related works have been discussed in section 4.2.

The state-of-the-art studies have tested different RISC-V processors, such as the open source Rocket, Taiga, VexRisc, and NOEL-V. The target SRAM-based FPGAs are primarily Xilinx well-known devices like Artix-7 and Zynq-7000 and more modern ones like the Kintex UltraScale FPGA. The investigations lie around redundancy (TMR and EDAC) and scrubbing. Most works perform single-bit emulation fault injection in the CRAM, and few perform irradiation tests using neutron beam. A work instead performed laser and X-ray testing and assessed the TID influence in the system. This thesis has proven the efficacy of combining fault tolerance methods to improve the reliability of the RISC-V Rocket, NOEL-V, and NOEL-VFT in the Zynq-7000, Zynq UltraScale+, and Kintex UltraScale, respectively. Tests were performed by emulating accumulated upsets in the CRAM, single-bit faults in BRAMs, and SEE testing using heavy ion and proton particles. The results are summarized in table 9.1.

Few works setups and results are comparable to this thesis. Aranda et al. (2020) saw that single-bit faults in the essential bits of the Rocket processor running combined benchmarks led to 99.7% and 90% of correctness for TMR and unhardened designs, respectively. Compared to the Rocket results presented in chapter 6, we observed maximum reliability for single faults of 88% for unhardened core (section 6.1.3.1) and 98.8% for triplicated core (section 6.1.3.3). Different benchmarks, FPGA devices, and the triplication implementation of the Rocket processor explain the differences in the single-bit injection results. This thesis goes further in the analysis and shows the system's reliability under an accumulation of faults and different combinations of techniques under heavy ion testing.

Table 9.1 – This thesis comparison against the state-of-the-art.

Work	RISC-V soft processor	DUT	Fault tolerance / investigation	Experiments	Results summary
Ramos et al. (2019)	Rocket	Artix-7	Partial TMR for ALU	Single-bit FI (CRAM)	Partial TMR differs in 2% correctness vs. full TMR
Wilson and Wirthlin (2019)	Taiga	Kintex UltraScale	TMR; CRAM scrub.	Single-bit FI (CRAM); neutron testing	33× lower cross section; 24× better MWTF; 27% lower freq.; 5.6× more area
Aranda et al. (2020)	Rocket	Kintex UltraScale	TMR; critical bits analysis	Single-bit FI (CRAM)	99.7% of reliability for TMRed vs. 90% for unmitigated (all bench.)
Dörflinger et al. (2020)	Rocket and BOOM	Virtex UltraScale+	ECC; BRAM scrub.	Performance and area tests	Area overhead: more than 5% logic and 41% BRAMs
Santos et al. (2020)	Customized RV32I	Zynq-7000	TMR for ALU/control unit; ECC for PC and reg. file	SEUs and SETs emulation	16× reduction in error propagation
Neri (2021)	Customized RI5CY	-	TMR; ECC	Simulation	No errors observed with FT features enabled
Wilson and Wirthlin (2021)	Taiga, VexRiscv, PicoRV32, and Kronos	Artix-7	TMR	Single-bit FI (CRAM)	More than 80× reliability improvement
Wilson et al. (2021)	VexRiscv	Artix-7	TMR; CRAM scrub.; Linux applications	Single-bit FI (CRAM); neutron testing	10× lower cross section; 4× more area
Shukla and Ray (2022)	Customized RV32IM	Artix-7 / 32 nm CMOS	DMR	Single-bit FI (CRAM)	The FT mode detected all injected upsets; re-execution for error correction
Li et al. (2022)	Customized RV32IM	Kintex-7	SECDED for the pipeline; rollback recovery	Simulation	Coverage of all simulated faults
Walsemann et al. (2023)	Customized RV32IMC	65 nm CMOS	TMR; SRAM scrub.	Laser and X-ray testing	1490% more leakage due to TID; all SRAM SEUs corrected
Wilson et al. (2023)	Taiga, VexRiscv, PicoRV32, and NOEL-V	Kintex UltraScale	TMR SoC (including DSPs and BRAMs); ext. CRAM scrub.	Single-bit FI (CRAM); neutron testing	75× lower cross section; 4.8× more area; 12.4% lower performance
This thesis	Rocket	Zynq-7000	TMR core: coarse grain and distributed; CRAM scrub.; watchdog; freq. variation; reliability	Accumulated FI (CRAM); heavy ion testing	98.8% max. FI reliability; 3.7× lower cross section (51× fewer timeouts with watchdog); 4.8× better MWBF; 6× more area
	Rocket	Zynq-7000	L1 cache topology ; periodic flush	Single-bit FI (BRAM)	Icache more vulnerable; same SDCs over configs.; few timeouts (small sizes); 67× lower ER w/ flush
	NOEL-V	Zynq UltraScale+	TMR core and IU/cache controller; CRAM scrub.; L1 cache DWC and periodic flush; cache size	Accumulated FI (CRAM); proton testing	High susceptibility for L1 cache dis. (exec. time); 11× lower cross section; 4.5× more area
	NOEL-VFT	Kintex UltraScale	TMR; built-in SECDED and BRAM scrub.; ext. CRAM scrub.; in-orbit ER	Proton testing	High coverage of CRAM and BRAM upsets; 85× lower cross section; 45 years to SEFI (LEO); 4.8× more area

Wilson and Wirthlin (2021) showed an improvement of more than 80 times for single-bit injection results in triplicated RISC-V soft processors (Taiga, VexRiscv, PicoRV32, and Kronos). The authors developed a fault injection platform to assess the CRAM sensitivity of the designs by implementing the processors in predefined partial block regions. In this thesis, we also used the partial block strategy to restrict the target area for fault injection in the Rocket soft processor.

Wilson et al. (2023) showed an improvement of about 75 times in the cross section of a triplicated NOEL-V soft processor under neutron testing. The experiments presented in chapter 7 resulted in up to 11 times cross section improvement for combined techniques on the NOEL-V soft processor. Distinct characteristics of the test setup may lead to those different results. In this thesis, only the NOEL-V core is triplicated due to the limited resources of the Zynq UltraScale+ ZU3EG chip, the internal SEM-IP is used for CRAM scrubbing in all designs (including non-triplicated), and the testing is performed under a proton beam. In (WILSON et al., 2023), the entire processor SoC is triplicated - including all BRAMs - combined with external CRAM scrubbing, and they perform neutron testing in the Kintex UltraScale FPGA. Additionally, the works evaluate different benchmarks running at different operation frequencies.

No works have been found that tested the commercial fault tolerant NOEL-VFT.

9.2 Comparative results between processors

9.2.1 Resource usage comparison

Table 9.2 describes the resource usage and tested frequency for Rocket (Zynq-7000), NOEL-V (Zynq UltraScale+), and NOEL-VFT (Kintex UltraScale) soft processors. Only a few NOEL-V configurations are presented for simplicity. The tested frequencies are the maximum allowed frequencies for the triplicated soft processors. The area information corresponds to the entire SoC implementation and is presented in chapters 6, 7, and 8, respectively. For instance, the high BRAM usage for the Rocket is due to the main memory being implemented in triplicated BRAMs, and the NOEL-VFT includes an L2 cache. Additionally, both processors feature an 8 KB L1 cache.

Table 9.2 – Resource usage and tested frequency for Rocket (Zynq-7000), NOEL-V (Zynq UltraScale+), and NOEL-VFT (Kintex UltraScale).

Soft processor	Design	Resource usage					LUT impact	Freq. (MHz)
		LUT	FF	Carry	DSP	BRAM		
Rocket	Unhard	17,329	12,164	544	9	68	1×	20
	CGTMR	39,232	21,495	369	5	68	2.3×	
	FDTMR	42,382	20,215	369	5	68	2.4×	
NOEL-V	NV_1KB	15,392	9,979	186	16	16	1×	50
	NV_TMRcore_1KB	63,386	23,511	486	48	16	4.1×	
	NV_8KB	17,522	10,406	191	16	26	1.1×	
	NV_TMRiucctrl_8KB	64,400	21,874	383	16	26	4.2×	
NOEL-VFT	NVFT	44,461	24,271	696	18	121.5	1×	30
	NV_TMR	211,643	74,842	2,744	39	126	4.8×	

The LUT usage impact is related to the unhardened version of the corresponding soft processor. As discussed in chapter 6, the TMR impact on the Rocket core area is more than 6 times, but when considering the total SoC area, the related impact is reduced to 2.4 times. Similarly, applying TMR to the NOEL-V core impacts more than 5 times the LUT usage (chapter 7) when considering only the core area, but it is reduced to 4.1 times considering the entire SoC. Since the TMR on the NOEL-VFT is employed for the whole SoC, the LUT impact corresponds to the actual TMR overhead.

9.2.2 FI comparison: Rocket vs. NOEL-V

Table 9.3 shows the error rate and mean faults to failure for the CRAM accumulation fault injection on the Rocket (Zynq-7000) and NOEL-V (Zynq UltraScale+) soft processors. Several factors lead to the high error rate on the Rocket designs. The FI target area is the Rocket Tile region, which narrows faults to the most sensitive parts in the processor. The entire CRAM is affected during the NOEL-V FI, which decreases the probability of an injected bit-flip leading to failures.

Another point is the TMR implementation. The coarse grain TMR is more vulnerable to cross-domain failures and has a low masking effect under multiple faults. Instead, the distributed TMR with feedback voters, the triplication method used on NOEL-V, has a high granularity to replicate elements and can restore the state of flip-flops. These characteristics allow the distributed TMR to maintain the error masking capability under multiple accumulated faults.

