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

JIMMY FERNANDO TARRILLO OLANO

Exploring the Use of Multiple Modular Redundancies for Masking Accumulated Faults in SRAM-Based FPGAs

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Dr. Fernanda Kastendmidt
Advisor

Porto Alegre, June 2014

CIP – CATALOGING-IN-PUBLICATION

Tarrillo Olano, Jimmy Fernando

Exploring the Use of Multiple Modular Redundancies for Masking Accumulated Faults in SRAM-Based FPGAs / Jimmy Fernando Tarrillo Olano. – Porto Alegre: PPGC da UFRGS, 2014.

112 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2014. Advisor: Fernanda Kastendmidt.

1. Fault tolerance. 2. FPGA. I. Kastendmidt, Fernanda. II. Título.

"Absence of evidence is not
evidence of absences."
— Carl Sagan

"Insanity: doing the same thing over
and over again and expecting different results."
— Albert Einstein

"It is not the strongest of the species that survives,
nor the most intelligent that survives.
It is the one that is most adaptable to change."
— Charles Darwin

ACKNOWLEDGMENT

More than seven years ago I arrived in Brazil, and today I am extremely happy, nostalgic, and grateful to close this wonderful period of my life, which has offered me so much more than I ever expected. I was fortunate to come to a county that has opened its doors to me, inviting me to take part of its projects. I have met wonderful people along the way, including outstanding professors, not only regarding their professionalism but also their human qualities, and also colleagues who became friends and who supported me and inspired me to become a better person. It is impossible to enumerate in such few words all the people I would like to thank, but I will do my best.

First, I want to thank God for taking care of me and my family, for guiding me in my life and always give me much more than I expected. I also want to thank my parents Atilano and Maria Flor, for their unconditional support and a lifetime of lessons of honesty, effort, willpower, work and joy.

I want to give special thanks to my beloved wife, Micaela, for helping me, for supporting me and for motivating me in this journey of research and travels that began in Peru 8 years ago. Thanks for giving me strength in difficult times, and for always walking along my side despite often being thousands of miles away due to the circumstances.

I also want to thank my advisor Fernanda Lima Kastensmidt for all these four years of work, trust, patience and lessons. This is a great opportunity for me to thank her for accepting me as a research and PhD candidate from the first email I sent her, and for pushing me forward me and encouraging me with energy in my academic and personal decisions. Thank you for allowing me to learn and grow as a researcher and as a person.

My eternal gratitude to the Universidade Federal do Rio Grande do Sul, to the Informatic Institute, to the Post Graduation in Computer Science, and to the Brazilian government institutions CAPES and CNPq, for putting their facilities at my disposal to develop my research. Particularly, I thank these institutions for helping me participate in national and international conferences, and collaborate with excellent laboratories around the world. All these experiences have enriched my research career.

Finally, I want to thank Paolo Rech and my colleagues Jorge Tonfat, Jose Rodrigo Azambuja, Lucas Tambara, Anelise Kologeski, Carol Concatto, Samuel Pagliarini and Gracieli Posser, for their consistent willingness and ability to offer always their helping hand. Thank you for helping me improve my presentations, my work, and for for their help in troubled times. Words can not explain how thankful I am for your patience, and for your genuine interest in my doubts, whether they were basic, or complex engineering and science concepts, or even in almost philosophical issues. I feel truly humbled and proud to have worked with you.

ABSTRACT

Soft errors in the configuration memory bits of SRAM-based FPGAs are an important issue due to the persistence effect and its possibility of generating functional failures in the implemented circuit. Whenever a configuration memory bit cell is flipped, the soft error will be corrected only by reloading the correct configuration memory bitstream. If the correct bitstream is not loaded, persistent soft errors can accumulate in the configuration memory bits provoking a system functional failure in the user's design, and consequently can cause a catastrophic situation. This scenario gets worse in the event of multi-bit upset, whose probability of occurrence is increasing in new nano-metric technologies.

Traditional strategies to deal with soft errors in configuration memory are based on the use of any type of triple modular redundancy (TMR) and the scrubbing of the memory to repair and avoid the accumulation of faults. The high reliability of this technique has been demonstrated in many studies, however TMR is aimed at masking single faults. The technology trend makes lower the dimensions of the transistors, and this leads to increased susceptibility to faults. In this new scenario, it is commoner to have multiple to single faults in the configuration memory of the FPGA, so that the use of TMR is inappropriate in high reliability applications. Furthermore, since the fault rate is increasing, scrubbing rate also needs to be incremented, leading to the increase in power consumption.

Aiming at coping with massive upsets between sparse scrubbing, this work proposes the use of a multiple redundancy system composed of n identical modules, known as n-modular redundancy (nMR), operating in tandem and an innovative self-adaptive voter to be able to mask multiple upsets in the system. The main drawback of using modular redundancy is its high cost in terms of area and power consumption. However, area overhead is less and less problem due the higher density in new technologies. On the other hand, the high power consumption has always been a handicap of FPGAs. In this work we also propose a model to prevent power overhead caused by the use of multiple redundancy in SRAM-based FPGAs. The capacity of the proposal to tolerate multiple faults has been evaluated by radiation experiments and fault injection campaigns of study case circuits implemented in a 65nm technology commercial FPGA. Finally we demonstrate that the power overhead generated by the use of nMR in FPGAs is much lower than it is discussed in the literature.

Keywords: Fault tolerance, FPGA.

Explorando Redundância Modular Múltipla para mascarar falhas acumuladas em FPGAs baseados em SRAM

RESUMO

Os erros transientes nos bits de memória de configuração dos FPGAs baseados em SRAM são um tema importante devido ao efeito de persistência e a possibilidade de gerar falhas de funcionamento no circuito implementado. Sempre que um bit de memória de configuração é invertido, o erro transiente será corrigido apenas recarregando o bitstream correto da memória de configuração. Se o *bitstream* correto não for recarregando, erros transientes persistentes podem se acumular nos bits de memória de configuração provocando uma falha funcional do sistema, o que consequentemente, pode causar uma situação catastrófica. Este cenário se agrava no caso de falhas múltiplas, cuja probabilidade de ocorrência é cada vez maior em novas tecnologias nano-métricas.

As estratégias tradicionais para lidar com erros transientes na memória de configuração são baseadas no uso de redundância modular tripla (TMR), e na limpeza da memória (*scrubbing*) para reparar e evitar a acumulação de erros. A alta eficiência desta técnica para mascarar perturbações tem sido demonstrada em vários estudos, no entanto o TMR visa apenas mascarar falhas individuais. Porém, a tendência tecnológica conduz à redução das dimensões dos transistores o que causa o aumento da susceptibilidade a falhos. Neste novo cenário, as falhas multiplas são mais comuns que as falhas individuais e consequentemente o uso de TMR pode ser inapropriado para ser usado em aplicações de alta confiabilidade. Além disso, sendo que a taxa de falhas está aumentando, é necessário usar altas taxas de reconfiguração o que implica em um elevado custo no consumo de potência.

Com o objetivo de lidar com falhas massivas acontecidas na mem[oria de configuração, este trabalho propõe a utilização de um sistema de redundância múltipla composto de n módulos idênticos que operam em conjunto, conhecido como (nMR), e um inovador *votador* auto-adaptativo que permite mascarar múltiplas falhas no sistema. A principal desvantagem do uso de redundância modular é o seu elevado custo em termos de área e o consumo de energia. No entanto, o problema da sobrecarga em área é cada vez menor devido à maior densidade de componentes em novas tecnologias. Por outro lado, o alto consumo de energia sempre foi um problema nos dispositivos FPGA.

Neste trabalho também propõe-se um modelo para prever a sobrecarga de potência causada pelo uso de redundância múltipla em FPGAs baseados em SRAM. A capacidade de tolerar múltiplas falhas pela técnica proposta tem sido avaliada através de experimentos de radiação e campanhas de injeção de falhas de circuitos para um estudo de caso implementado em um FPGA comercial de tecnologia de 65nm. Finalmente, é demostrado que o uso de nMR em FPGAs é uma atrativa e possível solução em termos de potencia, área e confiabilidade medida em unidades de FIT e *Mean Time between Failures* (MTBF).

Palavras-chave: Falhas múltiplas, efeitos da radiaçao, FPGAs, confiabilidade, potência.

LIST OF ABBREVIATIONS AND ACRONYMS

ASIC	Application Specific Iintegrated Circuit
BRAM	Block RAM
CGR	Galactic Cosmic Rays
CRC	Cyclic Redundancy Check
CLB	Configurable Logic Block
CMOS	Complementary Metal Oxide Silicon
COTS	Commercial Off The Shelf
CP	Configuration Port
DDR	Diversity Redundancy
DPR	Dynamic Partial Reconfiguration
CUT	Circuit Under Test
DMR	Dual Modular Redundancy
DTMR	Diversity TMR
ECC	Error Correction Code
ESF	Error Status Flag
FFO	Fault Free Output
FIT	Failure In Time
FPGA	Field Programmable Gate Array
FSM	Finit State Machine
ICAP	Internal Configuration Access Port
LEO	Low Earth Orbit
LET	Linear Energy Transfer
LFSR	Linear Feedback Shift Register
LUT	Look Up Table
MBU	Multiple Bit Upset
MTBF	Mean Time Between Failures

MTTF	Mean Time To Failure
MTTR	Mean Time to repair
MOS	Metal Oxide Silicon
NMF	Non Masked Fault
NMR	N Modular Redundancy
NMOS	n-channel Metal Oxide Silicon
NRE	Non-recurring Engineering
PC	Personal Computer
QFDR	Quadruple Force Decide Redundancy
QL	Quadded Logics
RAD	Radiation Absorbed Dose
RAM	Random Access Memory
SAv	Self-Adapted Voter
SEM	Single Event Upset
SEU	Soft Error Mitigation
SEE	Single Event Effects
SET	Single Event Transient
SEL	Single Event Latchup
SEB	Single Event Burnout
SEGR	Single Event Gate Rupture
SEFI	Single Event Functional Interrupt
TID	Total Ionization Dose
TMR	Triple Modular Redundancy

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are an attractive solution for aerospace applications due to its capacity to integrate complex systems into a single chip and the versatility to reconfigure the system during its lifetime. Nevertheless, the main concern for the use of FPGAs in aerospace (and other critical) applications is the high susceptibility to radiation effects that may provoke errors with catastrophic consequences. This work is related to the use of multiple redundancy in circuits implemented in state-of-the-art SRAM-based FPGAs to cope with radiation effects.

1.1 Radiation Effects in MOS-based devices

The development of technology induced society to use electronic systems in almost all daily activities. Currently, transport systems, health care equipment and communication systems and even electrical appliances can make use of complex computer systems. However, the consequences of an error in any of these equipments do not have the same level of criticality. A system can be classified as critical when an error in its behavior may cause life-threatening or generate very high economic losses. An example of critical system is the circuit that controls the brake system of modern cars since an error in its functionality can endanger the lives of people. Another example is the on-board computer of an aircraft. The space-crafts are also considered critical systems because of its costs, implications, and because it is almost impossible to be repaired in case of failure.

One source of faults in electronic devices is the radiation which may be defined as energy in transit in the form of high-speed particles and electromagnetic waves. The main sources of radiation in the solar system are the Galactic Cosmic Rays (GCR) and the solar activity (BARTH; DYER; STASSINOPOULOS, 2003). GCR is composed basically by protons and ionized atoms with energy about 11 MeV to 100 MeV and low flux rate. On the other hand, solar flares produce streams of energized particles called solar wind, composed mainly by protons, alpha particles and heavy ions, where their intensity depends on the solar activity. When the energized particles go against the Earth, they are stopped by the Van Allen belts generated by the earth magnetic field. The effect of such interaction is that one portion of particles are reflected back, others are trapped (trapped particles) into the Van Allen belts, and a small part (mainly energized neutrons) hit the Earth surface. Trapped particles are composed mainly by protons (up 30 MeV) and electrons (up 10 MeV). Low orbit satellites work in the Van Allen belt region, so the effect of radiation in electronic devices must to be considered during its design. Electrons and heavy ions can direct ionize the circuit, while protons mainly produce secondary particles that ionize the circuit.

At the ground level, the neutrons are the most frequent cause of upset as shown by

(NORMAND, 2001, 1996). Neutrons are created by cosmic ion interactions with the oxygen and nitrogen in the upper atmosphere. The neutron flux is strongly dependent on key parameters such as altitude, latitude and longitude. Table 1.1 from (QUINN; GRAHAM, 2005) shows the variation of the neutron flux (in $\frac{neutrons}{cm^2 \cdot hr}$) with energy higher than 10 MeV according to the latitude and altitude from sea level. Notice that for almost all cases, the neutron flux increases with higher altitudes.

Table 1.1: Terrestrial radiation levels.

Location	Altitude (feet)	> 10 MeV Flux ($\frac{n}{cm^2 \cdot hr}$)
San Jose	0	14.40
Albuquerque	5,200	53.28
Cheyenne	6,100	71,40
Los Alamos	7,200	90.00
Leadville	10,200	180.04
White Mountain	12,000	338.40
Mauna Kea	13,500	229.57
Commercial Aircraft	40,000	2041.24
Military Aircraft	60,000	4680.00

(QUINN; GRAHAM, 2005)

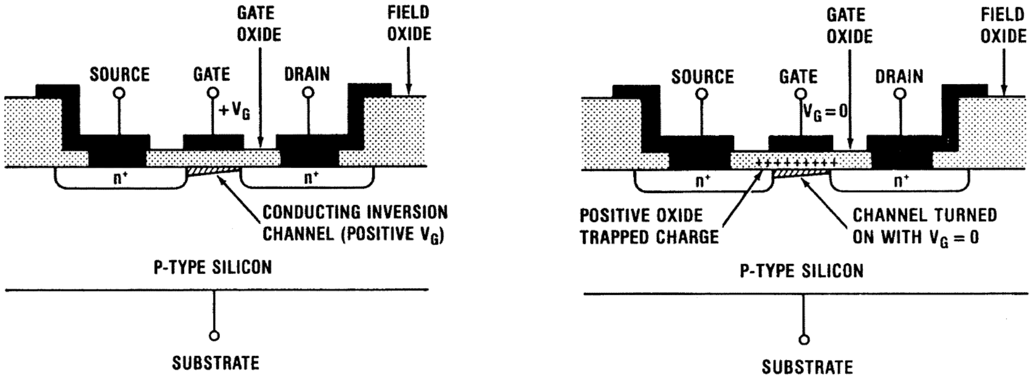
When an energetic particle traverses the material of an electronic device, it deposits energy along its path through the device. This energy is measured as a linear energy transfer (LET), which is defined as the amount of energy deposited per unit of distance traveled, normalized to the material's density. It is usually expressed in MeV-cm^2/mg. The ionized track contains equal numbers of electrons and holes (pairs electron-hole). The total number of charges is proportional to the LET of the incoming particle.

Integrated circuits operating in a space environment can be affected by permanent and transient effects. One cumulative effect is the long term ionizing damage due to protons and electrons, known as Total Ionizing Dose (TID). TID represents the degradation in performance of transistors as TID modifies the voltage threshold (Vth) of the transistor shifts and increases the leakage current (ANGHEL; NICOLAIDIS, 2008; DODD; MASSENGILL, 2003).

Figure 1.1 shows the normal operation of an n-channel Metal Oxide Silicon (NMOS) transistor, and the fault operation of the same transistor caused by TID effects. In normal operation (Figure 1.1a), the transistor may conduce (turned on) if a positive voltage is applied to the gate terminal: an electric field is created between the gate and the silicon substrate, which causes that the majority carriers in the substrate (holes in p-type) will be repelled from the gate-oxide substrate interface and minority carriers (electrons) will be attracted, forming what is called an inversion layer. Then, when a potential difference is applied between the source and drain terminals, the inversion layer provides a low resistance path for electrons to flow. Nevertheless, radiation makes that the gate oxide becomes ionized by the dose it absorbs due to the radiation induced trapped charges in the gate-oxide. The trapped charges in the gate-oxide generate additional space charge fields at the oxide substrate interface. After a sufficient dose, a large positive charge builds up, having the same effect as if a positive voltage was applied to the gate terminal (Figure 1.1b). Therefore, the transistor remains on permanently regardless of the value of voltage at the gate resulting in device failure (OLDHAM; MCLEAN et al., 2003; SMITH; MOSTERT, 2007).