Table 9.3 – Fault injection results for Rocket (Zynq-7000) and NOEL-V (Zynq UltraScale+).

Soft processor	Design	Error rate	Mean faults to failure
Rocket	Unhard	1.60×10^{-1}	6.37
	CGTMR	1.80×10^{-1}	5.49
NOEL-V	NV_1KB	1.65×10^{-2}	60.61
	NV_TMRcore_1KB	3.90×10^{-3}	256.41
	NV_8KB	1.88×10^{-2}	53.19
	NV_TMRiucctrl_8KB	6.50×10^{-3}	153.85

It should also be considered that the Rocket soft processor seems to be more susceptible to failures than other processors. The work performed by Harward et al. (2015) describes a normalized soft error sensitivity for distinct soft processors implemented in a Virtex-5 FPGA. The Cortex-M0 presented the lowest sensitivity (i.e., 3.79%), and the PicoBlaze the highest (i.e., 6.11%). The sensitivity of the LEON3 soft processor is 4.81%. From another work, the estimated Rocket (lowRISC) sensitivity is 6.86% (RAMOS; MAESTRO; REVIRIEGO, 2017), which indicates the Rocket might be more prone to failures than other cores.

9.2.3 Cross section comparison: NOEL-V vs. NOEL-VFT

The open source NOEL-V and the commercial fault tolerant NOEL-VFT soft processors were tested on the Zynq UltraScale+ and Kintex UltraScale FPGAs, respectively. Both DUTs were irradiated using a proton beam. Because the DUTs are built on different technologies (i.e., 16 nm FinFET and 20 nm CMOS), and due to setup differences, a direct comparison of the design cross sections between processors shall not be made. Instead, we roughly estimate the designs dynamic cross section targeting one DUT based on the ratio of static cross sections of both DUTs under protons. This analysis is not performed with the Rocket results since its test was executed with heavy ions.

Some assumptions are taken for this analysis:

- The differences between the processors' setups are negligible for the cross section: SoC, frequency, test software, etc.
- The difference between 53 MeV, 60 MeV, and 62 MeV proton irradiation is negligible for the cross section.
- The dynamic failure cross section per device increases at the same rate as the

CRAM static cross section per bit.

- The ratio between the static cross section per bit of the Zynq UltraScale+ and the Kintex UltraScale is valid.

It is worth mentioning that those assumptions are made only to allow this estimated comparison and do not reflect reality. For a fair comparison, one should perform irradiation testing on the same conditions (beam, DUT, SoC, etc).

The first step is to find the ratio for the static cross section per bit between DUTs. The analysis is performed for 60 MeV protons to simplify the data collection from the literature. The CRAM static cross section per bit for 60 MeV protons is about 2.5×10^{-15} cm^2/bit for the Kintex UltraScale (MAILLARD et al., 2019) and around 1×10^{-16} cm^2/bit for the Zynq UltraScale+ (AZIMI et al., 2022). The CRAM static cross section ratio of Kintex UltraScale and Zynq UltraScale+ is 25.

Considering the Kintex UltraScale as the target device, the second step is to find the estimated dynamic cross section of the open source NOEL-V on the Kintex UltraScale FPGA. For that, we can multiply the cross sections reported in chapter 7 by 25 (i.e., the ratio computed above). To reduce the number of comparisons, only the following designs are selected: NV_1KB, NV_TMRcore_1KB, NV_8KB, NV_TMRiucctrl_8KB, and NV_TMRiucctrl_Flush8KB. These designs' characteristics are described in table 7.1, from chapter 7. The NOEL-VFT designs are NVFT and NVFT_TMR from chapter 8.

Table 9.4 shows the dynamic failure cross sections obtained from empiric data of irradiation tests for Zynq UltraScale+ (ZUS+) and Kintex UltraScale (KUS) FPGAs, and the estimated data for the open source NOEL-V on the Kintex UltraScale FPGA. The NVFT and NVFT_TMR results are repeated in the KUS estimated column for comparison convenience.

From the KUS estimated column, one can observe the most susceptible design is the unprotected NOEL-V core with 8 KB L1 cache (NV_8KB). In comparison, the NVFT, which features a NOEL-VFT with a protected 8 KB L1 cache, has a 2.4 times lower cross section. Adding fault tolerance to the open source NOEL-V shows to be an opportunity for those low-budget missions. The NV_TMRiucctrl_Flush8KB presents 2.3 times lower cross section than the NVFT. The best design, however, is the NVFT_TMR, featuring a NOEL-VFT with distributed TMR SoC. The NVFT_TMR is 19.3 better than NV_8KB.

Table 9.4 – Comparison between NOEL-V and NOEL-VFT dynamic failure cross section. Zynq UltraScale+ (ZUS+) 53 MeV are results from chapter 7, Kintex UltraScale (KUS) 62 MeV are results from chapter 8, and Kintex UltraScale (KUS) estimated are the computed cross section values.

Soft processor	**Design**	**Empiric data (cm^2/device)**		**KUS estimated (cm^2/device)**
		ZUS+ 53MeV	**KUS 62 MeV**	
NOEL-V	NV_1KB	1.18×10^{-10}	-	2.95×10^{-9}
	NV_TMRcore_1KB	6.61×10^{-11}	-	1.65×10^{-9}
	NV_8KB	2.46×10^{-10}	-	6.15×10^{-9}
	NV_TMRiucctrl_8KB	2.04×10^{-10}	-	5.10×10^{-9}
	NV_TMRiucctrl_Flush8KB	4.37×10^{-11}	-	1.09×10^{-9}
NOEL-VFT	NVFT	-	2.52×10^{-9}	2.52×10^{-9}
	NVFT_TMR	-	3.19×10^{-10}	3.19×10^{-10}

10 CONCLUSIONS

This thesis characterized the SEE susceptibility of RISC-V soft processors embedded in SRAM-based FPGAs and demonstrated how combining fault tolerance techniques can significantly reduce system vulnerability. The investigation addressed the problems of using soft processors in SEE-prone environments and the complexities and trade-offs behind mitigation methods.

The case studies are the open source RISC-V Rocket and NOEL-V soft processors and the commercial fault tolerant RISC-V NOEL-VFT. The processors are embedded in the Xilinx Zynq-7000 APSoC, Zynq UltraScale+ MPSoC, and Kintex UltraScale, respectively. Protection is applied at the design level targeting the FPGA configuration memory, processor core, and embedded memories. The susceptibility to soft errors is assessed under emulation fault injection and accelerated ground testing.

The following sections summarizes the contributions of this thesis and presents the future works.

10.1 Contributions

10.1.1 RISC-V Rocket soft processor investigation

This thesis assessed the soft error vulnerability of the in-order single-issue RISC-V Rocket soft processor implemented in a COTS SRAM-based FPGA. The Rocket processor is embedded into the Xilinx Zynq-7000 APSoC, built on 28 nm CMOS technology. Results from accumulated fault injection and heavy ion testing show that CRAM scrubbing and periodic reset are essential for SRAM-based FPGA designs. Scrubbing prevents the accumulation of bit-flips in the configuration memory. The reset restores the soft processor SoC to a known state.

The unhardened Rocket core is highly susceptible to single bit-flips. It achieved a maximum of 88% of reliability in the fault injection experiments. Therefore, additional user-level mitigation techniques are required to mask or correct faults. Combining scrubbing with the Rocket core triplication resulted in a reduction of three times the cross section. Adding a watchdog monitor improved the timeout cross section more than 51 times, showing the high efficacy of the monitoring.

An outcome of the heavy ion testing was to prove that protecting only the proces-

sor core may not be enough, depending on the target mission error rate. Protection needs to be extended to other parts of the SoC for higher fault coverage. Soft errors in the L1 cache, peripherals, buses, and other SoC components can also lead the soft processor to fail.

This thesis investigated the influence of the L1 cache topology on the Rocket soft processor susceptibility by emulating single-bit faults in user memories. Results showed an increase in vulnerability by increasing the cache size, which is related to the lower refresh of the memory (i.e., a lower cache miss). Small caches have a high miss rate, and the data is more often updated, which reduces the vulnerability window. The opposite is valid for larger caches. The 16 KB cache size is up to 12.5 times more susceptible to errors than 1 KB cache. Results also demonstrated a similar SDC rate over the L1 cache configurations, demonstrating the high probability of application output errors caused by upsets in the user memories. A high timeout rate is observed in larger memories, while SDCs are more likely to occur in smaller caches.

Most failures observed in the Rocket cache are due to upsets in the instruction cache. Few timeouts and no SDCs were observed from bit-flips in the data cache. Soft errors in the instruction cache may lead to data or control flow errors. If a corrupted instruction is validated, the processor can execute an incorrect instruction or generate a trap, leading to SDCs or SEFIs. Upsets in the instruction cache are more likely to lead to failures, and data memories are more susceptible than control memories.

This thesis also explored periodic flush on data memories of the Rocket's L1 instruction cache to reduce the soft error vulnerability. Increasing the refresh rate impacted the application performance in all topologies but also reduced the number of failures. The application error rate is reduced at a more frequent flush. Larger caches have a high drop in the error rate regarding the use of a periodic refresh. The impact on performance is also lower for larger caches. Therefore, by employing a periodic flush, one can take advantage of the faster application execution of a larger cache with a similar error rate of a small memory.

10.1.2 RISC-V NOEL-V soft processor investigation

The NOEL-V is a cutting-edge high-performance RISC-V processor targeting space computing. It features an advanced pipeline and is a highly configurable processor IP core. This thesis performed a sensitiveness analysis of the COTS open source NOEL-

V soft processor in the Xilinx Zynq UltraScale+ MPSoC, fabricated on 16 nm FinFET technology. The investigation included emulation fault injection and proton testing to confirm that strategies such as cache refreshing, scrubbing, TMR, and duplication with comparison are effective in mitigating soft errors.

Different NOEL-V design configurations were investigated. Results demonstrated that combining distributed TMR and CRAM scrubbing can significantly improve processor enhancement. The TMR masking capabilities are maintained by the scrubbing ability to recover upsets, reducing the overall susceptibility to errors. The distributed TMR with feedback voters combined with scrubbing for cleaning faults can sustain the masking capability under more accumulated bit-flips.

Disabling the L1 cache reduces the processor's exposed area but at the expense of longer application execution time and, therefore, longer exposure time. The more time the design is exposed, the more prone to upsets. Better performance is achieved by enabling the L1 cache. However, mitigation techniques are mandatory for further improving reliability.

The NOEL-V soft processor with an unprotected larger cache has higher susceptibility due to the increase of vulnerable resources. Combining the distributed TMR, CRAM scrubbing, and periodic cache flush leads to a reduced cross section and more application executions between failures, improving the MEBF. A design including a larger L1 cache with mitigation allows better performance with reduced error propagation than designs with unprotected smaller memories. Therefore, increasing the L1 cache size and using the correct combination of mitigation techniques can reduce the overall SEE susceptibility in RISC-V processors implemented in SRAM-based FPGAs.

10.1.3 RISC-V NOEL-VFT soft processor investigation

This thesis explored the use of a NOEL-VFT-based SoC implemented in a 20 nm Xilinx Kintex UltraScale combined with the extra protection of an external FPGA supervisor. The NOEL-VFT soft processor features built-in fault tolerance to protect the user memories against SEUs. It is equipped with SECDED in its internal memories, which can be adaptable per target technology or technology-agnostic, and BRAM scrubbing. Moreover, a distributed TMR with feedback voters is implemented to enhance the availability of the design. The SEE characterization was performed using proton testing.

Results demonstrated the NOEL-VFT's fault tolerance effectiveness in handling

soft errors and the robust hardness solution combining the NOELV-FT (with and without TMR) with the external scrubbing for SRAM-based FPGAs. The scrubbing corrected all detected CRAM upsets, and the error rate meets the Xilinx reference data. The NOEL-VFT EDAC features successfully cope with single-bit upsets in the cache memories, avoiding the build-up of errors. The observed BRAM upsets also agree with the reference data for the DUT.

The effectiveness of the external scrubbing shows to improve the NOEL-VFT reliability by almost 11 times. Combining the scrubbing and triplication protection can boost the improvement by nearly 85 times. The triplicated NOELV-FT has a mean time to SEFI of 45.7 years compared to 7.1 years for the non-triplicated processor in a typical LEO orbit, both with scrubbing enabled.

An outcome of this thesis is the development of the external FPGA supervisor - GRSCRUB IP, which can be implemented in a Rad-Hard FPGA or ASIC. The GRSCRUB IP will also be included in the next generation of Frontgrade Gaisler's microcontrollers, such as the GR716B and GR765.

10.1.4 Summary

RISC-V-based soft processors are already being used in many terrestrial applications, and there is an expectation of adopting RISC-V also in aerospace systems. An outcome of this thesis is demonstrating that RISC-V soft-processors embedded into a COTS SRAM-based FPGA present analogous SEE sensitivity as the state-of-the-art processors, and the system enhancement through fault tolerance techniques may significantly improve the reliability. Results also indicate that the processor core is one of many critical elements in the system that requires protection. Hardening caches and peripherals is essential to increase the system's overall reliability. Results are promising for using RISC-V soft processors in new generations of FPGAs and aerospace missions.

10.2 Future works

- **Improve the fault injection methodology.**

Emulation fault injection was performed in the soft processor systems targeting the FPGA configuration memory. However, due to the usage concurrency of the ICAP

interface, CRAM scrubbing could not be tested simultaneously in the designs. An improvement can be made in the fault injection system methodology to allow both scrubbing and injection. Although a parallel use of the ICAP is not possible, one can implement a controller to multiplex its access between the injector and the scrubber engine.

A possibility is to use an external scrubbing system using either JTAG or SelectMap interfaces. Both interfaces also use the ICAP, but an external module can coordinate its usage if the fault injection is performed via the same interface. For instance, injecting one fault at a time and releasing the scrubbing between injections. In this case, the scrubbing would be expected to fix all injected bit-flips.

Another improvement would be to simultaneously perform fault injection in the FPGA CRAM and BRAM in the system. This thesis performed CRAM and BRAM injections but in individual campaigns.

Additionally, injecting only in the essential bits of the designs would boost the performance of the fault injection, which currently can take days depending on the required number of faults.

- **Evaluation of other SoC components.**

The L2 cache topology could be investigated to understand if the observed effects on the L1 cache are also valid for the L2. This thesis has used the L2 cache in the NOEL-VFT, but no further investigation was performed to verify different configurations.

Soft errors in FPU can also be investigated. An FPU can occupy a large portion of the soft processor area and, therefore, be highly susceptible to soft errors.

- **Partial scrubbing investigation.**

In high-density FPGAs, a large portion of the device may be unused by the design. Depending on the scrubbing implementation, the placement of the design will also influence the CRAM frames' scrubbing rate, as in the case of SEM-IP. Restricting the scrubbing to the design area can reduce the time to detect and correct upsets and reduce the error build-up. An investigation of the error latency can be performed to understand if there is a correlation between the scrubbing rate and the number of failures.

- **Using the TMR voting output to trigger scrubbing.**

In power-restricted systems, frequently scrubbing the FPGA might not be an option. A possibility to save power and keep the fault coverage is to use an error flag from the TMR voters to inform when the TMR is masking faults and then trigger a scrubbing.

It would be interesting also to coordinate a reset of the system to restore possible unsafe states. A study could be performed to investigate the latency from faults happening until being fixed and the trade-off in the system.

- **Power analysis.**

One of the initial goals of this thesis was to perform an in-deep power analysis and understand the power penalties by employing different fault tolerance. Unfortunately, the low possibility of resources limited us to working only with the Xilinx Vivado power estimations, which are not accurate. An interesting study would be to assess the voltage and current consumption through circuitry on the test boards to measure the power variations considering different mitigation methods and, additionally, analyze if those variations increase with TID.

- **Testing other FPGA and ASIC devices.**

High-density FPGAs can be used to test the full triplication of the soft processors. This thesis showed the limitations of applying TMR on low resources devices. One outcome was the reduction of the protection due to more cross-domain failures. Using larger FPGAs would avoid those issues.

Additionally, the RISC-V soft processors could be evaluated in modern devices, such as the cutting-edge Xilinx Versal, with a 7 nm architecture, and other technologies, such as Lattice FPGAs. Another possibility is to compare the SoC susceptibility into flash-based and Rad-Hard devices.

ASIC implementations of the processors would open the opportunity for various comparisons. All the motivation discussions presented in this thesis could be checked, such as the differences in area, power consumption, cost, and susceptibility.

- **Error rate comparison with real aerospace missions.**

An interesting study would be implementing a RISC-V soft processor in an SRAM-based FPGA embedded onboard in a CubeSat and evaluating the observed error rate. This data could be used to assess future missions.

REFERENCES

ABATE, F.; STERPONE, L.; VIOLANTE, M. A New Mitigation Approach for Soft Errors in Embedded Processors. **IEEE Transactions on Nuclear Science**, v. 55, n. 4, p. 2063–2069, Aug 2008. ISSN 0018-9499.

AGUIAR et al. Experimental setup for Single Event Effects at the São Paulo 8UD Pelletron Accelerator. **Nuclear Instruments and Methods in Physics Research B**, v. 332, p. 397–400, aug. 2014.

ALTERA. **Stratix III Device Handbook, Volume 1**. [S.l.], 2011.

AMIN, A. F. M. et al. Embedded System Implementation on FPGA System With mu CLinux OS. **IOP Conference Series: Materials Science and Engineering**, v. 17, p. 012049, 02 2011.

ANDERSSON, J. et al. LEON Processor Devices for Space Missions: First 20 Years of LEON in Space. In: **2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)**. [S.l.: s.n.], 2017. p. 136–141.

ANGHEL, L. et al. **Multi-level Fault Effects Evaluation**. Dordrecht: Springer Netherlands, 2007. 69–88 p. [Accessed July-2022]. ISBN 978-1-4020-5646-8. Available from Internet: <https://doi.org/10.1007/978-1-4020-5646-8_4>.

ARANDA, L. A. et al. Analysis of the Critical Bits of a RISC-V Processor Implemented in an SRAM-Based FPGA for Space Applications. **Electronics**, v. 9, n. 1, 2020. ISSN 2079-9292.

ARM. **Cortex™-M3 Technical Reference Manual, Revision: r1p1**. [S.l.], 2005.

ARM. **Cortex-R5 and Cortex-R5F Technical Reference Manual. Revision: r1p2**. 2011.