TID is measured in radiation absorbed dose (rad) units, which is the amount of energy deposited in the material. For space vehicles or satellites in Low Earth Orbit (LEO), typical dose rates due to trapped Van Allen electrons and protons are up to 10 krad/year (ASADI; TAHOORI, 2005; ATHAN; LANDIS; AL-ARIAN, 1996). In (KASTENSMIDT et al., 2011) was reported around 30% of propagation delay degradation in a 130-nm commercial device for an accumulated dose of 40 krad(Si). In (TARRILLO et al., 2011) an embedded system implemented also in a 130-nm commercial device works properly until 47 krad(Si) of accumulated dose and stop to work in 63 krad(Si).

Figure 1.1: Radiation-induced charging of the gate oxide of n-channel MOSFET.



(a) Normal operation of NMOS transistor.

(b) Failure operation caused by TID effects.

On the other hand, the interaction of the charged particles with the transistor may provoke transient and permanent effects. The effects that are caused by a single event interaction are called Single Event Effects (SEE), and they can be transient as the Single Event Upset (SEU) and Single Event Transient (SET), or permanent as single event latchup (SEL), single event gate rupture (SEGR), or single event burnout (SEB) (BERG, 2006; DODD et al., 2004).

If an energetic particle passes through the pn-junction of a CMOS transistor in the off state, a short low resistance path is momentarily created between the substrate and the struck drain terminal. The amount of charge that is collected produces a transient current pulse that lasts until the deposited charge disappears by recombination or is conducted away via open current paths to V_{DD} or ground, returning the logic node to its original state. Figure 1.2 shows a collected charge occurring in the drain junction of the p-channel transistor. Originally the node held the value ‘0’. As current flows through the pn-junction of the struck transistor, from the bulk connected to V_{DD} and the drain, the transistor in the on-state (n-channel transistor in Figure 1.2) conducts a current that attempts to balance the current induced by the particle strike. If the collected charge induced by the particle strike is high enough that the on-transistor can not balance the current before the node capacitance is charged, a voltage change at the node will occur. This voltage change lasts until the charge is conducted away by the current feed through the on-transistor.

The electron hole pair track creates a temporary short cut between the substrate and the drain of the transistor in off-state mode. This situation can charge or discharge that stroke node provoking a SET as shown in Figure 1.3a. If the particle hits a transistor that belongs to a memory element such as a latch of flip-flop in Figure 1.3b, the SET is captured in the loop and the effect is a bit-flip of the memory cell, which is classified as SEU. A SET can propagate through the logic and be captured by a flip flop. However, SETs are harmless for the system if theirs effects are masked for especial conditions: logical, latch window, or electrical masking. Logical masking happens when logical conditions of the circuit do not allow the propagation of the error. Latch window masking happens when the voltage pulse is not stored into the memory element due it does not arrive to the input memory element during the rising (or falling) signal of the clock cycle. Electrical masking occurs when the voltage pulse caused by the SET is attenuated during its passage through the logic gates prior to the memory element, so that its effect is masked.

Figure 1.2: Charge Collection Mechanism in inverter gate.

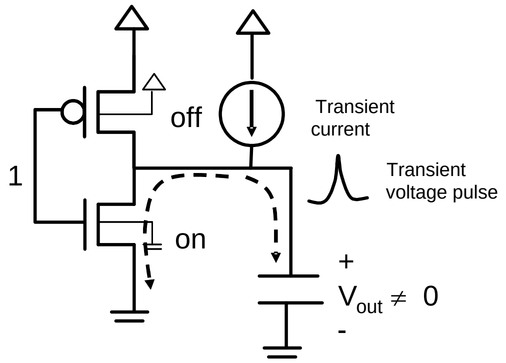
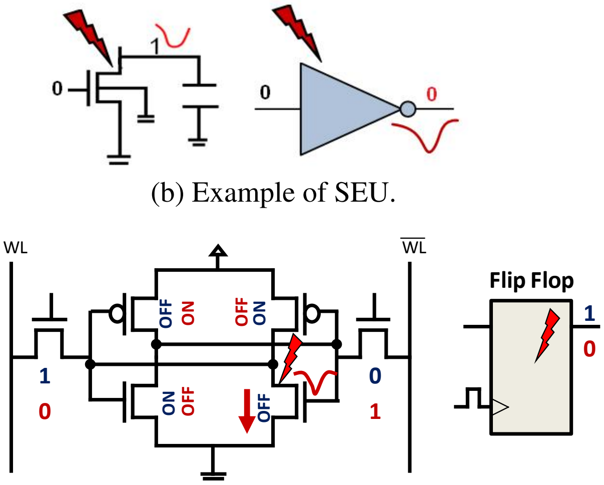


Figure 1.3: Soft errors effects in CMOS devices.

(a) Example of SET.

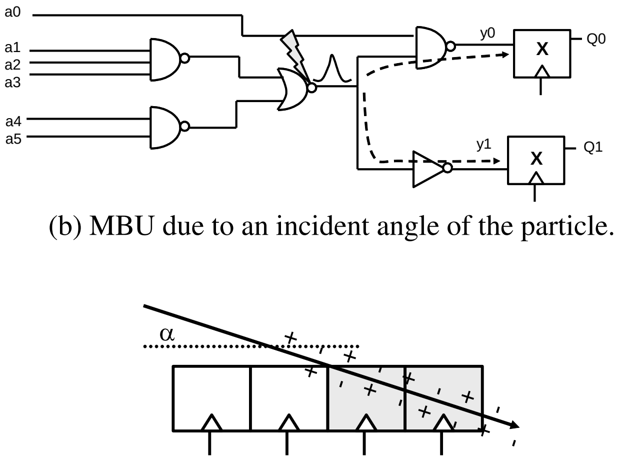


(b) Example of SEU.

Multiple bit upsets (MBU) are also becoming a concern because of the process technology shrinking. MBU can appear due to SETs in nodes with fan-out higher than one as shown in Figure 1.4a; or from double node ionizations due to angle of incidence of the particle due to charge sharing, as shown in Figure 1.4b, which is more common in highly dense memory arrays.

The impact of radiation effects in MOS devices depends on the evolution of digital technology because it depends on the equivalent capacitance of the transistor stroke node, the amount of energy collected by that node and the voltage supply. As we known, the

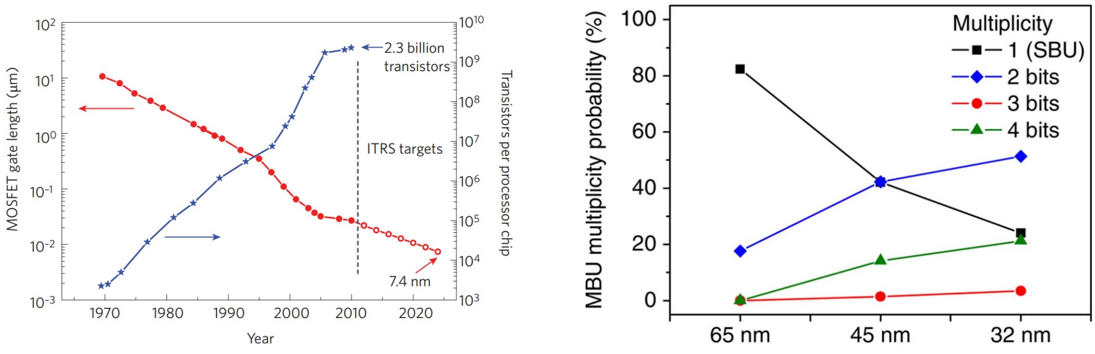
Figure 1.4: Multiple Bit Upset sources.



(a) MBU due to a single SET.

(b) MBU due to an incident angle of the particle.

Figure 1.5: SEU and MBU effects according the technology trend.



(a) MOSFET gate length and density evolution.

(b) Increased possibility of MBUs in new technologies.

(SCHWIERZ, 2010)

(RAINE et al., 2011)

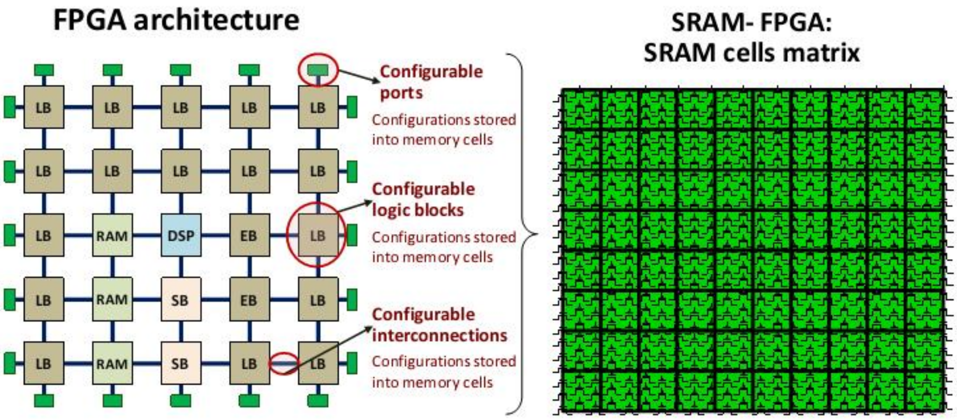
trend to implement more and more complex circuits have led manufacturers to reduce the size of transistors in order to implement more of them in the same area of silicon, that is, increase the density of transistors with low voltage supply (ITRS, 2011).

As shown in Figure 1.5a (SCHWIERZ, 2010), the technology trends to the shrinking of the gate length, and consequently increasing the transistor density of integrated circuits. However, this technology evolution increases the possibility of faults caused by SEUs (RAINE et al., 2011). Memory cells are the ones that mostly scale with technology due to the necessity to reaches high levels of density integration, and consequently, they are high susceptible to soft errors. As shown in (MAIZ et al., 2003), the probability to have MBUs in SRAM cells due a single energized particle is duplicated with the reduction of the gate length from 130 nm to 90 nm, and also the reduction of threshold voltage increases that probability. This trend is confirmed in many publications as (SEIFERT et al., 2008) and (RAINE et al., 2011) which is shown in 1.5b.

1.2 Programmable devices

Field Programmable Gate Array (FPGA) is a device where its hardware functionality can be reconfigurable by the user. FPGAs are very attractive to be used in complex systems due to their high capability of design integration, low NRE costs and configurability (QUINN et al., 2013). Depending on which technology is used to store the configuration of its elements, FPGAs can be reconfigurable. SRAM-based FPGAs are the most popular ones due to its high density and reconfiguration capability. It is composed by a matrix of configurable logic blocks (CLB) where logic can be implemented through look up tables (LUTs) and sequential logic by CLB's flip flops. Also, some devices use special blocks as embedded RAM blocks and DSP blocks. The interconnections between all resources are done through configurable interconnections. The FPGA is configured by loading a bitstream into the configuration memory bits. The Fig. 1.6 shows a general architecture of the FPGA.

Figure 1.6: FPGA architecture as a SRAM matrix memory.


FPGA architecture
LB
RAM
DSP
EB
SB
Configurable ports
Configurations stored into memory cells
Configurable logic blocks
Configurations stored into memory cells
Configurable interconnections
Configurations stored into memory cells
SRAM- FPGA:
SRAM cells matrix

In Flash-based FPGAs, SEUs affects only the user flip-flops because the configuration memory cells are composed of Flash memory cells which have low susceptible to SEUs. On the other hand, SEUs effects in SRAM-based FPGAs are more critical since not only the user flip flops are affected, but also the configuration memory bits where all configuration bits are stored. The modification of any configuration bit may change the functionality of the circuit implemented, and is not corrected until the reload of the original value.

In order to deal with SEU effects in SRAM-based FPGAs, it is necessary to apply some masking technique to guarantee the correct output, and also to implement some repair technique to avoid accumulation of faults.

The most used masking technique is based on spatial redundancy known as triple modular redundancy (TMR). TMR technique consists on the triplication off the original module whose outputs are voted by majority voters to select the correct output value. To majority voter selects the correct output, at least 2 out 3 modules must to be fault-free, hence, TMR only copes with single module faults. The TMR can be implemented in different ways by using coarse grain TMR, or by fine grain that consists in breaking it into small blocks and adding extra voters. Partial TMR (PRATT et al., 2006) proposes the triplication of most critical configuration bits preselected by fault injection. In a SRAM-based FPGA, if a single particle affects two redundancy modules, the majority voter will give a wrong answer. To reduce this effect, the module is partitioned into subgroups inserting more voters between them. Such TMR implementation is known as Partitioned TMR or fine grain TMR and was studied in (KASTENSMIDT et al., 2005; WANG, 2010). XTMR (BRIDGFORD; CARMICHAEL; TSENG, 2008) is proposed by Xilinx (CARMICHAEL, 2006). In the XTMR, all the logic is triplicated and majority voters are used in feedback of the flip-flops to repair the SEUs. In (MANUZZATO et al., 2007), both techniques were compared against a no protected version and a coarse grain TMR to know effectiveness of masking when faults are accumulated. XTMR has a better masking capability of accumulation faults, but also uses more resources than the other TMR implementations. In the same work, it was also shown that coarse grain TMR (with only one majority voter at the outputs) uses fewer resources but has less mitigation capabilities. Diversity TMR (DTMR) consists on the triplication of the same function by different methods, so redundancy modules are not identical but have the same functionality. The use of DTMR in SRAM-based FPGA was presented in (TAMBARA L.; RECH, 2013), where it was shown that DTMR scheme can mask a higher number of accumulated bit-flips compared to coarse grain TMR.

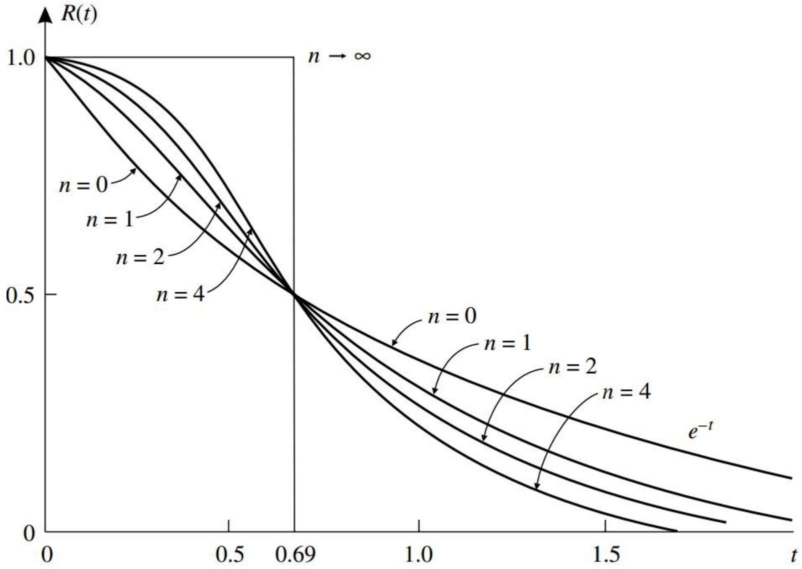
Once any configuration bit is flipped, the fault persists until the rewrite of the correct value is performed. The scrubbing is the process by which memory is rewritten to correct value and prevent the accumulation of SEU in the configuration memory bits without the need of stopping the application. Scrubbing has two main purposes: correct bit-flips and avoid its accumulation into the configuration memory. It is recommended its implementation with a rate of at least 10 times the soft error rate (SER) (ADELL; ALLEN, 2008). As for TMR technique, the scrubbing can be implemented in different ways. Commonly full scrubbing is performed to avoid the accumulation faults in any cell of the configuration memory. Partial scrubbing is also possible in some FPGAs by means of dynamic partial reconfiguration (DTMR) (BOLCHINI; MIELE; SANTAMBROGIO, 2007; HEINER et al., 2009), where it is possible to use less power since only a portion of the configuration memory will be scrubbed. In (AZAMBUJA et al., 2009), the voter of TMR circuit is not only used to mask one fault module output, but also the voter is used to indicates to the scrubber module which module must to be corrected by dynamic partial reconfiguration. After the scrubbing of the faulty module, the system must to be resynchronized. On the other hand, in (BERG et al., 2008) the effectiveness of an internal scrubber was tested against and external one. In (OSTLER et al., 2009), it was demonstrated that the effectiveness of TMR technique is higher when is used with scrubbing.