ARNOLD, S. S.; NUZZACI, R.; GORDON-ROSS, A. Energy budgeting for CubeSats with an integrated FPGA. In: **2012 IEEE Aerospace Conference**. [S.l.: s.n.], 2012. p. 1–14.

ASADI, G.; TAHOORI, M. B. Soft Error Rate Estimation and Mitigation for SRAM-Based FPGAs. In: **Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-Programmable Gate Arrays**. New York, NY, USA: Association for Computing Machinery, 2005. (FPGA '05), p. 149–160. ISBN 1595930299. Available from Internet: <https://doi.org/10.1145/1046192.1046212>.

ASANOVIĆ et al. **The Rocket Chip Generator**. [S.l.], 2016. [Accessed November-2022]. Available from Internet: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html>.

ATITALLAH, A. B. et al. Real-Time video system design based on the NIOS II processor and μClinux. 01 2005. [Accessed December-2020]. Available from Internet: <https://www.design-reuse.com/articles/12547/real-time-video-system-design-based-on-the-nios-ii-processor-and-clinux.html>.

AVNET. **Ultra96-V2 Single Board Computer Hardware User's Guide, Version 1.3**. [S.l.], 2021.

AZAMBUJA, J. R.; KASTENSMIDT, F.; BECKER, J. **Hybrid Fault Tolerance Techniques to Detect Transient Faults in Embedded Processors**. [S.l.: s.n.], 2014. ISSN 1467-9280. ISBN 9780874216561.

AZIMI, S. et al. A comparative radiation analysis of reconfigurable memory technologies: FinFET versus bulk CMOS. **Microelectronics Reliability**, p. 114733, 2022. ISSN 0026-2714. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0026271422002578>.

BACHRACH, J. et al. Chisel: Constructing hardware in a Scala embedded language. In: **Design Automation Conference**. [S.l.: s.n.], 2012. p. 1216–1225.

BANSAL, A. et al. Evaluating the Memory Subsystem of a Configurable Heterogeneous MPSoC. In: **Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT)**. [S.l.: s.n.], 2018.

BAUMANN, R. C. Radiation-induced soft errors in advanced semiconductor technologies. **IEEE Transactions on Device and Materials Reliability**, v. 5, n. 3, p. 305–316, Sept 2005. ISSN 1530-4388.

BENDEL, W. L.; PETERSEN, E. L. Proton upsets in orbit. **IEEE Transactions on Nuclear Science**, v. 30, n. 6, p. 4481–4485, 1983.

BENEVENUTI, F.; KASTENSMIDT, F. L. Comparing Exhaustive and Random Fault Injection Methods for Configuration Memory on SRAM-based FPGAs. In: **2019 IEEE Latin American Test Symposium (LATS)**. [S.l.: s.n.], 2019. p. 1–6.

BENITES, L. A. C. et al. Reliability Calculation With Respect to Functional Failures Induced by Radiation in TMR Arm Cortex-M0 Soft-Core Embedded Into SRAM-Based FPGA. **IEEE Transactions on Nuclear Science**, v. 66, n. 7, p. 1433–1440, 2019.

BENITES, L. A. C.; KASTENSMIDT, F. L. Automated design flow for applying Triple Modular Redundancy (TMR) in complex digital circuits. In: **2018 IEEE 19th Latin-American Test Symposium (LATS)**. [S.l.: s.n.], 2018. p. 230–233.

BENSO, A. et al. A watchdog processor to detect data and control flow errors. In: **9th IEEE On-Line Testing Symposium, 2003. IOLTS 2003.** [S.l.: s.n.], 2003. p. 144–148.

BERG, M. et al. **Analyzing System on A Chip Single Event Upset Responses using Single Event Upset Data, Classical Reliability Models, and Space Environment Data**. 2017. [Accessed April-2023]. Available from Internet: <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20170009889.pdf>.

BERG, M. et al. Effectiveness of Internal Versus External SEU Scrubbing Mitigation Strategies in a Xilinx FPGA: Design, Test, and Analysis. **IEEE Transactions on Nuclear Science**, v. 55, n. 4, p. 2259–2266, 2008.

BERG, M. et al. An Analysis of Single Event Upset Dependencies on High Frequency and Architectural Implementations within Actel RTAX-S Family Field Programmable Gate Arrays. **IEEE Transactions on Nuclear Science**, v. 53, n. 6, p. 3569–3574, 2006.

BOWEN, N. S.; PRADHAM, D. K. Processor- and memory-based checkpoint and rollback recovery. **Computer**, v. 26, n. 2, p. 22–31, Feb 1993. ISSN 0018-9162.

BROSSER, F. et al. Assessing scrubbing techniques for Xilinx SRAM-based FPGAs in space applications. In: **2014 International Conference on Field-Programmable Technology (FPT)**. [S.l.: s.n.], 2014. p. 296–299.

BUCHNER, S. et al. Variable Depth Bragg Peak Method for Single Event Effects Testing. **IEEE Transactions on Nuclear Science**, v. 58, n. 6, p. 2976–2982, 2011.

BYU CONFIGURABLE COMPUTING LAB. **SpyDrNet TMR**. 2020. [Accessed October-2022]. Available from Internet: <https://byuccl.github.io/spydrnet-tmr/docs/stable/index.html>.

CALVEL, P. et al. An empirical model for predicting proton induced upset. **IEEE Transactions on Nuclear Science**, v. 43, n. 6, p. 2827–2832, 1996.

CARLO, S. D. et al. A fault injection methodology and infrastructure for fast single event upsets emulation on Xilinx SRAM-based FPGAs. In: **2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)**. [S.l.: s.n.], 2014. p. 159–164.

CERN. **RADiation facility Network for the EXploration of effects for indusTry and research**. 2023. [Accessed April-2023]. Available from Internet: <https://radnext.web.cern.ch/>.

CHIELLE, E. et al. S-SETA: Selective Software-Only Error-Detection Technique Using Assertions. **IEEE Transactions on Nuclear Science**, v. 62, n. 6, p. 3088–3095, Dec 2015. ISSN 0018-9499.

CHO, H. Impact of Microarchitectural Differences of RISC-V Processor Cores on Soft Error Effects. **IEEE Access**, v. 6, p. 41302–41313, 2018.

COX, C. **Success Story: How RISC-V Is Enabling the Internet of Space**. Embedded Computing Design, 2022. [Accessed April-2023]. Available from Internet: <https://embeddedcomputing.com/technology/open-source/risc-v-open-source-ip/success-story-how-risc-v-is-enabling-the-internet-of-space>.

CPCL. **The CubeSat standard**. Cal Poly CubeSat Laboratory (CPCL), California Polytechnic State University, 2018. [Accessed October-2021]. Available from Internet: <https://www.cubesat.org/cubesatinfo>.

DI MASCIO et al. Leveraging the Openness and Modularity of RISC-V in Space. **Journal of Aerospace Information Systems**, v. 16, n. 11, p. 454–472, 2019. Doi:10.2514/1.I010735.

DIMOND, R. G.; MENCER, O.; LUK, W. Combining Instruction Coding and Scheduling to Optimize Energy in System-on-FPGA. **2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines**, p. 175–184, 2006.

DÖRFLINGER, A. et al. "ECC Memory for Fault Tolerant RISC-V Processors". In: BRINKMANN, A. et al. (Ed.). **Architecture of Computing Systems – ARCS 2020**. Cham: Springer International Publishing, 2020. p. 44–55. ISBN 978-3-030-52794-5.

DOUCIN, B. et al. Model of single event upsets induced by space protons in electronic devices. In: **Proceedings of the Third European Conference on Radiation and its Effects on Components and Systems**. [S.l.: s.n.], 1995. p. 402–408.

ECKERT, M. et al. Comparison and evaluation of cache parameters for softcores on FPGAs. In: **2017 International Conference on FPGA Reconfiguration for General-Purpose Computing (FPGA4GPC)**. [S.l.: s.n.], 2017. p. 19–24.

ENTRENA, L. et al. Soft error sensitivity evaluation of microprocessors by multilevel emulation-based fault injection. **IEEE Transactions on Computers**, v. 61, n. 3, p. 313–322, 2012.

ESA. **Single Event Effects Test Method and Guidelines - ESCC Basic Specification No. 25100**. 2014.

ESA. **SPENVIS - The Space Environment Information System**. 2018. [Accessed July-2022]. Available from Internet: <https://www.spenvis.oma.be/intro.php>.

ESA. **Trisat-R Nanosatellite**. 2022. [Accessed April-2023]. Available from Internet: <https://www.eoportal.org/satellite-missions/trisat-r#trisat-r>.

ESPOSITO et al. COTS-Based High-Performance Computing for Space Applications. **IEEE Transactions on Nuclear Science**, v. 62, n. 6, p. 2687–2694, Dec 2015.

Ferlini, F. et al. Non-intrusive fault tolerance in soft processors through circuit duplication. In: **2012 13th Latin American Test Workshop (LATW)**. [S.l.: s.n.], 2012. p. 1–6.

FRONTGRADE GAISLER. **Comparison Between GR716A and GR716B, Application Note, Doc. GR716B-COMP-1, Issue 1.0**. [S.l.], 2022. [Accessed November-2022]. Available from Internet: <https://www.gaisler.com/doc/gr716/GR716B-COMP-1-1-0-Comparison_between_GR716A_and_GR716B.pdf>.

FRONTGRADE GAISLER. **GRLIB IP Core User's Manual, Version 2022.2**. [S.l.], 2022. [Accessed October-2022]. Available from Internet: <https://www.gaisler.com/products/grlib/grip.pdf>.

FRONTGRADE GAISLER. **GRMON3 User's Manual, ver. 3.2.15, Feb. 2022**. 2022.

FRONTGRADE GAISLER. **GR-CPCIS-XCKU Development Board**. 2023. [Accessed April-2023]. Available from Internet: <https://gaisler.com/index.php/products/boards/gr-cpcis-xcku>.

FRONTGRADE GAISLER. **GR765 Octa-Core LEON5 SPARC V8 Processor**. 2023. [Accessed April-2023]. Available from Internet: <https://gaisler.com/index.php/products/components/gr765>.

FUCHS, C. M. et al. A Fault-Tolerant MPSoC For CubeSats. In: **2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)**. [S.l.: s.n.], 2019. p. 1–6. Doi:10.1109/DFT.2019.8875417.

GARDENYES, R. B. **Trends and patterns in ASIC and FPGA use in space missions and impact in technology roadmaps of the European Space Agency**. Dissertation (Master) — T. U. Delft and ESA, 2012.

GOLOUBEVA, O. et al. **Software-implemented hardware fault tolerance**. [S.l.]: Springer Science & Business Media, 2006.

GOMEZ-CORNEJO, J. et al. Fast context reloading lockstep approach for SEUs mitigation in a FPGA soft core processor. In: **IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society**. [S.l.: s.n.], 2013. p. 2261–2266. ISSN 1553-572X.

GROVER, N.; K.SONI, M. Reduction of Power Consumption in FPGAs - An Overview. **International Journal of Information Engineering and Electronic Business**, v. 4, p. 50–69, 10 2012.

GUPTA, S. et al. SHAKTI-F: A Fault Tolerant Microprocessor Architecture. In: **2015 IEEE 24th Asian Test Symposium (ATS)**. [S.l.: s.n.], 2015. p. 163–168.

GUTHAUS, M. R. et al. MiBench: A free, commercially representative embedded benchmark suite. In: **Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)**. [S.l.: s.n.], 2001. p. 3–14.

HARDGROVE, C. et al. The Lunar Polar Hydrogen Mapper CubeSat Mission. **IEEE Aerospace and Electronic Systems Magazine**, v. 35, n. 3, p. 54–69, 2020. Doi:10.1109/MAES.2019.2950747.

HARWARD, N. A. et al. Estimating Soft Processor Soft Error Sensitivity through Fault Injection. In: **2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines**. [S.l.: s.n.], 2015. p. 143–150.

HAUCK, S.; DEHON, A. **Reconfigurable Computing**. [S.l.]: Elsevier, 2008. ISBN 978-0-12-370522-8.

HEIDA, W. F. **Towards a fault tolerant RISC-V softcore**. Dissertation (Master) — Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Netherlands, 2016.

HEINER, J.; COLLINS, N.; WIRTHLIN, M. Fault Tolerant ICAP Controller for High-Reliable Internal Scrubbing. In: **2008 IEEE Aerospace Conference**. [S.l.: s.n.], 2008. p. 1–10.

HUFFMAN, W. C.; PLESS, V. **Fundamentals of Error-Correcting Codes**. [S.l.]: Cambridge University Press, 2003.

ISMAIL, M. et al. Eclipse intervals for satellites in circular orbit under the effects of Earth's oblateness and solar radiation pressure. **NRIAG Journal of Astronomy and Geophysics**, Taylor Francis, v. 4, n. 1, p. 117–122, 2015. Available from Internet: <https://doi.org/10.1016/j.nrjag.2015.06.001>.

JAYAKRISHNAN, V.; PARIKH, C. Embedded Processors on FPGA: Soft vs Hard. In: **2019 ASEE North Central Section Conference**. [S.l.]: American Society for Engineering Education, 2019. p. 1–8.

JEDEC. **JESD57 - Test procedures for the measurement of single-event effects in semiconductor devices from heavy ion irradiation**. Electronic Industries Assoc., Engineering Dept., Arlington, VA: [s.n.], 1996.

KASAP, S. et al. Survey of Soft Error Mitigation Techniques Applied to LEON3 Soft Processors on SRAM-Based FPGAs. **IEEE Access**, v. 8, p. 28646–28658, 2020.

KASTENSMIDT, F. L. et al. Designing and Testing Fault-Tolerant Techniques for SRAM-Based FPGAs. In: **Proceedings of the 1st Conference on Computing Frontiers**. New York, NY, USA: Association for Computing Machinery, 2004. (CF '04), p. 419–432. ISBN 1581137419. Available from Internet: <https://doi.org/10.1145/977091.977150>.

KASTENSMIDT, F. L. et al. On the optimal design of triple modular redundancy logic for SRAM-based FPGAs. In: **Design, Automation and Test in Europe**. [S.l.: s.n.], 2005. p. 1290–1295 Vol. 2. ISSN 1530-1591.

KELLER, A. M.; WIRTHLIN, M. J. Benefits of Complementary SEU Mitigation for the LEON3 Soft Processor on SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, v. 64, n. 1, p. 519–528, Jan 2017. ISSN 1558-1578.

KLEIN, M. Static Power and the Importance of Realistic Junction Temperature Analysis. **Xilinx WP221 (v1.0) March 23, 2005**, 2005. [Accessed November-2021]. Available from Internet: <https://www.xilinx.com/support/documentation/white_papers/wp221.pdf>.

KO, Y. et al. Protecting caches from soft errors: A microarchitect's perspective. **ACM Transactions on Embedded Computing Systems**, v. 16, p. 1–28, 05 2017. Doi:10.1145/3063180.

KOGA, R. et al. Heavy Ion and Proton Induced Single Event Effects on Xilinx Zynq UltraScale+ Field Programmable Gate Array (FPGA). In: **2018 IEEE Radiation Effects Data Workshop (REDW)**. [S.l.: s.n.], 2018. p. 311–315.

KRASICH, M. How to estimate and use MTTF/MTBF would the real MTBF please stand up? In: **2009 Annual Reliability and Maintainability Symposium**. [S.l.: s.n.], 2009. p. 353–359.

KULU, E. **Nanosats Database**. 2022. [Accessed November-2022]. Available from Internet: <www.nanosats.eu>.

KUMAR, U. K.; UMASHANKAR, B. S. Improved hamming code for error detection and correction. In: **2007 2nd International Symposium on Wireless Pervasive Computing**. [S.l.: s.n.], 2007.

LADBURY, R. Statistical Properties of SEE Rate Calculation in the Limits of Large and Small Event Counts. **IEEE Transactions on Nuclear Science**, v. 54, n. 6, p. 2113–2119, 2007.

LAPPAS, V.; KOSTOPOULOS, V. **A Survey on Small Satellite Technologies and Space Missions for Geodetic Applications**. [S.l.]: IntechOpen, 2020.

LATTICE. **CertusPro-NX Family, Preliminary Data Sheet, FPGA-DS-02086-0.81**. 2021.

LAUENSTEIN, J. **JESD57 Test Standard, Procedures for the Measurement of Single-Event Effects in Semiconductor Devices from Heavy-Ion Irradiation Revision Update**. 2016. [Presented at Single Event Effects (SEE) Symposium and Military and Aerospace Programmable Logic Devices (MAPLD) Workshop, San Diego, CA, United States, 23-26 May 2016.].

LEE, D. S.; SWIFT, G.; WIRTHLIN, M. An Analysis of High-Current Events Observed on Xilinx 7-Series and Ultrascale Field-Programmable Gate Arrays. In: **2016 IEEE Radiation Effects Data Workshop (REDW)**. [S.l.: s.n.], 2016. p. 1–5.

LEIBSON, S. **Designing SOCs with Configured Cores: Unleashing the Tensilica Xtensa and Diamond Cores**. Morgan Kaufmann Publishers, 2006. (Electronics & Electrical). ISBN 9780123724984. Available from Internet: <https://books.google.se/books?id=h79BlQEACAAJ>.

LI, J. et al. DuckCore: A Fault-Tolerant Processor Core Architecture Based on the RISC-V ISA. **Electronics**, v. 11, n. 1, 2022. ISSN 2079-9292. Available from Internet: <https://www.mdpi.com/2079-9292/11/1/122>.

LINDOSO, A. et al. A Hybrid Fault-Tolerant LEON3 Soft Core Processor Implemented in Low-End SRAM FPGA. **IEEE Transactions on Nuclear Science**, v. 64, n. 1, p. 374–381, 2017.

LIVANY, M. A.; SALEHI, M.; KARGAR, M. Effect of cache run-time parameters on the reliability of embedded systems. In: **2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)**. [S.l.: s.n.], 2020. p. 1–6. Doi:10.1109/RTEST49666.2020.9140080.

LowRISC. **Ibex: An embedded 32 bit RISC-V CPU core**. 2018. ETH Zurich and University of Bologna. [Accessed April-2023]. Available from Internet: <https://ibex-core.readthedocs.io/en/latest/index.html>.

MAHMOOD, A.; MCCLUSKEY, E. J. Concurrent error detection using watchdog processors - a survey. **IEEE Transactions on Computers**, v. 37, n. 2, p. 160–174, Feb 1988. ISSN 0018-9340.

MAILLARD, P. et al. Total Ionizing Dose and Single-Events characterization of Xilinx 20nm Kintex UltraScale™. In: **2019 19th European Conference on Radiation and Its Effects on Components and Systems (RADECS)**. [S.l.: s.n.], 2019. p. 1–6.