On the other hand, as it was previously mentioned, the possibility of multiple SEUs increases as the size of the manufacturing technology of transistors decreases. According to the results presented (QUINN et al., 2007), in 65 nm FPGA, more than 50% of events were multiple bit upsets, mainly composed by 2, 3, and 4 bit upsets. Despite efforts to propose new TMR implementations that allow better tolerance of accumulated SEU, these techniques will always be limited to the fact that were designed to mask single faults. Despite XTMR show better results, TMR techniques allow the masking of a single fault, and it cannot cope alone with multi-bit upset on configuration memory bit cells, and consequently burst errors or accumulation of faults between scrubbings (MANUZZATO et al., 2007; OSTLER et al., 2009) may overcome the masking capability of TMR. In (NIKNAHAD; SANDER; BECKER, 2012) a new technique based on quad redundancy is proposed but is aimed to work in logic information and does not considers faults in interconnection lines. In (NAZAR; CARRO, 2012), the approach is to implement a fast detection technique to avoid the increasing fault accumulation, but correction is always performed by frame. In this new scenario, TMR techniques are inefficient to protect the circuits implemented in the FPGA because the possibility of multiples failure is higher, and consequently, the possibility that more than one of the 3 modules fail is also greater. Moreover, the fact that the number of multiple faults is greater in new technology makes scrubbing rate has to be increased also. Finally, the increased density of components makes error detection and correction is slower.

The use of n modular redundancy (nMR) has been explored for the implementation of high reliability computer systems (SHOOMAN, 2002; KIM; SHANBHAG, 2012; SATORI; SLOAN; KUMAR, 2009). nMR is the generic case of TMR, then, instead of 3 modules, nMR uses n identical modules with a voter to select the correct output. In (SATORI; SLOAN; KUMAR, 2009), authors propose a voter able to cope with different number of redundancy modules according to power consumption and module reliability conditions of a framework. In this case, each modules represent a computation system and the number of modules are selectable externally. Since the reliability R is the probability of no failure within a given operating period t, and $p = e^{-\lambda t}$ is the probability of success of a single redundant module where λ is the constant failure rate (failures per million operating hours), the reliability of n parallel redundant systems and an ideal voter (which never fails) is represented by the equation 1.1. This equation is plotted in the Figure 1.7 considering $\lambda = 1$, and for 1 ($n = 0$), 3 ($n = 1$), 5 ($n = 2$) and 9 ($n = 4$) redundancy modules. Notice that nMR is only superior to a no protected module ($n = 0$) until $t = 0.69$ ($0.69/\lambda$).

$$R = \sum_{i=n+1}^{2n+1} \binom{2n+1}{i} p^i (1-p)^{2n+1-i} \tag{1.1}$$

Figure 1.7: nMR reliability with ideal voter according to Equation 1.1 and considering $p = e^{-\lambda t}$ and λ=1.



(SHOOMAN, 2002)

1.3 Thesis Proposal

As explained, in new technologies the probability of MBUs is higher than SEUs, which makes inefficient the traditional mitigation techniques based on TMR. In order to mitigate the effects of multiple faults and its accumulation in the configuration memory of SRAM-based FPGAs, this work proposes the use of multiple modular redundancy (nMR) technique composed by *n* identical modules operating in tandem with an innovative self-adaptive majority voter (SAv) which changes the voting criteria according to the number of fault-free modules.

According to the Figure 1.7, the reliability of a nMR system is higher for higher values of *n* at the beginning of the operation. However, the reliability of a system with *n* redundant modules decreases with the time (because the reliability of each module also decreases), becoming less reliable than systems with fewer redundant modules. Moreover, a system with an static number of redundancies has also an static maximum number of tolerated faulty modules as shown in Table 1.2. For example, in a 7MR system, the maximum number of tolerated faulty modules are 3, since 4 faulty modules can not be voted. Our proposal is focused on the possibility to expand the maximum number of tolerated faulty modules adapting the voting policy, so for example, if at the beginning the system is 7MR and 3 modules fail, the system can continue working as 4MR and then as 3MR tolerating 5 faulty modules instead of the 3 ones of the classical 7MR. Figure 1.8 shows the reliability curves of 7MR, 6MR, 5MR, 4MR and 3MR systems according to Equation 1.1, and considering different voting policies. Hence, the nMR system proposed tolerates more number of faults and, consequently, increases the mean time between system failures.

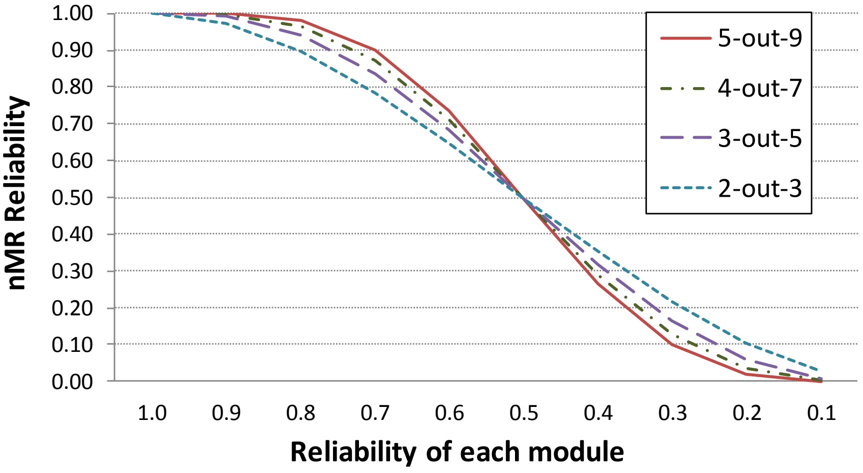
Table 1.2: Possible voting policies in nMR systems.

nMR	Voting policy	Maximum number of failed modules tolerated
9MR	5-out-of-9	4
7MR	4-out-of-7	3
5MR	3-out-of-5	2
3MR	2-out-of-3	1

In the context of the technological trend we pose the following questions: what is the impact of the area overhead when using multiple modular redundancy in the new generations of SRAM-based FPGAs? How much is the penalty in power when using multiple modular redundancy in the new generations of SRAM-based FPGAs? How much the use of multiple modular redundancy in the new generations of SRAM-based FPGAs can increase system reliability? And finally, based on the analysis and studies, is it feasible to use more than three redundant modules in critical systems implemented in SRAM-based FPGAs?

It is well known that the main drawbacks of this technique are area and power overhead. In the first case, it is expected that the area overhead will always be at least n times the number of redundancies. However, the technological trend shows that FPGAs have more and more reconfigurable resources, which are independent of the implemented cir-

Figure 1.8: Reliability of nMR systems according to their voting policies.



cuit. With this, we can consider that in FPGAs the area overhead is a minor drawback for new technologies as new generations of FPGAs may fit thousands of soft-core processors into a single chip. According to the literature, the use of modular redundancy increases considerably the power consumed when applied to ASICs and multiple chip systems. In the case of SRAM-based FPGAs, for any circuit-size that fits in the FPGA, the amount of transistors in every device is constant, it is expected that part of the power consumed is also constant regardless of the amount of resources used in the design (TUAN; LAI, 2003; KUON; ROSE, 2007). This fact must attenuate the power overhead used by the nMR technique getting values less than *n*. In this work we propose a mathematical model which estimates the overhead of power for the use of *n* redundancies.

On the other hand, majority voters are used in TMR designs to select the correct output. In this case, the unique policy of voting is that two out of three inputs must to be correct to select correctly the output. In the case of nMR, this majority voter represent a challenge because more combinations are taken into account for voting and different policies should be implemented depending on the number of fault-free modules exists. In this work, we propose an innovative self-adaptive voter (SaV) to mask multiple upsets in the system, and capable to change the policy of voting according to the modules that remain faultless. For example, for 7MR the correct result is determined by 4-out-7 fault-free modules. When one module fails, we have a 6MR system and the policy remains as the previous case. As the faults continue accumulating, a new module fails and the system is now 5MR. Them, the voting policy changes and the correct result is determined by 3-out-7 fault-free modules. This behavior continues until only two modules remain working correctly. Hence, the proposed system allows to change on the fly the number of redundancies as shown in Figure 1.8, starting with a high number of redundancies and decreasing that number with the time, guarantying always the best possible reliability. In case of an even number of redundant modules may incur in equal and in not electable majority situation (uncommon situation), the voter considers as a non-correctable situation and a reconfiguration of the system is needed.

The validation of the proposal is made by performing terrestrial radiation experiments (in ISIS laboratory facilities) and by fault injection campaigns in two study case circuits. The fault injection consists in the generation of bitflips in the configuration memory in a similar way to the energized particles affects the FPGA. To do this, we developed a fault injector platform able to flip multiple and accumulative faults in the configuration memory bits of the SRAM-based FPGA using a distribution collected in real neutron radiation experiments.

This Thesis is composed as follows.

- Chapter 2: Dependability in SRAM-based FPGAs: It is presented the taxonomy used in this work, dependability concepts, an overview of state-of-the-art SRAM-based FPGAs and its radiation effects, and the measurements methods of such effects.

- Chapter 3: Mitigation techniques for SRAM-based FPGAs: A brief description of the classical and state-of- the-art FPGAs methods to mask and correct faults, and methodologies for testing, is presented.

- Chapter 4: Proposed Self-adaptive n-Modular Redundancy technique: Presents the system proposed and depicts the architecture, functionality and scalability of the novel Self-adaptive voter.

- Chapter 5:Proposed fault injector platform: It is presented a novel fault injection platform that allows the analysis of accumulation faults effects.

- Chapter 6: Power analysis in nMR systems in SRAM-based FPGAs: It is proposed a model to prevent the power overhead of an nMR system and analyze the real effects when the proposal is implemented in a SRAM-based FPGA.

- Chapter 7: Reliability analysis of nMR systems in SRAM-based FPGAs: Some circuits under test were implemented in 7MR and 6MR modes, and they were irradiated by neutrons and tested by fault injection experiments. Results are presented in this chapter.

- Chapter 8: Conclusions and discussions: We presents the conclusions, discussions and future works of the Thesis, as the list of publications during the Thesis process.

2 DEPENDABILITY IN SRAM-BASED FPGAS

This chapter is composed by three subsections. In the first one, the taxonomy of dependability and measurements of reliability are presented. After that, an overview of SRAM-based FPGA including the architecture and main features of one commercial device is detailed. Finally, it is depicted a brief description of radiation effects in SRAM-based FPGAs.

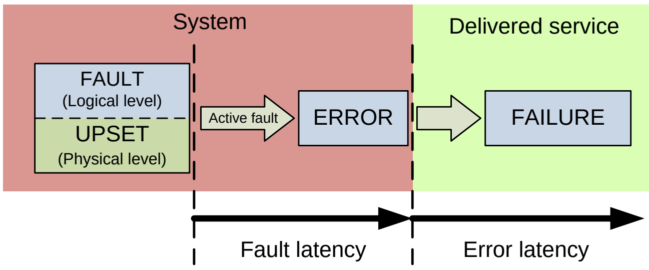
2.1 Dependability concepts

2.1.1 Defect, Fault, error

Defect (or upset), fault, error and failure are defined considering system concept. A system is an entity that interacts with other entities, such as other systems, hardware, software and humans. From a structural viewpoint, a system is composed of a set of components bound together in order to interact, where each component is another system. The component is considered atomic when its internal structure cannot be discerned or can be ignored. Every system has a functional specification that describes the function what the system is intended to do, and its service delivered is perceived by its user(s) which is another system. A failure (service failure) is an event that occurs when the delivered service deviates from correct service. Then, failure is defined as a system malfunction, or in other words, when the delivered service deviates from the correct one. A service fails either because it does not comply with the functional specification, or because this specification did not adequately describe the system function. Failure is caused by the deviation in one of the system's sequence of states. Such deviation, named error, may compromise a system service, thus leading to a service failure. It is important to note that an error not always leads to a failure. Furthermore, an error may be caused by a fault that describes a deviation from the expected behavior of logic. When a fault leads an error, such fault is defined as active, and it is defines as dormant when it does not. Faults are usually classified in transients, intermittent or permanents. A fault is defined as a logic level abstraction of a physical defect or upset. Finally, defect or upset is defined as an unintended difference between the implemented hardware and its intended function. Errors can be caused by a defective manufacture process, or transient upset that happen during some perturbation of the environment. Figure 2.1 shows the cause-effect relationship between fault, error and failure. The period of time from the fault event to its manifestation (if happens) as an error event is known as fault latency, and error latency is the period of time since error event until its manifestation (if happens) as a failure.

Since upsets depend on manufacturing process and/or environment conditions, an upset can always happen and may be propagated to generate a failure. According to

Figure 2.1: Fault, error and failure.



(AVIZIENIS et al., 2004), dependability can be defined as the ability of a system to avoid service failures that are more frequent or more severe than is acceptable. Dependability is an integrating concept that encompasses attributes (PRADHAN, 1996). Two main attributes for this work are:

- Availability ($A(t)$): probability that a system is operating correctly and is available to perform its functions at the instant of time, t.
- Reliability ($R(t)$): conditional probability that the component operates correctly throughout the time interval (t_0, t_1), given that it was operating correctly at the time t_0. In other words, reliability is the probability of no failure within a given operating period (SHOOMAN, 2002).

Dependability can be achieved by means of the use of fault tolerant that aims at failure avoidance. Fault tolerant systems are systems that can deliver its service according to the functional specification despite the presence of faults. Fault tolerance techniques are carried out via error detection and system recovery. In the first step, error identification can be performed during normal service delivery (concurrent detection) or when is suspended (preemptive detection). There are two strategies for system recovery: eliminating the error from the system state (error handling), or preventing faults are activated again (fault handling). In error handling, redundancy can be used to mask the error. Such masking will conceal a possibly progressive and eventually fatal loss of protective redundancy. So, practical implementations of masking generally involve error detection (and possibly fault handling), leading to masking and recovery. On the other hand, fault handing can be implemented by isolation or reconfiguration: isolation consists in excluding the faulty components from further participation in service delivery, and reconfiguration consists in the use of spare components or the reassignment of tasks among non-failed components.

2.1.2 Reliability and availability measurements

In critical missions, a minimum level of reliability is required to achieve. Such levels can be quantified through parameters whose indicate how good the system is and how frequently it goes down (PRADHAN, 1996; SHOOMAN, 2002). In this work we use the following nomenclatures:

- $R(t)$: Reliability function. It is equal to the probability of success $P_s(t)$ in a time t.
- $P_f(t)$: Probability of failure in a time t.
- $z(t)$: Hazard function Fault rate function.

- λ: constant fault rate, expected number of failures of a type of system per a given time period.
- FIT: Failures in time unit. 1 FIT=one error per 10^9 device hours.
- MTTF: Mean time to failure is the expected time that a system will operate before the first failure occurs.
- MTTR: Mean time to repair is the average time to repair a system.
- MTBF: Mean time between failure is the average time between failure of a system.

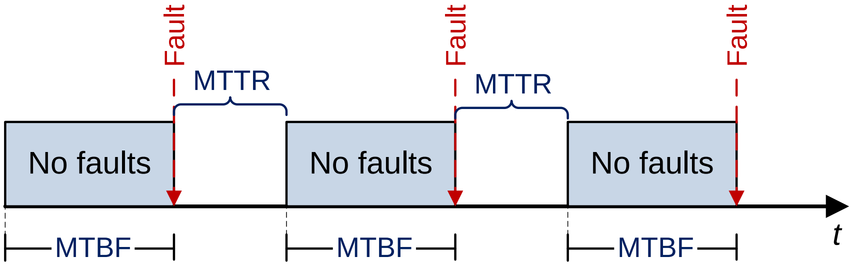
All these functions and parameters can be related between them. For example, Equation 2.1 shows that reliability is a exponential function of the failure rate λ, in other words the system will decrease its reliability with the time in an exponential factor of λ.

$$R(t) = e^{-\lambda t} \tag{2.1}$$

On the other hand, the availability is related to the time that the system is available to be used. In a system where failures can be repaired, the system behavior follows the sequence presented in Figure 2.2 (STRAKA; KOTASEK, 2009): first the system works correctly until a fault appears (MTBF), then it is necessary to correct the fault (MTTR) to still working until the following fault. Availability function is defined by:

$$Availability = \frac{MTBF}{MTBF + MTTR} \tag{2.2}$$

Figure 2.2: MTBF and MTTR sequence.