MANSOUR, W.; VELAZCO, R. SEU fault-injection in VHDL-based processors: A case study. In: **2012 13th Latin American Test Workshop (LATW)**. [S.l.: s.n.], 2012. p. 1–5.

MONDRAGóN-TORRES, A.; CHRISTMAN, J. Hard Core vs. Soft Core: A Debate. In: **2012 ASEE Annual Conference Exposition2012 ASEE Annual Conference Exposition**. [S.l.]: American Society for Engineering Education, 2012.

MUKHERJEE, S. S.; KONTZ, M.; REINHARDT, S. K. Detailed design and evaluation of redundant multi-threading alternatives. In: **Proceedings 29th Annual International Symposium on Computer Architecture**. [S.l.: s.n.], 2002. p. 99–110. ISSN 1063-6897.

NANOXPLORE. **NG-ULTRA**. 2022. [Accessed November-2022]. Available from Internet: <https://nanoxplore-wiki.atlassian.net/wiki/spaces/NAN/pages/9961482/NG-ULTRA>.

NASA. **CubeSat101 Basic Concepts and Processes for First-Time CubeSat Developers - NASA CubeSat Launch Initiative**. [S.l.], 2017. [Accessed October-2021]. Available from Internet: <https://www.nasa.gov/sites/default/files/atoms/files/nasa_csli_cubesat_101_508.pdf>.

NASA. **Mars2020 mission Perseverance rover**. 2020. [Accessed April-2023]. Available from Internet: <https://mars.nasa.gov/mars2020/>.

NERI, M. **Design of a fault tolerant instruction decode stage in RISC-V core against soft and hard errors**. Dissertation (Master) — Politecnico di Torino, Corso di laurea magistrale in Ingegneria Elettronica (Electronic Engineering), 2021.

NICOLAIDIS, M. [S.l.: s.n.], 2011. ISSN 0929-1296. ISBN 978-1-4419-6993-4.

NIETO-PEROY, C.; EMAMI, M. R. CubeSat Mission: From Design to Operation. **Applied Sciences**, v. 9, n. 15, 2019. ISSN 2076-3417. Available from Internet: <https://www.mdpi.com/2076-3417/9/15/3110>.

NIKULAINEN, M. **Usage of COTS EEE Components in ESA Space Programs**. ESCCON, 2019. [Accessed October-2021]. Available from Internet: <https://escies.org/download/webDocumentFile?id=67090>.

NORMAND, E. Correlation of inflight neutron dosimeter and SEU measurements with atmospheric neutron model. **IEEE Transactions on Nuclear Science**, v. 48, n. 6, p. 1996–2003, Dec 2001. ISSN 0018-9499.

OH, N.; SHIRVANI, P.; MCCLUSKEY, E. Error detection by duplicated instructions in super-scalar processors. **IEEE Transactions on Reliability**, v. 51, n. 1, p. 63–75, 2002.

OLDHAM, T.; MCLEAN, F. Total ionizing dose effects in MOS oxides and devices. **IEEE Transactions on Nuclear Science**, v. 50, n. 3, p. 483–499, 2003.

OLIVEIRA, Á. et al. Dynamic heavy ions SEE testing of NanoXplore radiation hardened SRAM-based FPGA: Reliability-performance analysis. **Microelectronics Reliability**, v. 100-101, p. 113437, 2019. ISSN 0026-2714. 30th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0026271419304809>.

OLIVEIRA, Á. B. de et al. Lockstep Dual-Core ARM A9: Implementation and Resilience Analysis Under Heavy Ion-Induced Soft Errors. **IEEE Transactions on Nuclear Science**, v. 65, n. 8, p. 1783–1790, 2018.

OSINSKI, L.; LANGER, T.; MOTTOK, J. A survey of fault tolerance approaches on different architecture levels. In: **ARCS 2017; 30th International Conference on Architecture of Computing Systems**. [S.l.: s.n.], 2017. p. 1–9.

OSTLER, P. S. et al. SRAM FPGA Reliability Analysis for Harsh Radiation Environments. **IEEE Transactions on Nuclear Science**, v. 56, n. 6, p. 3519–3526, 2009. Doi:10.1109/TNS.2009.2033381.

OZTURK, Z.; TOPCUOGLU, H. R.; KANDEMIR, M. Studying error propagation on application data structure and hardware. **The Journal of Supercomputing**, v. 16, p. 18691–18724, 2022. Doi:10.1007/s11227-022-04625-x.

PEREZ, F. Overview of ESA Cubesat Missions and Radiation Testing on Cubesat Electronics. In: **2021 Radiation and its Effects on Components and Systems (RADECS) - Short Course**. [S.l.: s.n.], 2021.

PETERSEN, E. et al. Rate prediction for single event effects-a critique. **IEEE Transactions on Nuclear Science**, v. 39, n. 6, p. 1577–1599, 1992.

PETERSEN, E. et al. Rate predictions for single-event effects - critique ii. **IEEE Transactions on Nuclear Science**, v. 52, n. 6, p. 2158–2167, 2005.

PHAM, H. M.; PILLEMENT, S.; PIESTRAK, S. J. Low-overhead fault-tolerance technique for a dynamically reconfigurable softcore processor. **IEEE Transactions on Computers**, v. 62, n. 6, p. 1179–1192, June 2013. ISSN 0018-9340.

PULLINI, A. et al. Mr.Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing. **IEEE Journal of Solid-State Circuits**, v. 54, n. 7, p. 1970–1981, 2019.

QIN, T. et al. Performance Analysis of Nanoelectromechanical Relay-Based Field-Programmable Gate Arrays. **IEEE Access**, v. 6, p. 15997–16009, 2018.

Quinn, H. Challenges in Testing Complex Systems. **IEEE Transactions on Nuclear Science**, v. 61, n. 2, p. 766–786, April 2014. ISSN 1558-1578.

QUINN, H. et al. Domain Crossing Errors: Limitations on Single Device Triple-Modular Redundancy Circuits in Xilinx FPGAs. **IEEE Transactions on Nuclear Science**, v. 54, n. 6, p. 2037–2043, Dec 2007. ISSN 0018-9499.

QUINN, H. et al. Using Benchmarks for Radiation Testing of Microprocessors and FPGAs. **IEEE Transactions on Nuclear Science**, v. 62, n. 6, p. 2547–2554, 2015.

QUINN, H.; WIRTHLIN, M. Validation Techniques for Fault Emulation of SRAM-based FPGAs. **IEEE Transactions on Nuclear Science**, v. 62, n. 4, p. 1487–1500, 2015.

QUINN, H. M. et al. Fault simulation and emulation tools to augment radiation-hardness assurance testing. **IEEE Transactions on Nuclear Science**, v. 60, n. 3, p. 2119–2142, 2013.

RAMOS, A.; MAESTRO, J. A.; REVIRIEGO, P. Characterizing a RISC-V SRAM-based FPGA implementation against Single Event Upsets using fault injection. **Microelectronics Reliability**, v. 78, p. 205 – 211, 2017. ISSN 0026-2714.

RAMOS, A. et al. An ALU Protection Methodology for Soft Processors on SRAM-Based FPGAs. **IEEE Transactions on Computers**, v. 68, n. 9, p. 1404–1410, 2019.

RAMOS, A. et al. Efficient Protection of the Register File in Soft-Processors Implemented on Xilinx FPGAs. **IEEE Transactions on Computers**, v. 67, n. 2, p. 299–304, 2018.

RAMOS, P. F. et al. Assessing the Static and Dynamic Sensitivity of a Commercial Off-the-Shelf Multicore Processor for Noncritical Avionic Applications. **Journal of Nanotechnology**, 2018. ISSN 1687-9503. [Accessed July-2022]. Available from Internet: <https://doi.org/10.1155/2018/2926392>.

RECH, P. et al. Impact of GPUs Parallelism Management on Safety-Critical and HPC Applications Reliability. In: **2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks**. [S.l.: s.n.], 2014. p. 455–466.

REINHARDT, S. K.; MUKHERJEE, S. S. Transient fault detection via simultaneous multithreading. In: **Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No.RS00201)**. [S.l.: s.n.], 2000. p. 25–36. ISSN 1063-6897.

REIS, G. et al. SWIFT: software implemented fault tolerance. In: **International Symposium on Code Generation and Optimization**. [S.l.: s.n.], 2005. p. 243–254.

REZGUI, S. et al. Estimating error rates in processor-based architectures. **IEEE Transactions on Nuclear Science**, v. 48, n. 5, p. 1680–1687, 2001.

RODRIGUES, C. et al. Towards a Heterogeneous Fault-Tolerance Architecture based on Arm and RISC-V Processors. In: **IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society**. [S.l.: s.n.], 2019. v. 1, p. 3112–3117. Doi:10.1109/IECON.2019.8926844.

ROGENMOSER, M.; TORTORELLA, Y. **PULP in Space: Leveraging an Open Hardware Platform to build reliable Multicore SoCs for Space**. 2022. [Accessed March-2023]. Available from Internet: <http://microelectronics.esa.int/riscv/rvws2022/presentations/05-ESA_PULP.pdf>.

RONAK, B.; FAHMY, S. A. Evaluating the efficiency of DSP Block synthesis inference from flow graphs. In: **22nd International Conference on Field Programmable Logic and Applications (FPL)**. [S.l.: s.n.], 2012. p. 727–730.