When it is not possible to repair the fault, it is usual to use the parameter MTTF instead of MTBF to indicate the expected time to occur a fault. MTTF is defined by the Equation 2.3 and is related to the fault rate as shown in Equation 2.5.

$$MTTF = \int_{0}^{\infty} R(t)dt \tag{2.3}$$

$$MTTF = \frac{1}{\lambda} \tag{2.4}$$

As a practical example, assume that there are 200 systems being testing, and after 1000 hours, 4 faulty systems are detected. Then, the probability of failure is:

$$P_f(1000) = \frac{4}{200} = 0.02$$

Consequently, the reliability of the system for $t = 1000$ is:

$$R(1000) = 1 - P_f(1000) = 0.98$$

For the same case, the failure probability in failures per million operating hours for $t = 1000$is:

$$z(1000) = \frac{4 failures}{200x1000} = 20$$

Considering the fault rate as a constant function in a time interval, the fault rate λ is calculated as following.

$$\lambda = \frac{4 failures}{200} \cdot \frac{1}{1000 hours} = 2x10^{-5} hours^{-1}$$

The reliability function can be defined using the constant fault rate presented in Equation 2.1, then, the reliability function of the example in time (hours in this case) is defined by:

$$R(t) = e^{2x10^{-5}t}$$

The reliability function after 1000 hours of testing is defined by:

$$R(1000) = e^{2x10^{-5}x1000} = 0.98$$

This result is consistent with our previous result.

MTTF is also used as parameter of reliability. In our example, the MTTF can be estimated using the Equation 2.5 as follows

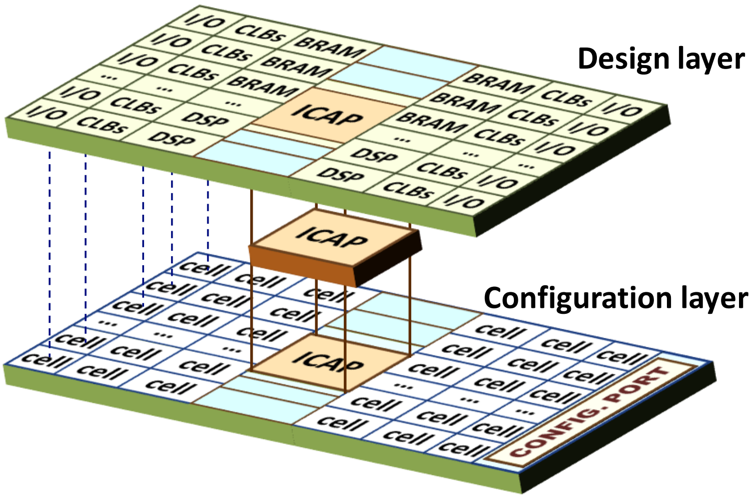
$$MTTF = \frac{1}{\lambda} = \frac{1}{2x10^{-5} hours^{-1}} = 50000h \tag{2.5}$$

This means that according to the tests, it is expected one fault each 50000 hours. In the case of digital devices, industry uses commonly the Failure In Time (FIT) as unit to measure the reliability of the system. Since FIT represents 10^9 device hours, the MTTF can be expressed in terms of FITs, then, in our example the MTTF is 5.0x10^{-5} FIT

2.2 SRAM-based FPGA overview

From the point of view of the design, we can split the SRAM-based FPGA into design and configuration layer as depicted in Figure 2.3. This approach is close to presented in (HERRERA-ALZU; LÓPEZ-VALLEJO, 2013). In the design layer, the user implements the functional blocks through some HDL language. All configurations are stored into memory cells through some configuration port. Each memory cell is known as configuration bit and a group of them is known as configuration bitsream

Figure 2.3: Abstraction layers of a SRAM-based FPGA.

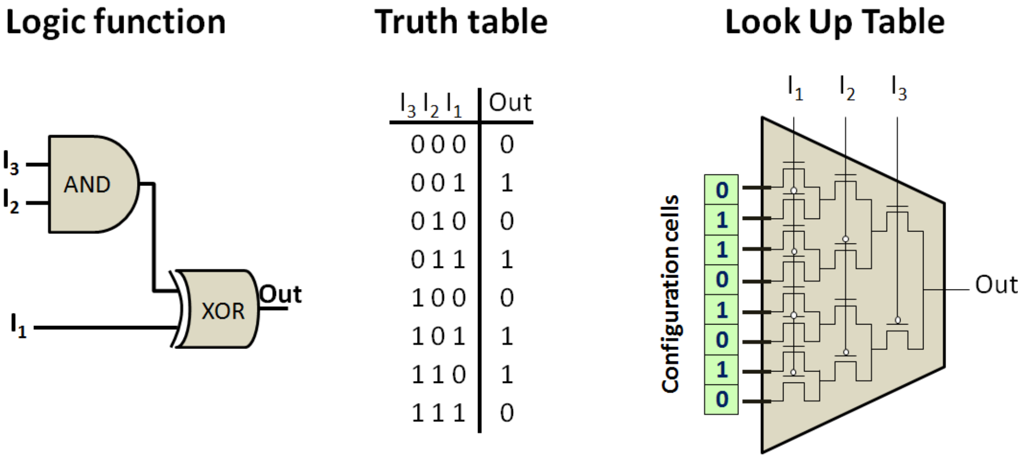


2.2.1 Design layer

The SRAM-based FPGA consists of logic blocks, I/O blocks, special blocks and routing resources. All of them are configured by the configuration bitstream stored into the SRAM cells and loaded during the power on of the device.

Logic blocks are capable to implement a combinational and sequential logic function which is defined inside the FPGA configuration memory. Commonly a logic block contains a Look-Up Table (LUT) to implement combinational functions, flip-flops and multiplexers for implementing different signal forwarding strategies. LUTs are used to implement the truth table of combinational functions. Internally, they work as a multiplexer where the selectors are the inputs of the function and the possible outputs are configured into the SRAM cells. The number of inputs depends on the type of LUT, for example, Figure 2.4 shows the implementation of a simple logic function $Out = (I_1 \cdot I_2) \oplus I_3$ implemented by a 3-input LUT. To implement more complex combinational logic, manufacturers offer higher input LUTs. For example, Virtex-5, Virtex-6 and Virtex-7 FPGAs use 6-LUT (XILINX, 2012a,b, 2014b).

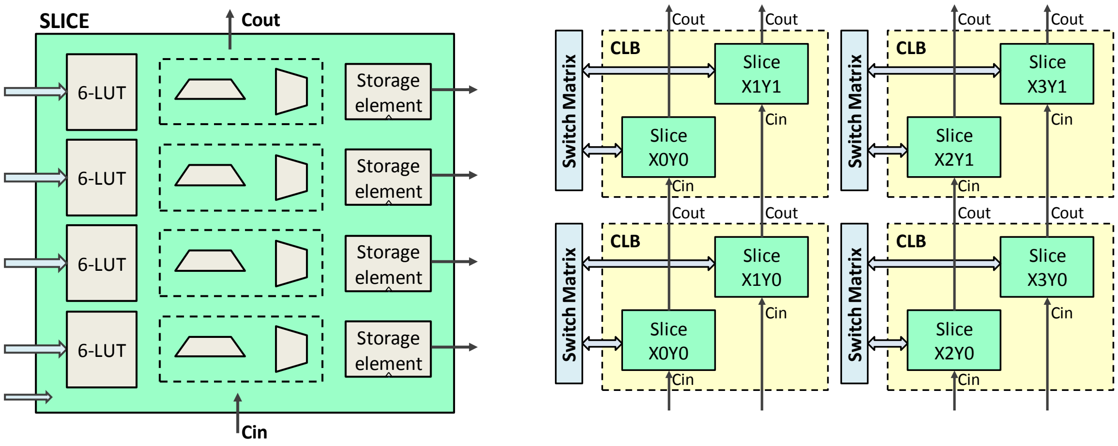
Figure 2.4: Example of implementation of a combinational function in a 3-LUT.



$I_3\,I_2\,I_1$	Out
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	0

In the case of XILINX FPGAs, the logic block is known as configuration logic block (CLB) and is divided into slices which combines LUTs, storage elements, multiplexers and carry logic to provide logic arithmetic, storing data (i.e. ROM functions, distributed RAM, FF, latch) and shifting data with 32-bit shift registers (32-SRL). The slices of a CLB are interconnected through a switch matrix as shown in Figure 2.5b, and use a coordinate system (X, Y) to identify the position of each slice within the FPGA. The number of slices per CLB and internal elements depends on the FPGA family. For example, in Virtex-5 family each CLB is composed by 2 slices of four 6-LUTs, four storage elements, wide-function multiplexers, and carry logic (XILINX, 2012a) as shown in Figure 2.5a. Virtex 6 and 7 family uses eight storage elements instead of four (XILINX, 2012b, 2014b).

Figure 2.5: Logic Blocks in Virtex FPGAs.


(a) Diagram of Virtex-5 Slice.

(b) Example of CLBs interconnection.

The clocking of the sequential elements of the FPGA is performed by global and local clocks signals. These signals divide the FPGA into clock regions and are controlled by clock buffer primitives as IBUFG and IBUFGDS (XILINX, 2014c). In the case of I/O resources, in modern FPGAs it is possible to configure some features as the level voltages, directionality, and delays. CLBs, I/O blocks and special blocks are interconnected through wiring segments (long and local lines) that can be connected or disconnected by programmable interconnection points (PIPs) (VIOLANTE, 2007; TAHOORI; MITRA, 2003). The basic PIP structure consists of a pass transistor controlled by a configuration memory bit. There are several types of PIPs:

- Cross-point PIPs: connect wire segments located in disjoint planes (one in the horizontal plane and one in the vertical plane).
- Break-point PIPs: connect wire segments in the same plane.
- Compound PIPs: consist of a combination of n cross-point PIPs and m break-point PIPs, each controlled separately by groups of configuration bits.
- Decoded Multiplexer PIPs: groups of cross-point PIPs sharing common output wire segments controlled by configuration memory bits.
- Non-de-coded MUX PIPs: wire segments controlled by configuration bits.

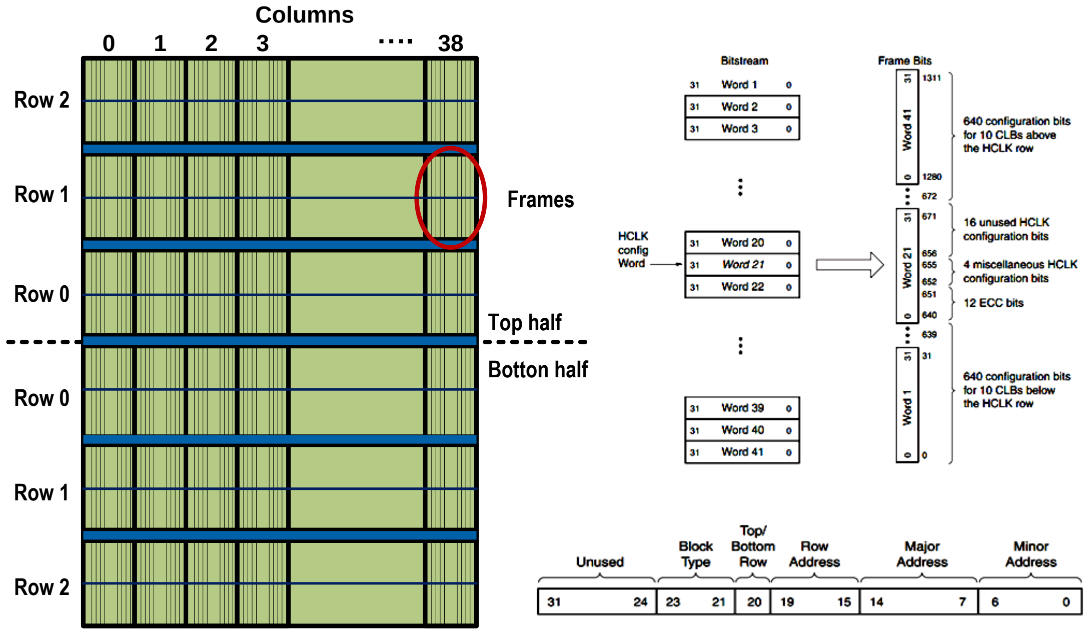
FPGAs vendors usually offer in the device some hardware blocks (primitives) to facilitate the design of complex circuits optimizing resources. For example, blocks of internal configurable RAM are presented in Virtex FPGAs and are known as BRAMs. In these blocks, the size word and deep are configurable, as the way to read and write, and also the possibility to protect the data by means of error correction codes (ECC). DSP blocks are offered to implement arithmetic operations which have better features that when implemented by CLBs. Clock features are also possible to be configured by using digital clock managers. Some FPGAs have complex hardwired microprocessors that enable the development of systems on chip (SoC). In latest Virtex FPGAs there are primitives able to interact with the configuration bits without the need to use the classical external configuration ports. This is helpful because it allows the control (reading, writing and analysis) of the configuration bits from the user logic implemented in the design layer. The primitives of this type are presented in the following subsection.

2.2.2 Configuration layer

The FPGA is configured by loading the application-specific configuration data (known as bitstream) into the internal configuration memory at the time of power on. In Virtex FPGAs, the configuration bitstream can be loaded by using serial (Master/Slave, Serial Peripheral Interface - SPI) or parallel (SelectMAP, Byte Peripheral Interface - BPI) modes (XILINX, 2012c).

The configuration memory is composed by frames that are the smallest addressable segments of the memory. The size of the family device, for example Virtex-5 frames are composed by 41 words of 32-bits (this is 1311 bits), and Virtex-6 frames are composed by 81 words of 32-bits (RAO et al., 2014). All frames and commands form the bitstream. In latest FPGAs, the configuration of a portion of the FPGA can do it. This is known as dynamic partial reconfiguration DPR) due such partial configuration does not stop the application (XILINX, 2010b). Figure 2.6 depicts the relationship between the floorplan of Virtex-5 FPGA and the structure of its configuration bitstream.

Figure 2.6: Memory configuration and frames in Virtex FPGAs.

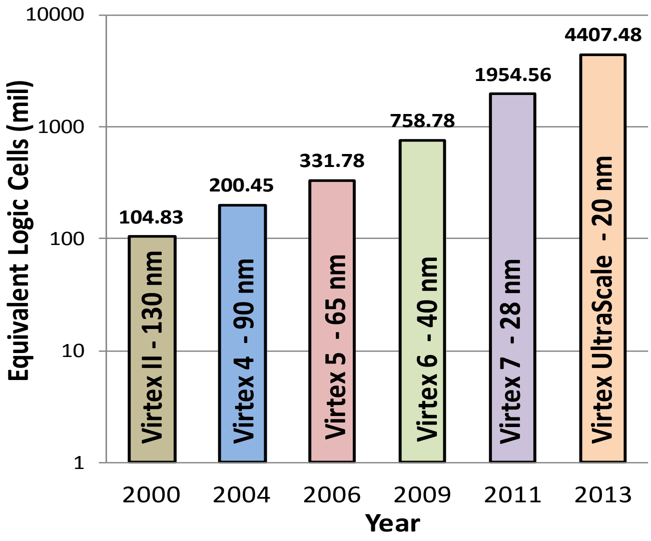


Virtex FPGAs have special primitives to access and analyze the configuration bits. The Internal Configuration Access Port (ICAP) can be accessed from the design level to read and write the configuration memory. Its operation is similar to the SelectMAP port (XILINX, 2012c), and its data bus width is selectable among 8, 16 or 32 bits. For Virtex-4, Virtex-5 and Virtex-6, the ICAP can run up to a clock frequency of 100 MHz (XILINX, 2010c). ICAP can be used to implement DPR. The control of the ICAP is performed by an embedded processor or also by an FSM. Classical Xilinx flow uses MicroBlaze soft processor with a dedicated IP to control the ICAP. However, the control of ICAP can be performed by simpler IP, as presented in (TARRILLO et al., 2014). ICAP reads and writes static bit configurations, and not dynamic as user Flip Flops, data BRAMs and SRLs.

Being aware of the susceptibility to faults of configuration bits, manufacturers provide error detection codes in the configuration memory (CHAPMAN, 2010a). Each frame is protected by ECC that can detect the position of a flipped bit, and detect up to bitflips errors in the frame. It uses SECDED (Hamming code) parity values based on the frame data generated by the synthesis tool. Additionally, a 32-bit CRC is used to detect any change in as many as 2^{32} bits configuration memory. With them, it is possible to detect whether there is data corruption in memory, but it is not possible to know the position or positions of faults. Both ECC and CRC codes can be accessed by the FRAME_ECC_VIRTEXx primitive (x depends on the Virtex device).

Following the trend technology of semiconductor devices, SRAM-based FPGAs evolution increases their resources each generation. In the case of Xilinx FPGAs, the largest family is named Virtex. Figure 2.7 depicts the number of largest product of each product family since the year 2000. In the same figure, the node technologies are also shown.

Figure 2.7: Xilinx FPGAs evolution since the year 2000.



2.3 Radiation Effects on SRAM-based FPGAs

2.3.1 Susceptibility parameters

When a charged particle (as protons or heavy ions) hits a device, part of its charge lets in the device. This is known as linear energy transfer (LET) and expresses the energy loss per unit length (dE/dx) of a particle and is a function of the mass and energy of the particle as well as the target material density. The units of LET are commonly expressed as $MeVcm^2/g$ (BARNABY, 2006). Radiation experiments with charged particles commonly relates the relation between cross-section and LET, which also depends on the incidence angle.

Any circuit implemented in an SRAM-based FPGA designed to operate in radiation environments requires to have minimum values of reliability parameters, which are classified in static and dynamic parameters. The first group is related to the reliability of the device. SEUs are caused by radiation, and the SEU rate parameter gives the information about the frequency at which one SEU occurs in specific radiation conditions. In a SRAM-based FPGA, the SEU rate can be calculated using the number of bitflips observed during a period of time, as presented in Equation 2.6.

$$SEU\ rate = \frac{\#bitflips}{time} \tag{2.6}$$

Static cross-section (σ) helps designers to quantify the sensitivity of the FPGA technology to a specific radiation source (VIOLANTE et al., 2007). Device cross-section (σ_{device}) is related to the minimum susceptible area of the device to the effects caused by an specific radiation source, and it is specified in cm^2. σ_{device} (also known as static cross-section) is calculated considering the irradiating conditions and the number of radiation-induced faults. In the case of proton and heavy-ion static cross-sections, σ_{device} is calculated using the Equation 2.7, where θ is the incident angle of the particles flux ϕ in $neutrons/cm^2 \cdot s$ (QUINN et al., 2009; BERG et al., 2012; QUINN et al., 2005). Usually, vendors prefer to use the bit cross-section (σ_{bit}), which is determined by dividing the device cross-section by the number of configuration bits in the device, as presented in Equation 2.8.

$$\sigma_{device} = \frac{\#events}{fluence \times cos(\theta)} = \frac{\#events}{\phi \times time \times cos(\theta)} \tag{2.7}$$

$$\sigma_{bit} = \frac{\sigma_{device}}{bitstream_{size}} \tag{2.8}$$

On the other hand, dynamic cross section $\sigma_{dynamic}$ is defined as the ratio between the number of SEUs that produces a wrong output (failure) in the design, and the fluence of hitting particles to the device. Then, dynamic cross section quantifies the sensitivity of the implemented circuit to any specific radiation source (VIOLANTE, 2007), and can be calculated experimentally by means of the Equation 2.9.

$$\sigma_{dynamic} = \frac{\#errors}{\phi \cdot time} \tag{2.9}$$

The rate at which soft errors occurs is called the Soft Error Rate (SER). From the system point of view, the SER can be consider as the failure rate, and can be expressed in FITs that is the number of faults in 10^9 operation hours by device tested. Notice that SER is proportional to both device size and flux as shown in Equation 2.10 (BAUMANN, 2005;

QUINN; GRAHAM, 2005). Experimentally, SER can be also calculated dividing the number of observed errors during an experiment time interval, by such time, as presented in Equation 2.11.

$$SER = flux \times \sigma_{dynamic} \tag{2.10}$$

$$SER = \frac{\#errors}{time} \tag{2.11}$$

Finally, knowing the cross-section of a device, the results obtained from accelerated neutrons can be taken to estimate the SER in place using the Equation 2.12. We highlight that to use the Equation 2.12, the distribution of the particles must be similar to the target place. For example, the neutron distribution of the neutron source from ISIS facilities (situated at the Rutherford Appleton Laboratory) remains the neutron flux at see level, as shown in (VIOLANTE et al., 2007)

$$SER = Cross - section \times \#Particles \tag{2.12}$$

2.3.2 Radiation effects in state-of-the-art FPGAs

FPGAs are susceptible to total ionizing dose (TID) and single-event effects (SEEs). In the case of TID, the power supply current of the FPGAs increases due the ionizing radiation absorption. In (MACQUEEN et al., 1999), several 250 nm FPGA devices were irradiated wit γ rays from Co-90 source, detecting the increase of the power supply current since approximately 16 krad of accumulated dose. Other experimental reports can be found in (SMITH; MOSTERT, 2007) and (TARRILLO et al., 2011). However, the shrinking of transistors allows a less ionization of the transistor, and consequently, FPGAs built using new technologies are more robust against TID effects.

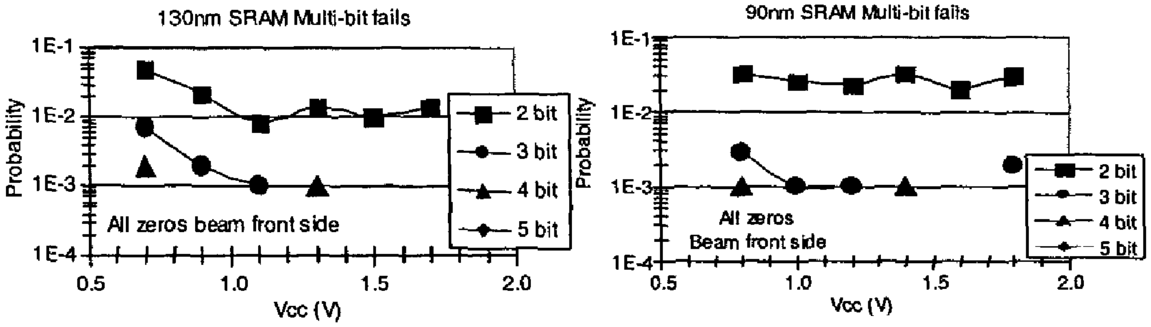
In contrast, the same shrinking and the reduction of threshold voltage make them mores susceptible to SEE caused by protons, heavy ions and energized neutron particles. Single-event latchup (SEL) is a hard error and results in a high operating current, above device specifications, that must be corrected by a power reset. Single-event latchup (SEL) is a hard error caused by the disruption of electrical systems that turns on the complimentary metal-oxide-semiconductor (CMOS) parasitic bipolar transistors between well and substrate. This effect causes high operating currents (above device specifications) that must be corrected by a power reset, if not, SEL can cause a permanent damage (BAUMANN, 2005). However, recent FPGAs provide high immunity to latchup effects (QUINN et al., 2009)

Soft errors do not damage the device, but can cause serious malfunctions in the application. Radiation-induced faults from SEUs (also called 'upsets') cause the flip of one configuration bit (single-bit upset or SBU), or more than one configuration bit (multi-bit upsets or MBUs). SEU and MBU are the main concerts for latest SRAM-based technology.

The SEU effects in SRAM-based technologies are constantly studied (MAIZ et al., 2003; SEIFERT et al., 2008; RAINE et al., 2011; IBE et al., 2010). As demonstrated in the literature, the possibility of MBUs increases with the reduction of the technology nodes. For example, according to (MAIZ et al., 2003) the reduction from 130 nm to 90 nm duplicates the probability of 2 bitflips, as shown in Figure 2.8. In (IBE et al., 2010), the SEUs effects produced by neutron flux (from 0 to 200 MeV) in SRAM technologies from 250 nm to 22 nm were predicted using the Monte-Carlo simulator. As shown in Figure

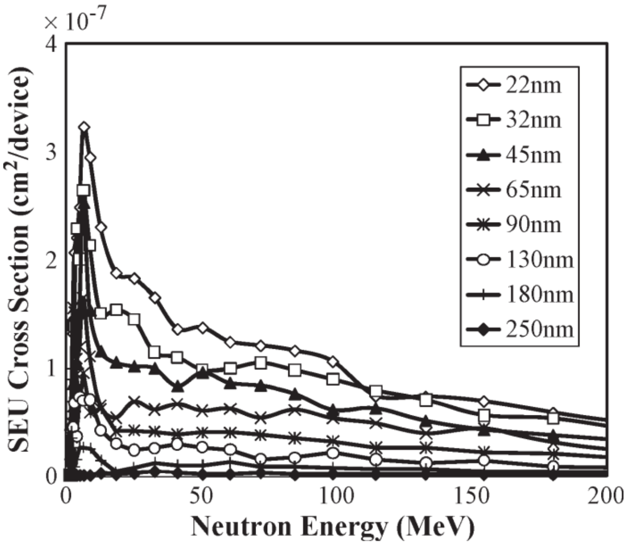
2.9, the cross-section is higher for technologies based on smaller transistors (IBE et al., 2010), this is, the susceptibility to faults increases as the size transistors reduction.

Figure 2.8: Probability of 2 bit-flips in 90nm is twice of the 130 nm.



(MAIZ et al., 2003)

Figure 2.9: Changes in a SEU cross section in SRAM with scaling.

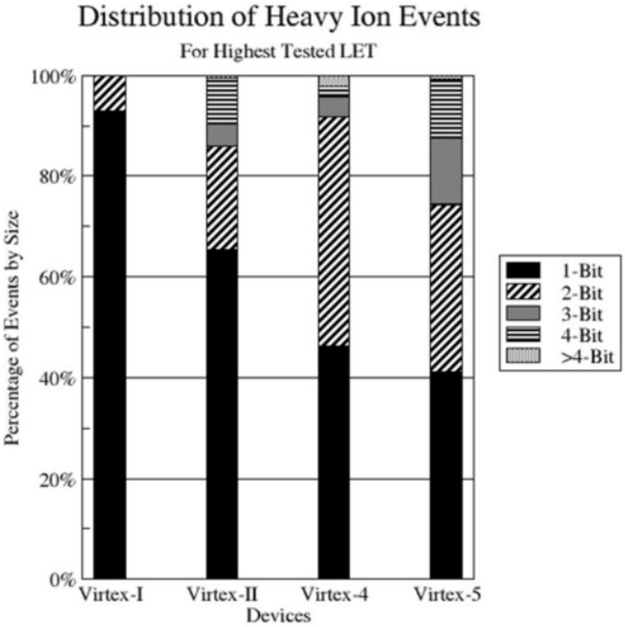


(IBE et al., 2010)

The effects of the SEUs in FPGAs depend on the location where the upset happened. For example, if a SEU hit the device control, it is possible to provoke a device functional error known as single-event functional interrupt or SEFI. To recover the control of the device, it is necessary to reconfigure completely the device, or even do the power on. Fortunately, SEFI error rates are very low (QUINN et al., 2009). In the case of a SEU affects some user flip-flop, its correct value can be reloaded during the normal application (the effect is masked by the application) or can be corrected by some mitigation technique at design level. Even more, the possibility of a SEU in flip flop of user is very low due to the fact that compared with the memory of configuration, the susceptibility is very much minor and in addition there are very much fewer flip-flops than configuration memory cells. For example, the XC5VLX50T has approximately 0.03 Mb of flip-flops and more than 13 Mb of configuration bits. For these reason, SEUs effects in flip-flops user are normally omitted (CHAPMAN, 2010a). Furthermore, the storage cells of the blocks RAM are more susceptible than the configuration memory elements, though the effects are less critical SEUs. For example, in Virtex-5 FPGA there are around five times more bits in the configuration memory than in the BRAM blocks. Additionally, commonly BRAMS are protected with ECC codes to mask its effects in the applications.

On the other hand, the probability of a single charged particle causes multiple bit upsets (MBU) in new FPGAs was reported in many works (QUINN et al., 2005, 2007; BAUMANN, 2005). Figure 2.10 shows some published data (QUINN et al., 2005). In (QUINN et al., 2009), the effects of heavy ions radiation in Virtex-4 and Virtex-5 FPGAs to different LET values were presented. Figure 2.11 shows that by a similar LET, the percentage of MBUs is about 10% greater in Virtex-5 (65 nm) compared to Virtex-4 (90 nm). In neutron radiation experiments, the LET is not considered since neutron does not have any energy to transfer to the device.

Figure 2.10: Probability of 1, 2, 3, 4, and more upset bits in configuration memory of Virtex, Virtex-II, Virtex-4 and Virtex-5 FPGAs.

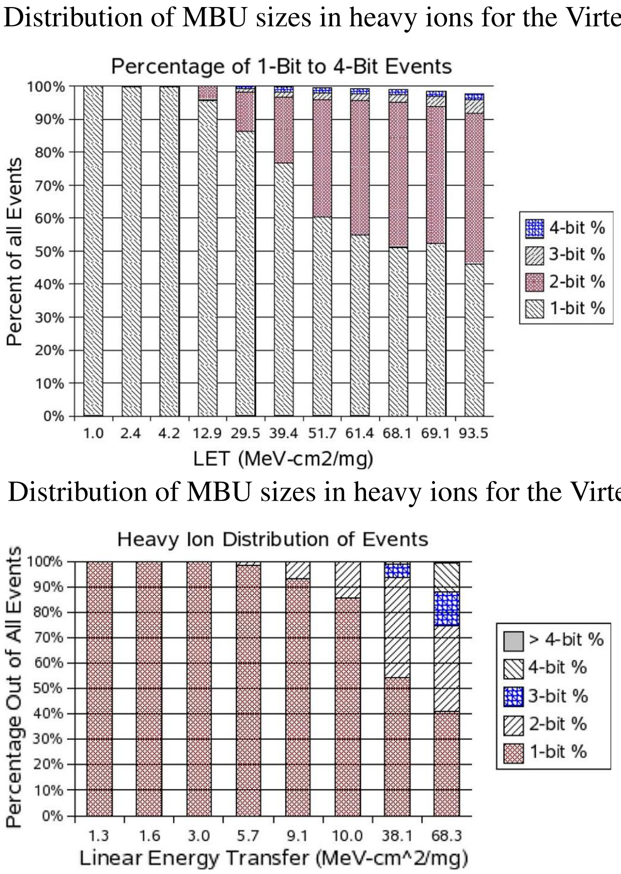


(QUINN et al., 2005)

Xilinx publishes the reliability of their devices 4 times a year in the report ‘Device reliability report’ (XILINX, 2014a) based on Rosetta experiments and beam radiation experiments. Rosetta experiments aims to show the atmospheric neutron effects by means of continuing real-time atmospheric experiments of a large Xilinx FPGAs fabricated (XILINX, 2011a). Reliability results are presented in neutron cross-section and soft error rates terms. Neutron cross-sections are determined by experiments performed at the Los Alamos Neutron Science Center (LANSCE), and soft error rates (in FIT/Mb) are determined from real time measurements in various locations and altitudes and corrected for New York city.

Reliability results for configuration bits presented in Table 2.1 are taken from the last report of 2013 (XILINX, 2014a). As shown, the resilience of memory cells is improved

Figure 2.11: MBU distribution for heavy ions experiments in Virtex-4 (90nm) and Virtex-5 (65nm).



(a) Distribution of MBU sizes in heavy ions for the Virtex-4.

(b) Distribution of MBU sizes in heavy ions for the Virtex-5.

(QUINN et al., 2009)

in almost each FPGA generation since the bit cross-section is reduced and the FIT/Mb is also reduced. However, it is necessary to highlight that although manufacturing efforts achieve the reduction of the bit cross-section, the number of configurable bits increases with each technology (BRIDGFORD; CARMICHAEL; TSENG, 2007; XILINX, 2007, 2009a, 2012c, 2013a,b) as shown in Figure 2.12. Hence, although the FIT/MB is reducing with technology, the amount of bits present in the FPGA is increasing drastically with in overall is making the number of failures to increase. Figure 2.13 shows the device cross-section of the largest component of each family, where mainly in the last two generations, the cross-section increases considerably.