SANCHEZ-ELEZ, M. et al. Radiation-hardened DSP configurations for implementing arithmetic functions on FPGA. In: **2016 Design, Automation Test in Europe Conference Exhibition (DATE)**. [S.l.: s.n.], 2016. p. 1501–1504.

SANTINI, T. et al. Reducing embedded software radiation-induced failures through cache memories. In: **2014 19th IEEE European Test Symposium (ETS)**. [S.l.: s.n.], 2014. p. 1–6.

SANTOS, D. A. et al. A Low-Cost Fault-Tolerant RISC-V Processor for Space Systems. In: **2020 15th Design Technology of Integrated Systems in Nanoscale Era (DTIS)**. [S.l.: s.n.], 2020. p. 1–5. Doi: 10.1109/DTIS48698.2020.9081185.

SANTOS, D. A. et al. Neutron Irradiation Testing and Analysis of a Fault-Tolerant RISC-V System-on-Chip. In: **2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)**. [S.l.: s.n.], 2022.

SARI, A.; PSARAKIS, M. A fault injection platform for the analysis of soft error effects in FPGA soft processors. In: **2016 IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)**. [S.l.: s.n.], 2016. p. 1–6.

SCHMIDT, A. G.; FRENCH, M.; FLATLEY, T. Radiation hardening by software techniques on FPGAs: Flight experiment evaluation and results. In: **2017 IEEE Aerospace Conference**. [S.l.: s.n.], 2017. p. 1–8.

SHUKLA, S.; RAY, K. C. A Low-Overhead Reconfigurable RISC-V Quad-Core Processor Architecture for Fault-Tolerant Applications. **IEEE Access**, v. 10, p. 44136–44146, 2022. Doi:10.1109/ACCESS.2022.3169495.

SIEWIOREK, D. P.; SWARZ, R. S. **Reliable Computer Systems: Design and Evaluation, Third Edition**. [S.l.]: A K Peters/CRC Press, 1998. ISBN 9780429065101.

SONG, W. **Untethered lowRISC tutorial**. 2015. [Accessed November-2022]. Available from Internet: <https://www.lowrisc.org/docs/untether-v0.2>.

SRINIVASAN, G.; TANG, H.; MURLEY, P. Parameter-free, predictive modeling of single event upsets due to protons, neutrons, and pions in terrestrial cosmic rays. **IEEE Transactions on Nuclear Science**, v. 41, n. 6, p. 2063–2070, 1994.

STERPONE, L.; VIOLANTE, M. Static and Dynamic Analysis of SEU Effects in SRAM-Based FPGAs. In: **12th IEEE European Test Symposium (ETS'07)**. [S.l.: s.n.], 2007. p. 159–164.

STODDARD, A. et al. A Hybrid Approach to FPGA Configuration Scrubbing. **IEEE Transactions on Nuclear Science**, v. 64, n. 1, p. 497–503, 2017.

SYNOPSYS. **Synplify Pro and Premier Datasheet**. 2015. [Accessed October-2022]. Available from Internet: <synopsys.com/content/dam/synopsys/implementation& signoff/datasheets/synplify-pro-premier.pdf>.

SYNOPSYS. **Synplify Pro and Premier Datasheet**. 2015. [Accessed April-2023]. Available from Internet: <https://www.synopsys.com/content/dam/synopsys/ implementation&signoff/datasheets/synplify-pro-premier.pdf>.

TAMBARA et al. Heavy Ions Induced Single Event Upsets Testing of the 28 nm Xilinx Zynq-7000 All Programmable SoC. In: **2015 IEEE Radiation Effects Data Workshop (REDW)**. [S.l.: s.n.], 2015. p. 29–34.

TAMBARA, L. A. **Analyzing the Impact of Radiation-induced Failures in All Programmable System-on-Chip Devices**. Thesis (PhD) — Universidade Federal do Rio Grande do Sul, Instituto de Informática, Programa de Pós-Graduação em Microeletrônica, Brazil, 2017.

TARRILLO, J. et al. Neutron Cross-Section of N-Modular Redundancy Technique in SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, v. 61, n. 4, p. 1558–1566, 2014.

TAUSCH, J. et al. Neutron Induced Micro SEL Events in COTS SRAM Devices. In: **2007 IEEE Radiation Effects Data Workshop**. [S.l.: s.n.], 2007. p. 185–188.

TONFAT, J. et al. Analyzing the Influence of the Angles of Incidence and Rotation on MBU Events Induced by Low LET Heavy Ions in a 28-nm SRAM-Based FPGA. **IEEE Transactions on Nuclear Science**, v. 64, n. 8, p. 2161–2168, 2017.

TONFAT, J. et al. Analyzing the Effectiveness of a Frame-Level Redundancy Scrubbing Technique for SRAM-based FPGAs. **IEEE Transactions on Nuclear Science**, v. 62, n. 6, p. 3080–3087, 2015.

TONFAT, J. et al. Method to Analyze the Susceptibility of HLS Designs in SRAM-Based FPGAs Under Soft Errors. In: **Proceedings of the 12th International Symposium on Applied Reconfigurable Computing - Volume 9625**. Berlin, Heidelberg: Springer-Verlag, 2016. p. 132–143. ISBN 9783319304809.

TONFAT, J. et al. Soft error susceptibility analysis methodology of HLS designs in SRAM-based FPGAs. **Microprocessors and Microsystems**, v. 51, p. 209–219, 2017. ISSN 0141-9331. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S014193311730217X>.

TRAD. **OMERE Radiation Software**. 2022. [Accessed July-2022]. Available from Internet: <https://www.trad.fr/en/space/omere-software/>.

TSELONIS, S. et al. Microprocessor reliability-performance tradeoffs assessment at the microarchitecture level. In: **2016 IEEE 34th VLSI Test Symposium (VTS)**. [S.l.: s.n.], 2016. p. 1–6.

UNIVERSITY OF JYVÄSKYLÄ. **RADiation Effects Facility**. 2022. [Accessed April-2023]. Available from Internet: <https://www.jyu.fi/science/en/physics/research/infrastructures/accelerator-laboratory/radiation-effects-facilitys>.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN. **Light Ion Facility (LIF)**. 2023. [Accessed April-2023]. Available from Internet: <https://uclouvain.be/en/research-institutes/irmp/crc/light-ion-facility-lif.html>.

UNOOSA. Online Index of Objects Launched into Outer Space. **United Nations Office for Outer Space Affairs (UNOOSA)**, 2022. [Accessed November-2022]. Available from Internet: <https://www.unoosa.org/oosa/osoindex/search-ng.jspx>.

UPADHYAYA, J. S.; SALUJA, K. K. A watchdog processor based general rollback technique with multiple retries. **IEEE Transactions on Software Engineering**, SE-12, n. 1, p. 87–95, 1986.

VARAPRASAD, B. et al. Design of Watchdog Circuit using Decision Trees for Detection of Single Event Upsets in Processor. In: **2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)**. [S.l.: s.n.], 2021. p. 1306–1311.

VELAZCO, R.; FAURE, F. **"Error Rate Prediction of Digital Architectures: Test Methodology and Tools"**. Dordrecht: Springer Netherlands, 2007. 233–258 p. [Accessed June-2022]. ISBN 978-1-4020-5646-8. Available from Internet: <https://doi.org/10.1007/978-1-4020-5646-8_11>.

VELAZCO, R.; REZGUI, S.; ECOFFET, R. Predicting error rate for microprocessor-based digital architectures through C.E.U. (Code Emulating Upsets) injection. **IEEE Transactions on Nuclear Science**, v. 47, n. 6, p. 2405–2411, 2000.

VIOLANTE, M. et al. A Low-Cost Solution for Deploying Processor Cores in Harsh Environments. **IEEE Transactions on Industrial Electronics**, v. 58, n. 7, p. 2617–2626, July 2011. ISSN 0278-0046.

WALSEMANN, A. et al. STRV — a radiation hard RISC-V microprocessor for high-energy physics applications. **Journal of Instrumentation**, IOP Publishing, v. 18, n. 02, p. C02032, feb 2023. Doi:10.1088/1748-0221/18/02/C02032.

WARNER, J. et al. Displacement damage correlation of proton and silicon ion radiation in GaAs. **IEEE Transactions on Nuclear Science**, v. 52, n. 6, p. 2678–2682, 2005.

WATERMAN et al. **The RISC-V instruction set manual, volume I: User-level ISA, version 2.0. Technical Report UCB/EECS- 2014-54**. EECS Department, University of California, Berkeley, 2014.

WEIBULL, W. A Statistical Distribution Function of Wide Applicability. **Journal of Applied Mechanics**, p. 293–297, 1951.

WEULERSSE, C. et al. Prediction of proton cross sections for SEU in SRAMs and SDRAMs using the METIS engineer tool. **Microelectronics Reliability**, v. 55, n. 9, p. 1491–1495, 2015. ISSN 0026-2714. Proceedings of the 26th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0026271415300718>.

WILSON, A. E. et al. Neutron Radiation Testing of a TMR VexRiscv Soft Processor on SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, v. 68, n. 5, p. 1054–1060, 2021.

WILSON, A. E. et al. Neutron Radiation Testing of RISC-V TMR Soft Processors on SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, v. 70, n. 4, p. 603–610, 2023. Doi:10.1109/TNS.2023.3235582.