Summarizing, we consider the radiation effects on configuration memory as the major concern due any configuration bitflip may potentially cause a malfunction of the implemented design and only can be recovered after the rewrite of the upset bitflips. If these actions are not performed, the bitflips not only remain but in addition will be accumulated, increasing considerably the possibility of failure Moreover, sometimes the repair of the upset bits is not enough to restore the circuit operation. These errors are known as persistent and further the correction of the bit, the perform of some type of resynchronization as a reset (MORGAN et al., 2005).

Table 2.1: Xilinx reliability report accessed in the fourth quarter of 2013.

Technology Node (nm)	Product Family	σ per bit (LANSCE)	FIT/Mb (Rosetta experiment)
250	Virtex	9.9E-15	160
180	Virtex-E	1.12E-14	181
150	Virtex-II	2.56E-14	405
130	Virtex-II Pro	2.74E-14	437
90	Virtex-4	1.55E-14	263
65	Virtex-5	6.70E-15	165
40	Virtex-6	1.26E-14	99
28	7 Series FPGA	6.99E-15	85

(XILINX, 2014a)

Figure 2.12: Amount of configuration bits in largest components of Virtex FPGAs.

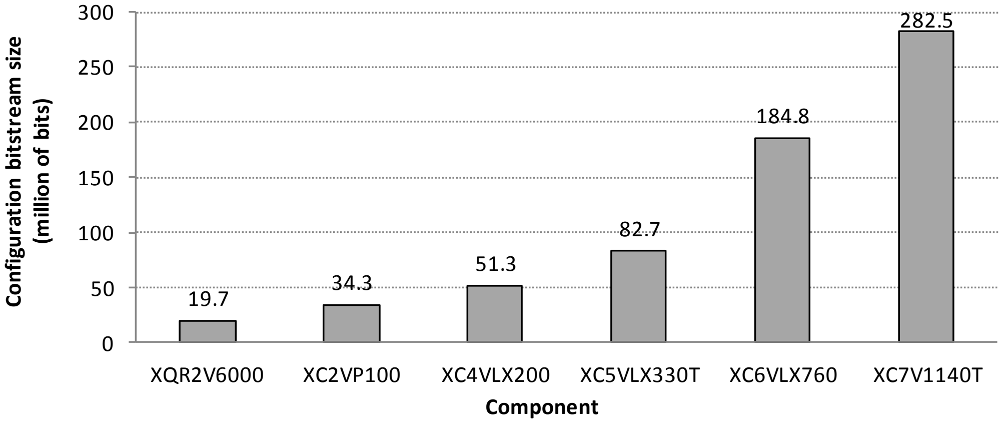
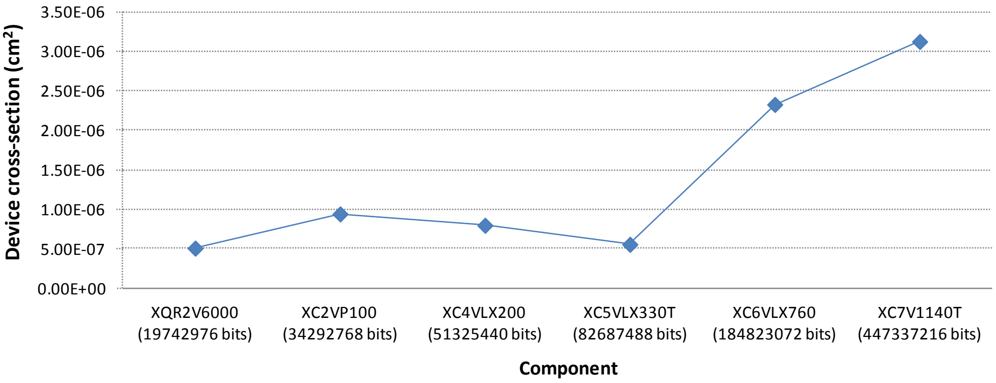


Figure 2.13: Device cross-section in largest components of Virtex FPGAs based on (XILINX, 2014a).



2.3.3 Example of reliability measurements

Radiation experiments are performed to characterize the reliability parameters of a SRAM-based FPGA. As example, Table 2.2 shows the information of a neutron radiation experiments where 5 runs of a design implemented in an FPGA with 14043648 configuration bits were performed. Each run ends when a functional fault of the target design is detected.

Table 2.2: Example of neutron radiation experiment.

	Time (min)	Neutron flux	Bit-flips (#)
Run 1	22	4.11 E04	70
Run 2	25	3.69 E04	76
Run 3	20	4.11 E04	60
Run 4	32	4.10 E04	114
Run 5	27	3.58 E04	78

With the information of Table 2.2, we can calculate the SEU rate, static device and bit cross-section, the dynamic cross-section, and the soft error rate (SER), using the equations 2.6, 2.7, 2.8, 2.9, and 2.11 respectively. Results are presented in as presented in Table 2.3. Notice that since many runs were performed, results are presented considering the average of the results as the confidence interval of 95%. The confidence interval indicates the precision of the results if the experiment is repeated, and it is calculated considering the standard deviation, the number of runs, confidence level, and, in this case, a normal distribution.

Table 2.3: Example of reliability results calculation of neutron radiation experiment.

	SEU rate (s^{-1}) (Eq.2.6)	σ_{device} (cm^2) (Eq.2.7)	σ_{bit} (cm^2) (Eq.2.8)	$\sigma_{dynamic}$ (cm^2) (Eq.2.9)	SER (s^{-1}) (Eq.2.11)
Run 1	5.30 E-2	1.29 E-6	9.19 E-14	1.84 E-8	7.58 E-4
Run 2	5.07 E-2	1.37 E-6	9.78 E-14	1.81 E-8	6.67 E-4
Run 3	3.33 E-2	8.11 E-7	5.78 E-14	1.35 E-8	5.56 E-4
Run 4	5.94 E-2	1.45 E-6	1.03 E-13	1.27 E-8	5.21 E-4
Run 5	5.00 E-2	1.40 E-6	9.95 E-14	1.79 E-8	6.41 E-4
Average	4.93 E-2	1.26 E-6	9.00 E-14	1.61 E-8	6.28 E-4
Confidence (95%)	8.46 E-3	2.27 E-7	1.62 E-14	2.43 E-9	8.22 E-5

Using dependability concepts, the reliability function and MTTF can be calculated using the Equations 2.1 and 2.5 respectively.

$$\lambda = SER = 6.28 \times 10^{-4} s^{-1}$$

$$R(t) = e^{-6.28 \times 10^{-4} t}$$

$$MTTF = 1/\lambda = 1.59 \times 10^{3} s$$

This result means that it is expected one fault in an interval of 1.59×10^3 seconds. On the other hand, this characteristic of the system can be expressed in terms of FITs.

Considering that a FIT is the number of faults in 10^9 hours device, and it is expected one fault each $1.59 \times 10^3 s$ per device, the soft error rate in FITs for the example is:

$$SER = \frac{10^9\ hours}{1.59 \times 10^3 s} = 2.26 \times 10^9 FIT$$

If the neutron spectrum used in the experiments is equivalent to the atmosphere spectrum and the neutrons flux is known in some location, the expected soft error rate of the application running in that place can be estimated by using the $\sigma_{dynamic}$ and the flux of neutron particles according to Equation 2.13. Considering that the neutron flux at sea level is around 13neutrons/cm^2/hours, the expected SER is calculated as follows.

$$SER_{sea} = \sigma_{dynamic} \times (\#neutronflux) \tag{2.13}$$

$$SER_{sea} = 1.61 \times 10^{-8} \times 13n/cm^2/hours$$

$$SER_{sea} = 2.10 \times 10^{-7} errors\ \ hours^{-1}$$

For this example, the MTTF is estimated from the SER_{sea} value as follow.

$$MTTF = (2.10 \times 10^{-7})^{-1} hours = 4.76 \times 10^6 hours$$

The failure rate also can be expressed in terms of FITs. Since one FIT represents the number of errors expected in 10^9 hours per device, and since in our example we are testing just one device, the failure rate can be calculated from MTTF as follows.

$$SER_{sea} = \frac{10^9}{MTTF} = 210\,FITs$$

MTTF = 4.76×10^6 hours means that for the application of the example, one error is expected in 4.76×10^6 hours, which equates to around 544 years per device. In this Thesis, reliability results are shown in terms of cross-section and MTTF.

3 MITIGATION TECHNIQUES FOR SRAM-BASED FPGAS

Although in typical design of SRAM-based FPGAs less than 10% of the configuration bits may affect the design(CHAPMAN, 2010a), high reliability applications require the implementation of strategies to mitigate the failures to which the device is susceptible. These strategies should address masking and correction of faults produced. In this chapter, the most important techniques to mask and correct failures are presented.

3.1 Masking techniques

Masking techniques are used in order to ensure the correct output of the circuit. The masking is achieved by the redundancy of information, time and space. This work is based on the use of spatial redundancy, of which the most commonly used technique is the triple modular redundancy (TMR). Since our goal is to mitigate multiple bit upsets, TMR implementations and other new masking techniques are analyzed from the point of our goal.

3.1.1 TMR based techniques

Spatial redundancy is based on the replication of *n* times the original module building *n* identical redundant modules. Usually, *n* is an odd number and the most common case of n-modular redundancy (nMR) is when *n* is equal to 3, where it is called Triple Modular Redundancy (TMR).

In a first moment, TMR implementation may be classified according to how the elements are tripled (BERG, 2010). TMR can be implemented in different ways by using coarse grain (CG) TMR, or by fine grain (FG). As shown in Figure 3.1, in CG the entire target block is tripled (redundant logic), the same input signals are used by each redundant block, and their outputs are voted by a majority voter. The basic implementation of CG is also known as Block TMR (BTMR). Fine grain consists on breaking the target block it into small blocks. Then, each small block is tripled and voted as in CG TMR.

Some examples of fine grain TMR implementation are Local TMR, and Global TMR. Local TMR (LTMR) triplies each flip-flop of internal functional block. Global TMR (GTMR) uses longest area and is very complex. It triplies all elements of the design: flip-flops, inputs/outputs, routing lines, reset lines, clock lines (different clocks) and also the voters.

In (PRATT et al., 2006) it is proposed the use of a Partial TMR (PTMR), consisting on the triplication of preselected most critical elements. PTMR focus on the fact that just a portion of the configuration bits may affect the design (known as sensitive bits), and only a small part of such bits are identified as ”persistent” bits (MORGAN et al., 2005), in

Figure 3.1: Coarse grain implementation of TMR technique.

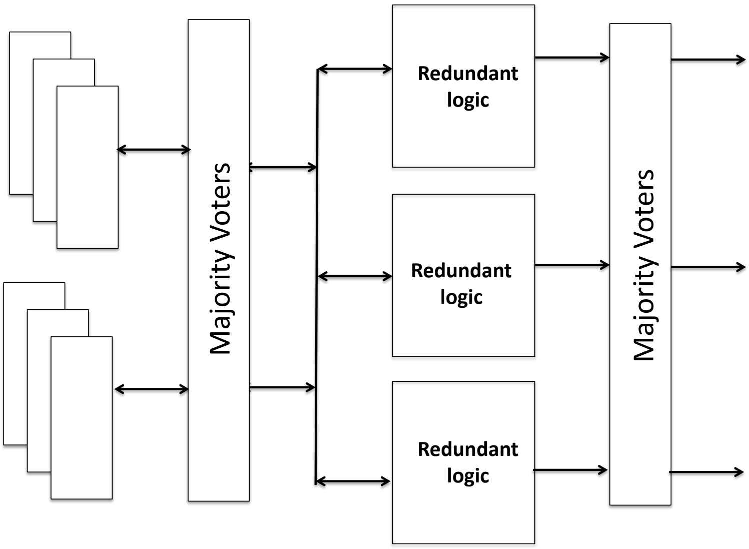


Table 3.1: Configuration sensitivity and persistence for several designs.

Design	Utilization (slices)	Sensitive bits	Persistent bits
DSP Kernel	5,746 (46.8%)	575,448 (9.9%)	13,841 (0.24%)
Syntetic	2,538 (20.6%)	189,835 (3.3%)	77,159 (1.3%)
Multiplier	10,305 (83.9%)	550,228 (9.5%)	0 (0%)
Counter	2,151 (17.5%)	201,691 (3.5%)	108,750 (1.9%)

(PRATT et al., 2006)

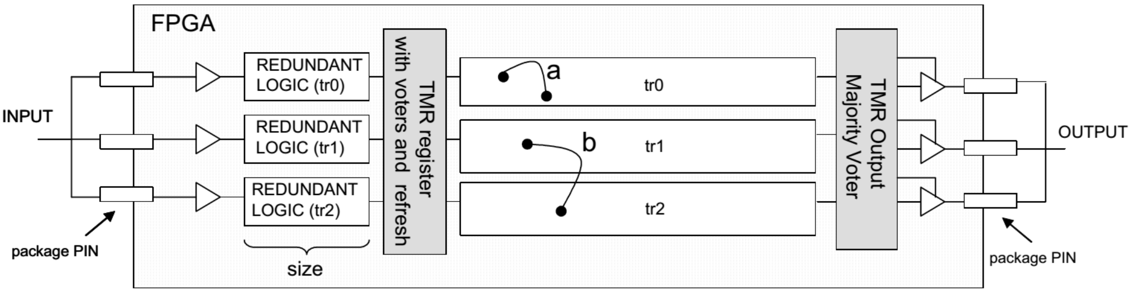
which it is not enough to correct the fault but must also reset the circuit to return to normal operation. The acknowledgment of the sensitive bits and persistent bits is performed by fault injection campaigns in the target circuit. In (PRATT et al., 2006), four different designs (Syntetic is made from feedback LFSRs that feed an array of multipliers and adders) were implemented and the percentage of their sensitive and persistent bits are presented in Table 3.1. These results verify the low rate (around 10%) of configuration bits that can affect the design, and consequently, this fact justifies the protection by TMR of only some elements aiming to have a good level of reliability with a minimum overhead of resources.

TMR implementations in FPGAs have the drawback that a single fault may affect more than one module, causing the crash of the system due TMR only masks a single fault in the circuit copies that feeds the voter inputs. For example, if a single upset hits on a routing cell, a fault that affects two modules at the same time may be happened and consequently two out of three voter inputs receive faulty results and the circuit will produce wrong answers. Figure 3.2 from (KASTENSMIDT et al., 2005) shows an example where one single upset (*b*) produces the shortcut of lines ”tr1” and ”tr2”.

Aiming the reduction of the possibility that a single upset affects two redundant modules, Partitioned TMR was studied in several works (KASTENSMIDT et al., 2005; MC-

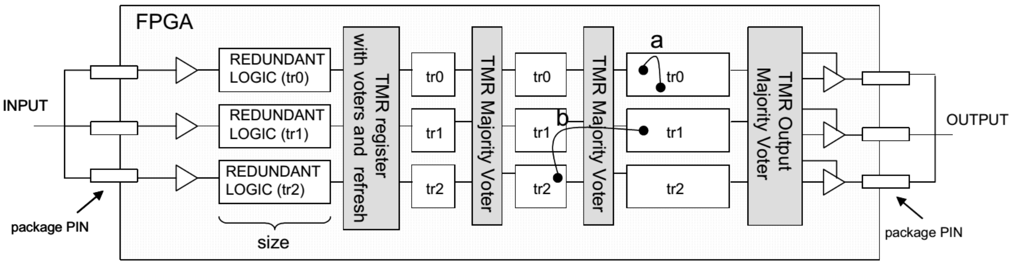
MURTREY et al., 2006; WANG, 2010; MANUZZATO et al., 2007). The idea is to divide each block into smaller blocks and vote the output of each block. Figure 3.3 from (KASTENSMIDT et al., 2005) is an example of this proposal, where the communication line is divided and voted avoiding that the single fault *b* provokes that two modules fail.

Figure 3.2: Wrong result of a traditional coarse TMR implemented in FPGA affected by a single upset.



(KASTENSMIDT et al., 2005)

Figure 3.3: Correct output in fine grain TMR implemented in FPGA affected by a single upset.