WILSON, A. E.; WIRTHLIN, M. Neutron Radiation Testing of Fault Tolerant RISC-V Soft Processor on Xilinx SRAM-based FPGAs. In: **2019 IEEE Space Computing Conference (SCC)**. [S.l.: s.n.], 2019. p. 25–32.

WILSON, A. E.; WIRTHLIN, M. Fault Injection of TMR Open Source RISC-V Processors using Dynamic Partial Reconfiguration on SRAM-based FPGAs. In: **2021 IEEE Space Computing Conference (SCC)**. [S.l.: s.n.], 2021. p. 1–8. Doi:10.1109/SCC49971.2021.00008.

WIRTHLIN, M. High-Reliability FPGA-Based Systems: Space, High-Energy Physics, and Beyond. **Proceedings of the IEEE**, v. 103, n. 3, p. 379–389, March 2015. ISSN 1558-2256.

XILINX. **7-Series Architecture Overview**. 2013. [Accessed November-2021]. Available from Internet: <http://xilinx.eetrend.com/files-eetrend-xilinx/forum/201509/9204-20390-7_series_architecture_overview.pdf>.

XILINX. **7 Series FPGAs Configurable Logic Block, User Guide, UG474 (v1.8), September 27, 2016**. 2016. [Accessed November-2021]. Available from Internet: <https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf>.

XILINX. **Zynq-7000 All Programmable SoC Technical Reference Manual UG585 (v1.11)**. [S.l.], 2016.

XILINX. **7 Series DSP48E1 Slice, User Guide, UG479 (v1.10) March 27, 2018**. 2018. [Accessed November-2021]. Available from Internet: <https://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf>.

XILINX. **7 Series FPGAs Data Sheet: Overview, Product Specification, DS180 (v2.6.1), September 8, 2020**. 2020. [Accessed October-2021]. Available from Internet: <https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf>.

XILINX. **MicroBlaze Processor Reference Guide, UG984 (v2021.2), October 27, 2021**. 2021. [Accessed November-2021]. Available from Internet: <https://www.xilinx.com/support/documentation/sw_manuals/xilinx2021_2/ug984-vivado-microblaze-ref.pdf>.

XILINX. **UltraFast Design Methodology Guide for Xilinx FPGAs and SoCs, UG949 (v2020.2), February 18, 2021**. 2021. [Accessed November-2021]. Available from Internet: <https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_2/ug949-vivado-design-methodology.pdf>.

XILINX. **Vivado Design Suite User Guide, Implementation, UG904 (v2021.1) August 30, 2021**. 2021. [Accessed November-2021]. Available from Internet: <https://www.xilinx.com/content/dam/xilinx/support/documentation/sw_manuals/xilinx2021_2/ug904-vivado-implementation.pdf>.

XILINX. **Zynq-7000 SoC Technical Reference Manual, UG585 (v1.13) April 2, 2021**. 2021. [Accessed November-2021]. Available from Internet: <https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf>.

XILINX. **Device reliability report, UG116 (v10.16) June 29, 2022**. 2022. [Accessed October-2022]. Available from Internet: <https://docs.xilinx.com/viewer/book-attachment/66EpZI5rDuOf2Bd0djXvSg/wHyVZkT9321xbDF9suFemA>.

XILINX. **Soft Error Mitigation Controller, v4.1 LogiCORE IP Product Guide, Vivado Design Suite, PG036, May 2022**. 2022. [Accessed October-2022]. Available from Internet: <https://www.xilinx.com/content/dam/xilinx/support/documents/ip_documentation/sem/v4_1/pg036_sem.pdf>.

XILINX. **UltraScale Architecture and Product Data Sheet: Overview, Product Specification, DS890 (v4.3), November 7, 2022**. 2022. [Accessed April-2023]. Available from Internet: <https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf>.

XILINX. **UltraScale Architecture Soft Error Mitigation Controller v3.1, LogiCORE IP Product Guide (PG187)**. 2022.

Xilinx. **Zynq UltraScale+ Device Packaging and Pinouts, UG1075 (v1.11)**. 2022.

XILINX. **Zynq UltraScale+ MPSoC Data Sheet: Overview, DS891 (v1.10)**. 2022.

YIU, J. **System-on-Chip Design with Arm® Cortex®-M Processors**. [S.l.]: Arm Education Media, 2019. ISBN 978-1-911531-19-7.

Yuanwen Huang; Mishra, P. Reliability and energy-aware cache reconfiguration for embedded systems. In: **2016 17th International Symposium on Quality Electronic Design (ISQED)**. [S.l.: s.n.], 2016. p. 313–318.

ZIADE, H.; AYOUBI, R.; VELAZCO, R. A survey on fault injection techniques. **Int. Arab J. Inf. Technol.**, v. 1, p. 171–186, 01 2004.

ZIEGLER, J. F. **SRIM - The Stopping and Range of Ions in Matter**. 2013. [Accessed July-2022]. Available from Internet: <http://www.srim.org>.

APPENDIX A — PUBLICATIONS

The evaluation performed on the RISC-V Rocket soft processor implemented in the Zynq-7000 FPGA, detailed in Chapter 6, was presented at the 2019 Radiation Effects on Components and Systems (RADECS) conference and published in the 2020 IEEE Transactions on Nuclear Science (TNS) journal:

- Á. B. de Oliveira et al., "Evaluating Soft Core RISC-V Processor in SRAM-Based FPGA Under Radiation Effects," in IEEE Transactions on Nuclear Science, vol. 67, no. 7, pp. 1503-1510, July 2020, doi: 10.1109/TNS.2020.2995729.

The work performed on the open source NOEL-V soft processor in a Zynq UltraScale+ FPGA under protons testing and emulation fault injection was presented at the 2022 RADECS conference and published in the 2023 IEEE TNS journal:

- Á. B. De Oliveira and F. L. Kastensmidt, "Evaluating Fault Tolerant Techniques on COTS RISC-V NOEL-V Processor in Zynq UltraScale+ FPGA under Proton Testing," in IEEE Transactions on Nuclear Science, doi: 10.1109/TNS.2023.3281396.

The evaluation of the commercial fault tolerant NOEL-VFT soft processor with external FPGA supervisor (GRSCRUB IP) under proton testing in a Kintex UltraScale FPGA was presented at 2022 RADECS conference:

- Á. B. de Oliveira et al., "NOEL-V FT and GRSCRUB IP: Fault Tolerance Characterization of a Complex System-on-Chip on Xilinx Kintex UltraScale FPGA," in 2022 European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2022. [To be published].

The following papers were published by the Ph.D. student and are not directly related to this thesis proposal:

- Á. B. de Oliveira et al., "Multi chips heavy ions SEE testing of the COTS Myriad-2 vision processing unit," in 2021 European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2021.
- Á. B. de Oliveira et al., "Dynamic heavy ions SEE testing of NanoXplore radiation hardened SRAM-based FPGA: Reliability-performance analysis," in Microelectronics Reliability, v. 100-101, p. 113437, 2019. ISSN 0026-2714. 30th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis.
- Á. B. de Oliveira et al., "Lockstep Dual-Core ARM A9: Implementation and Re-

silience Analysis Under Heavy Ion-Induced Soft Errors," in IEEE Transactions on Nuclear Science, v. 65, n. 8, p. 1783–1790, 2018.

Additionally, the Ph.D. student also published the following works in project collaboration:

- F. Benevenuti, F. Kastensmidt, A. Oliveira, N. Added, V. Aguiar, N. Medina, M. Guazzelli, "Robust Convolutional Neural Networks in SRAM-based FPGAs: a Case Study in Image Classification," in Journal of Integrated Circuits and Systems (Jics), 2021, doi: 10.29292/jics.v16i2.504.
- F. Benevenuti, A. Oliveira et al., "Heavy Ions Testing of an All-Convolutional Neural Network for Image Classification Evolved by Genetic Algorithms and Implemented on SRAM-Based FPGA," 2019 19th European Conference on Radiation and Its Effects on Components and Systems (RADECS), 2019, pp. 1-4, doi: 10.1109/RADECS47380.2019.9745650.
- L. A. C. Benites, F. Benevenuti, A. Oliveira et al., "Reliability Calculation With Respect to Functional Failures Induced by Radiation in TMR Arm Cortex-M0 Soft-Core Embedded Into SRAM-Based FPGA," in IEEE Transactions on Nuclear Science, vol. 66, no. 7, pp. 1433-1440, July 2019, doi: 10.1109/TNS.2019.2921796.
- Rodrigues, G.S., Barros de Oliveira, Á., Kastensmidt, F.L. et al., "Assessing the Reliability of Successive Approximate Computing Algorithms under Fault Injection," in J Electron Test 35, 367–381, 2019, doi: 10.1007/s10836-019-05806-y.
- G. S. Rodrigues, A. Barros de Oliveira, A. Bosio, F. L. Kastensmidt and E. Pignaton de Freitas, "ARFT: An Approximative Redundant Technique for Fault Tolerance," in 2018 Conference on Design of Circuits and Integrated Systems (DCIS), 2018, pp. 1-6, doi: 10.1109/DCIS.2018.8681499.
- Rodrigues, G.S., de Oliveira, Á.B., Kastensmidt, F.L., Bosio, A., "Analyzing the Use of Taylor Series Approximation in Hardware and Embedded Software for Good Cost-Accuracy Tradeoffs," in Applied Reconfigurable Computing. Architectures, Tools, and Applications. ARC 2018. Lecture Notes in Computer Science, vol 10824. Springer, Cham, doi: 10.1007/978-3-319-78890-6_52.