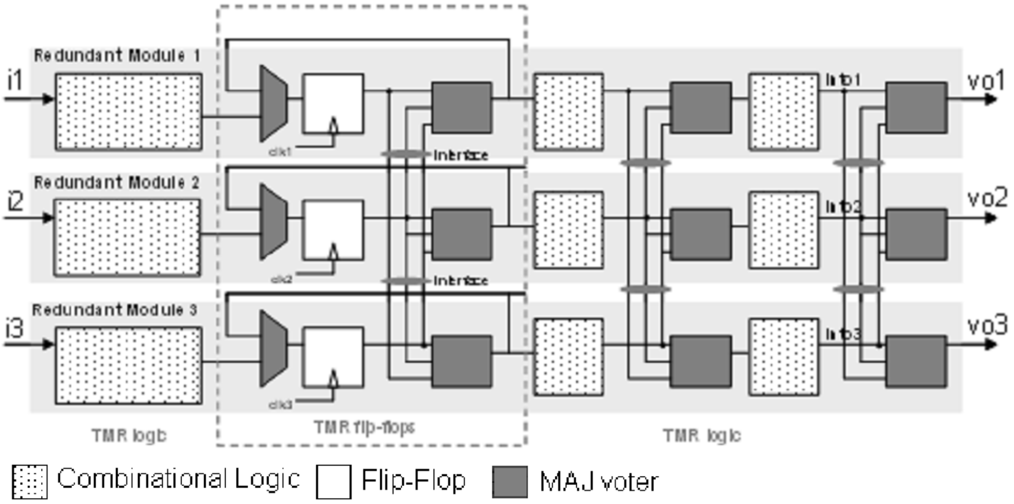


(KASTENSMIDT et al., 2005)

Following the same philosophy, XTMR (CARMICHAEL, 2006) proposes the use of the voting of all user flip-flops of the circuit connecting them in feed back to correct the upsets in those registers as shown in Figure 3.4. The implementation of XTMR is a complex process, but can be automatized by software tools as XTMR Tool from Xilinx. Having each flip-flop in feed back with the corresponding output voter, it is guarantied not only the masking of faults but also the correction of faults in user flip-flops. This is very useful when the goal is avoid the continues resynchronizing of the system. On the other hand, with the insertion of voters in each flip-flop, the level of partitions is very high and the resources overhead is also higher than other versions.

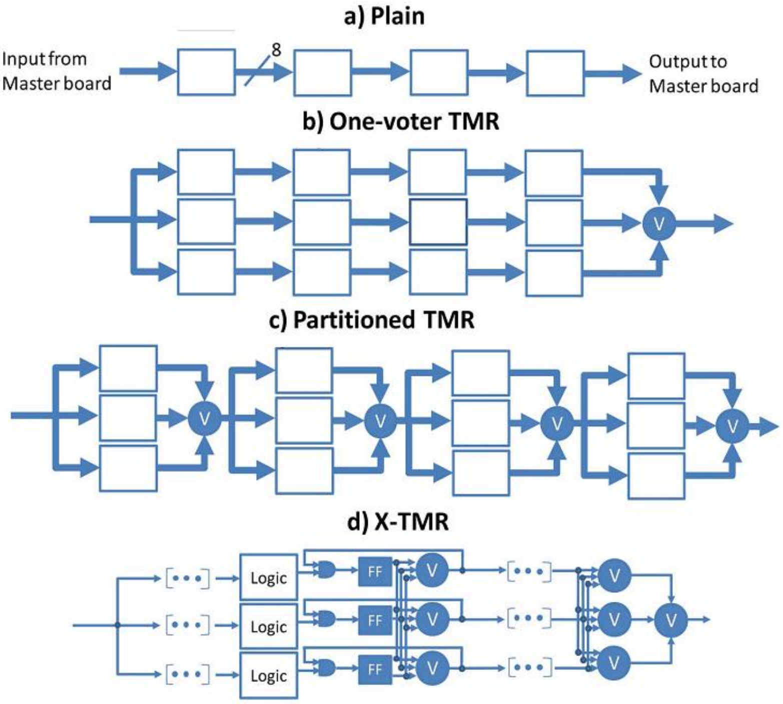
Commonly, voters have low fault sensibility due they are small circuits compared with the triplied logic block. Since the increase of partitions, and hence the number of voters, increases the reliability of the circuit, XTMR is an efficient masking technique. In (MANUZZATO et al., 2007), the common one voter TMR (LTMR), partitioned TMR and XTMR techniques were applied to protect four PicoBlaze soft microcontrollers (XILINX, 2005) running simple averaging filter, and its results were compared against an unprotected version. Figure 3.5 shows the diagram of the four tested circuit. All of them were

Figure 3.4: Schematic of fine grain TMR known as XTMR.



(MANUZZATO et al., 2007)

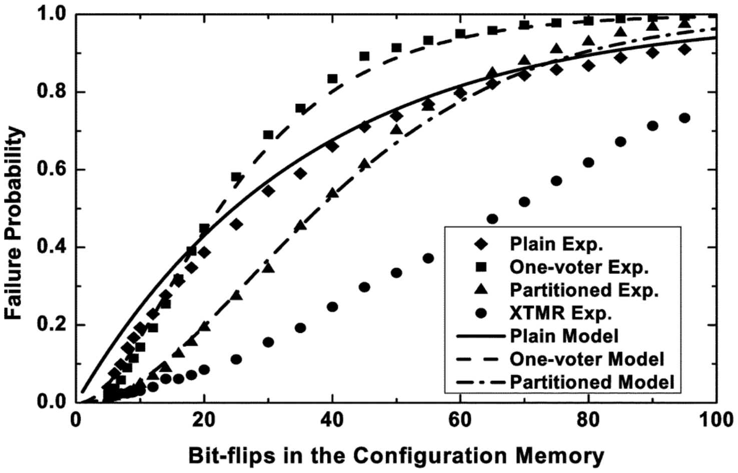
Figure 3.5: TMR schemes validated in (MANUZZATO et al., 2007).



(MANUZZATO et al., 2007)

implemented in 90-nm FPGA Spartan-3 XC3S200 and irradiated by using an Americium source emitting alpha particles with an energy of about 5.4 MeV and flux of 1.54310^4 alphas s^{-1} within a solid angle of 2π sr. Results show that in average, the number of errors per minute are 1.16 for the unprotected version, 1.43 for the one-voter TMR version, 0.91 for the partitioned TMR version, and for 0.51 XTMR version. According to these results, the mitigation capability for the circuits under test in the experiment conditions, XTMR mitigates about 2.8 times the one-voter TMR. Notice that although the unprotected version has less errors per minute compared with the one-voter TMR version, it is not possible to guarantee a free-fault output if no redundancies (and voters) are used. In the same paper, an analytic model to compare the reliability features of each implementation is proposed. Results for experimental and modeled implementations are presented in Figure 3.6. Figure 3.6 shows the failure probability of the unprotected version (Plain Exp.), common TMR (One-voter Exp.), partitioned TMR and XTMR as a function of the number of accumulated bitflips using the proposed model (except XTMR) and radiation experiments. As expected, XTMR has the lower failure probability for whatever number of accumulated faults. Compared to unprotected version, the one-voter TMR is only efficient until approximately 20 faults, after that it is better to use the unprotected version. The efficiency of the partitioned TMR (4 parts) is better than unprotected version until about 70 accumulated faults. Results also suggest one more time that when bitflips are accumulated, the failure rate for the common TMR is higher than for the unprotected version. This fact can be explained by the higher number of resources used by TMR compared with the unprotected version. However, as we explained, in case of unprotected version, it is not possible to signalize the moment when the module fails.

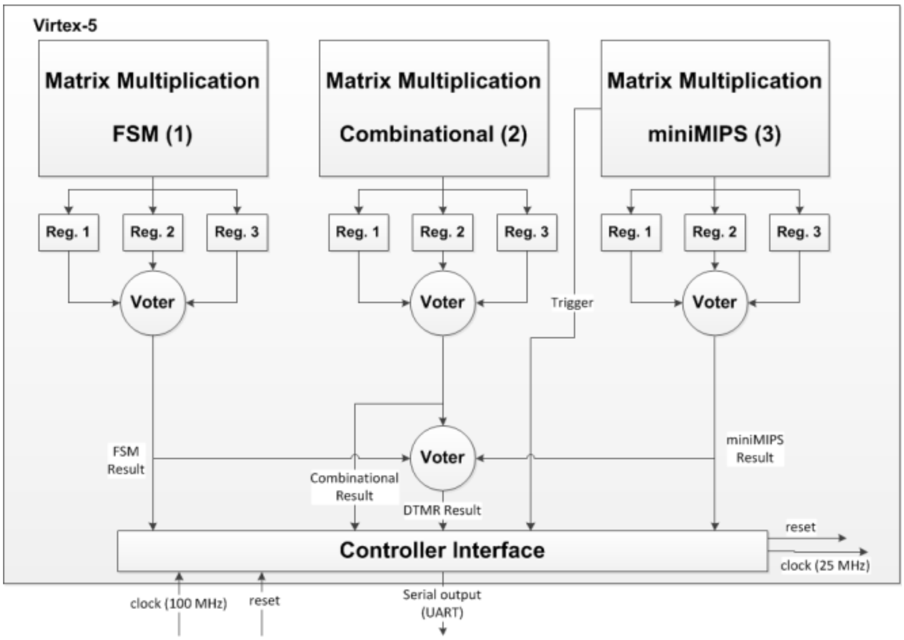
Figure 3.6: TMRs comparison results for different implementations.



(MANUZZATO et al., 2007)

In resent years, a new TMR implementation for SRMA-based FPGAs was proposed in (TAMBARA L.; RECH, 2013) known as Diversity TMR (DTMR), consisting in the implementation of the same function by three different designs. Diversity designs was explored many years ago (LALA; HARPER, 1994) and used in device level in on-board computers (RITER, 1995), bus networks (ASHRAF et al., 2011), and mixing-signals platform (HIARI; SADEH; RAWASHDEH, 2012). Since different implementations have different times of execution, it is necessary to latch the partial results to be voted at the same time. In (TAMBARA L.; RECH, 2013), DTMR was used to execute a matrix multiplication by three different methods: software implementation running in a miniMIPS soft processor, combinatorial implementation and finally using a finite state machine (FSM) as presented in Figure 3.7. The circuit under test was implemented in a Virtex-5 LX110T FPGA from Xilinx fabricated in 65-nm copper CMOS process technology, and was irradiated in ISIS facilities by neutron particles flux of $3.98x10^4 n/cm^2/s$ with energies above 10 MeV during 1,268 minutes. The results show that the cross-section of DTMR scheme is 36.1% smaller than traditional TMR, reflecting the higher masking capacity of DTMR technique.

Figure 3.7: DMR-MIPS proposed in (TAMBARA L.; RECH, 2013).



.

(TAMBARA L.; RECH, 2013)

3.2 Correction techniques

Masking techniques avoid the propagation of faults provoked by upsets to the output system, but however, the upset bits remain and accumulate into the configuration memory reducing the reliability of the device and consequently increasing the probability to have a wrong output. In order to correct the faults, it is necessary to rewrite the correct configuration bits into the memory. In this section we review the most important ways to implement the correction of faults.

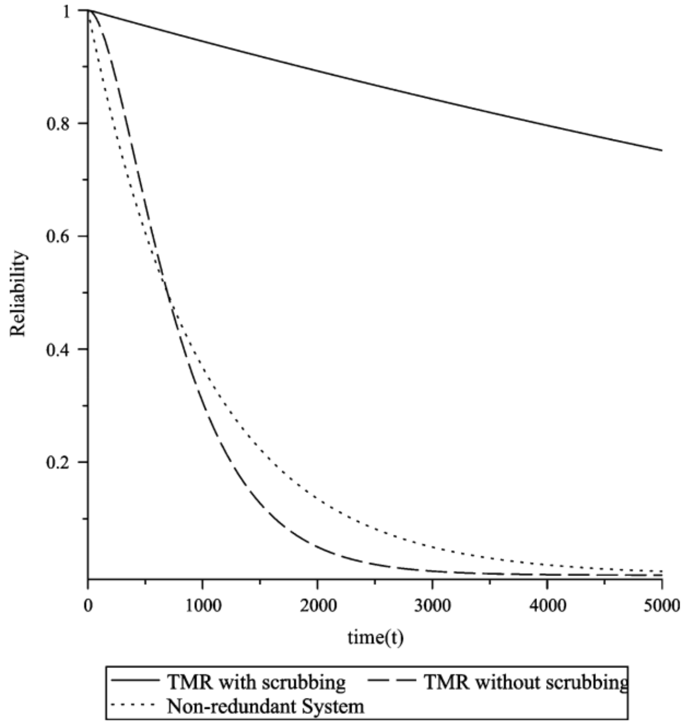
Upsets in configuration memory bits of the FPGA can only be corrected by reloading the original value of bitstream. One possible method is to reconfigure the FPGA during power cycling (when the FPGA is powered up), or at idle state (CHAPMAN, 2010b), depending on the application. However, the recommended option is to refresh the configuration memory bits without the need to stop the system application, process known as scrubbing (HERRERA-ALZU; LÓPEZ-VALLEJO, 2013). Scrubbing is a process used also in SRAM-based memories, and consists in rewriting a portion or the entire data memory. In high reliability applications, the use of masking technique with scrubbing is mandatory. Figure 3.8 shows the reliability increment of TMR when used scrubbing.

There are several implementations that we explain according to the scrubber location respect the device (internal or external), the portion of configuration bits repaired (full or partial scrubbing), and depending of the time of the execution (blind or fault location). In the following subsection, we explain these scrubbing implementations.

3.2.1 Blind or fault detection

In blind scrubbing, the reload of the golden bitstream is performed in a fix rate which. It is recommendable use an scrubbing rate of 10 times the expected SEU rate (ADELL; ALLEN, 2008). The goal of blind scrubbing is prevent the propagation of the fault considering the worst case in terms of rates, having the penalty to write the golden bitstream

Figure 3.8: Reliability effects of scrubbing in circuits protected by TMR.



(MCMURTREY et al., 2006)

from an external device, which increases the power penalties. Moreover, only a small part of fault bits have a real impact in the design. According to (CHAPMAN, 2010c), around 10% of the configuration bits of the bitstream are not used used by the FPGA, and any design may use just around 20% of the available configuration bits. Although a reduced portion of the configuration memory may impact in the design, in blind scrubbing, the memory refresh process considers the upset rate in any configuration bit of the bitstream as a criteria to perform the scrubbing, which may represent a waste of resource.

In order to reduce the scrubbing rate, we can read the configuration bitstream and compare it with a golden bitstream, prcess known as readback and scrubbing (BERG et al., 2008; LUO; ZHANG, 2011). Readback is the process by which the configuration memory of the FPGA is read. Thus, the rewrite of the configuration memory is performed only when the bitstream read by readback process and the original bitstream do not match. However, it is also necessary to have the golden bitstream, to read constantly the configuration memory and also compare them, which is a slow procedure.

Aiming to have a more efficient way to scrub the full memory, Virtex FPGAs protect the bitstream with CRC (Cyclic Redundancy Check) codes. CRC code is computed during the construction of the bitstream (golden CRC) by the synthesis tool, and then, it is not necessary to read the full bitstream to know is any configuration bit is upset. Moreover, these FPGAs provide a built-in circuit to detect periodically the current CRC code, and consequently, just both CRC codes must to be compared to determine if some bit is upset (OSTLER et al., 2009). The drawback is that CRC only detect until two upset bits, which is not enough in new technologies.

3.2.2 Full or partial scrubbing

Full scrubbing is the simplest implementation mode considering that it is not necessary to use any extra technique to locate the upset bit or bits. However, it is the worst option in terms of correction time (time to repair) and used resources because it is necessary to write the full configuration memory despite only a small portion of bits are upset, and for this, it is also necessary to access frequently to an external device where the full bitstream is stored.

Partial scrubbing is the most attractive solution in terms of correction time and power consumption because usually, only a set of bits are upset. The challenge is to detect the location (or locations) of the configuration frame (or frames) in which there are upset bits, and then, rewrite only such frames.

Many families of FPGAs, such as Virtex-5, have built-in ECC (error correction code) circuit by frame to detect and correct (if possible) some upset in program memory. At the begining, ECC code is computed and included into each configuration frame by the synthesis tool. Built-in ECC circuit may validate the current ECC of a specific frame indicated by a control circuit, and provides the appropriate information to correct the upset bit (if only one bit is upset), or detect up 2 upset bits. Notice that the correction of the frame must to be performed by a designed circuit. The main drawback of using ECC is that it can detect the position of a single fault and detect up two faults in the same frame.

For multiple faults, it may be necessary the use some design error detection circuit at the user design level. For example, in coarse grain TMR or nMR, the majority voter can be used not only to mask errors but also to signalize which redundancy module is faulty. If we have information about the placement of each block, we may rewrite only the portion of configuration bitstream that belongs the fault module independently of the number of upset bits in such block. This process can be performed through the use of dynamic partial reconfiguration (DPR) as proposed in many works (CARMICHAEL; CAFFREY; SALAZAR, 2000; PRATT et al., 2006; AZAMBUJA et al., 2009; STERPONE; ULLAH, 2013; BOLCHINI; MIELE; SANTAMBROGIO, 2007).

3.2.3 Internal and external scrubber

The circuit that controls the scrubbing process (scrubber) may be located into the FPGA or in a external device (BERG et al., 2008; HEINER; COLLINS; WIRTHLIN, 2008). Although the scrubbing process in the first case is simple and fast, the scrubber circuit is also susceptible to radiation effects. The effectiveness of both options was studied in (BERG et al., 2008). Xilinx proposes the use of a internal scrubber named Xilinx SEU-Controller (CHAPMAN, 2010b) for Virtex-5 FPGAs based on a Picoblaze soft processor (XILINX, 2005) and a similar Intellectual Property (IP) circuit (Soft Error Mitigation Controller - SEM) for latest (XILINX, 2010d) devices. SEU Controller checks frame by frame the ECC and CRC information to detect, and if possible, correct any flipped configuration bit. The lengths of ECC and CRC are only few bits, and they depend on the configuration frame size and on the configuration bitstream. In (HEINER; COLLINS; WIRTHLIN, 2008) a fault tolerant processor was proposed to control the scrubbing from the internal FPGA.

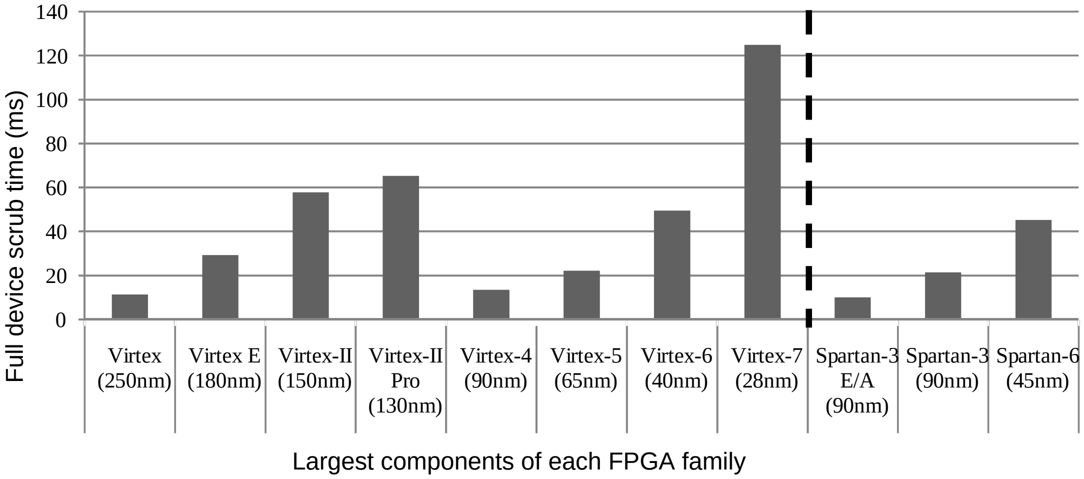
Table 3.2 presents an overview of the qualitative comparisons of fault correction techniques according to their implementations: scrubbing by using an internal or external configuration port (internal CP and external CP respectively), full or partial scrubbing, readback (for detection) and scrubbing. The correction time depends on the amount of bits to be reloaded. Since a frame is the smallest addressable portion of configuration bits, ECC frame protection presents the reduced correction time T_{ECC}. In some cases, the number of bits protected by techniques of low granularity can be as small as one frame, then T_{FineG} may be similar to T_{ECC}. Because the external configuration port access is slower than the internal one, their correction times will be higher. The power consumption depends on the number of bits to be scrubbed and the type of configuration port used. External configuration port consumes more than the internal one since it uses I/O pins. Correction time and power consumption comparisons presented in Table 3.2 are based on the features of the scrubbing technique, and represents a qualitative comparison. Notice that a single event functional interrupt (SEFI) can be induced by a SEU hitting into the configuration circuit, and then the configuration port is not reliable. This situation can be detectable by implementing a readback.

Table 3.2: Qualitative comparison of configuration memory correction techniques.

	Fault repair techniques	**SEU detection technique**	**SEU correction capability**	**Special placement and floor-planning needed**	**Detect SEFI**	**Correction time**	**Power consumption**
Internal CP	**Partial Scrubbing**	Detection at design level (low grain)	Single/multiple per small logic/resource module	Yes	No	$T_{FineG} \approx T_{EC}$	P_{FineG}
		Detection at design level (coarse grain)	Single/multiple per redundant module	Yes	No	$T_{CoarseG} > T_{FineG}$	$P_{CoarseG} > P_{FineG}$
		ECC per frame from Xilinx	Single/multiple per frame	No	No	T_{ECC}	$P_{ECC} < P_{CoarseG}$
External CP	**Full Scrubbing**	No need (blind)	Single/multiple	No	No	$T_{BS} > T_{PS}$	$P_{FS} > P_{PS}$
		CRC bitstream from Xilinx	Detect single per bitstream	No	No	$T_{CRC} \approx T_{BS}$	$P_{CRC} > P_{PS}$
	Partial Scrubbing	Any	Single/multiple	Yes	No	$T_{PS} > T_{CoarseG}$	$P_{PS} > P_{CoarseG}$
	Readback and Scrubbing	No need	Single/multiple	No	Yes	$T_{R\&S} > T_{BS}$	$P_{R\&S} > P_{FS}$

The scrubbing rate is an important design parameter that depends on the SEU rate. Moreover, it is necessary to take account that the scrub time depends on the size of the configuration bitstream of the FPGA. Due FPGAs have more and more resources, the size of the configuration bitstream is increasing and consequently the scrub time. Figure 3.9 shows the scrubbing time in milliseconds (ms) of the largest component of each family FPGA, when the scrubber is performed by a soft processor running at 100 Mhz.

Figure 3.9: Scrub time for different components of FPGAs.



Finally, Table 3.3 shows some examples of combinations of SEU masking and correction techniques. Due CRC and readback techniques detect SEUs in the full configuration bitstream, it is necessary to use full scrubbing, otherwise is possible to use partial scrubbing.

Table 3.3: Examples of combinations of SEU masking and correction techniques.

Fault repair techniques	**Masking techniques**			
	XTMR (CARMICHAEL, 2006)	Fine grain (PRATT et al., 2006; NIKNAHAD; SANDER; BECKER, 2012; NAZAR; SANTOS; CARRO, 2013)	Coarse grain TMR (AZAMBUJA et al., 2009; STERPONE; ULLAH, 2013)	Coarse grain NMR
Partial Scrubbing	Using ECC frame (BRIDGFORD; CARMICHAEL; TSENG, 2008)	Using ECC frame (BRIDGFORD; CARMICHAEL; TSENG, 2008)	Using ECC frame or faulty region detected by design level detection technique (BRIDGFORD; CARMICHAEL; TSENG, 2008)	Using ECC frame or, faulty region detected by design level detection technique (BRIDGFORD; CARMICHAEL; TSENG, 2008)
Full Scrubbing	Blind scrubbing (BRIDGFORD; CARMICHAEL; TSENG, 2008)	Blind scrubbing (BRIDGFORD; CARMICHAEL; TSENG, 2008)	Blind scrubbing (BRIDGFORD; CARMICHAEL; TSENG, 2008)	Blind scrubbing (BRIDGFORD; CARMICHAEL; TSENG, 2008)

3.3 Handling Multiple Bit Upsets in SRAM-based FPGAs

Knowing that in recent technologies the probability to have MBUs is higher, recent works propose different approaches to lead with this new context.

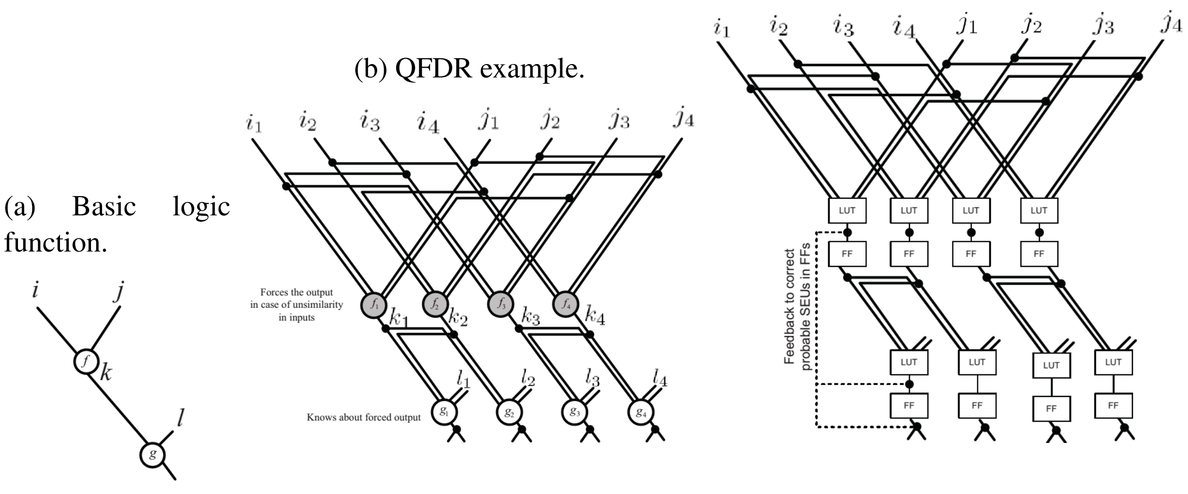
3.3.1 Quadruple Force Decide Redundancy

Other example of fine grain technique is presented in (NIKNAHAD; SANDER; BECKER, 2012), where the protection at LUT level through the use of Quadruple Force Decide Redundancy (QFDR) is proposed.

The QFDR is the generalization to boolean function of the Quadded Logics (QL) technique which are used to clean the errors up in logic gates. Considering a function f with two inputs i and j as presented in Figure 3.10a, the technique consists in the quadruplication of the logical function and in the duplication of their inputs as shown in Figure 3.10b. Any difference in duplicated inputs means that one of them has incorrect value and the output flag is forced to zero. This additional information is used by the next level to use only the correct information masking the fault. Figure 3.10c shows an example of the use of QFDR in FPGA using feedback. In the same work, a tool to automatized the implementation of QFDR was also proposed.

Aiming the validation of QFDR, authors protected six benchmark circuits using TMR and QFDR. Then, they use ModelSim simulation tool to inject faults. According to (NIKNAHAD; SANDER; BECKER, 2012), results shown that the best fault tolerance obtained by QFDR was with the PicoBlaze processor as benchmark. In such case TMR masked 42% of the injected faults and QFDR masked 56.4%.

Figure 3.10: Quadruple Force Decide Redundancy proposed in (NIKNAHAD; SANDER; BECKER, 2012).


(c) QFDR implementation in FPGA.

(b) QFDR example.

(a) Basic logic function.

i
j
f
k
l
g
i1 i2 i3 i4 j1 j2 j3 j4
Forces the output in case of unsimilarity in inputs
f1 f2 f3 f4
k1 k2 k3 k4
l1 l2 l3 l4
g1 g2 g3 g4
Knows about forced output
LUT
FF
Feedback to correct probable SEUs in FFs

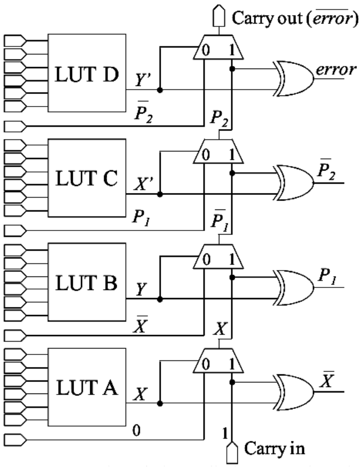
(NIKNAHAD; SANDER; BECKER, 2012)

3.3.2 Fast detection

In order to lead with high SEU rates, in (NAZAR; SANTOS; CARRO, 2013), authors propose a fine grain fast fault detection technique through dual modular redundancy (DMR) to reduce the MTTR. Since classical techniques require fine-grained use many voters, the resource overhead in terms of CLB could hinder its implementation. The technique proposed in (NAZAR; SANTOS; CARRO, 2013) takes advantage of hardwired dedicated carry chains available in CLBs of Virtex FPGAs which are used in a limited set of applications and, consequently, are often wasted in most typical designs.

Figure 3.11 depicts how the carry chains of a CLB is used to compare two pairs of duplicated LUTs: X (LUT A) and Y (LUT B), and their replicas X' (LUT C) and Y' (LUT D). Notice that this proposal is focused on the protection of combinatorial logic, and it does not mask any fault. In the same paper, 21 combinational benchmarks from the ISCAS 85 and MCNC benchmark were used to compare their area and delay overhead when traditional DMR coarse grained and proposal DMR multi grained are used. Despite the area overheads are closed, the number of clock cycles to detect the fault are less in DMR multi grained technique, getting the speed up of the detection process.

Figure 3.11: Carry propagation chain applied to error detection. X' denotes the replica of net X.



(NAZAR; SANTOS; CARRO, 2013)

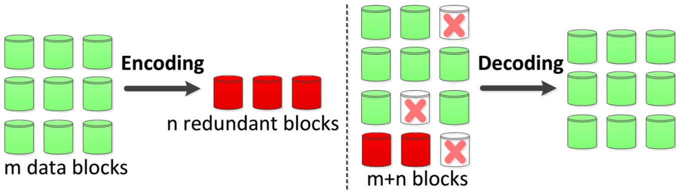
3.3.3 Use of erasure codes to correct MBUs in configuration frames

In (RAO et al., 2014), it is proposed a technique to detect MBUs in configuration memory frames through of of 2-dimension parity with intervals, and also correct them through erasure codes. They join a group of configuration frames as a matrix and calculate their parities (golden parities) by rows and columns in different interleaving distances. Then, it is necessary to read through ICAP primitive the information of the frames and recalculate the parities to compare them with the golden ones. The converge of multiple upset bits depends on the value of interleaving distances, but also these parameters will affect the overhead. Notice that the upset detection is made by frames instead of by bits.

Once the error frames are detected, erasure codes are used to reconstruct them as shown in Figure 3.12. In this technique, m blocks of frames are transformed into $m+n$ blocks, such that the original m blocks can be recovered using the n coded blocks. The value and dimension of n depends on the number of possible reconstructed frames of m, so, m and n define also a trade of between the capability to prepare multiple upset bits, and area and time overhead. Similar to ECC codes, the correction process requires the reading each the frames (through ICAP) that belongs to the block, calculating and comparing the current n block with the stored blocks, and repairing the upset frames (rebuild the frames and rewrite them through ICAP).

This technique was performed in a Virtex-6 VLX240T FPGA and compared against Xilinx fault detection techniques (ECC, CRC, SEU correction macro) and hamming codes. Results show that although the average detection time of the proposal presented in 3.12 is almost the same that the techniques proposed by Xilinx (9.343 ms), Xilinx techniques do not recover frame or group of frames with errors.

Figure 3.12: Encoding and Decoding of Erasure Codes.



(RAO et al., 2014)

3.4 Summary of techniques

Table 3.4 summaries the masking techniques presented in this section, comparing the masking capabilities of single, multiple and massive faults, the actions followed after one module is faulty and also the need of resynchronization after the scrubbing process.

Table 3.4: Characteristics of masking techniques.

	Capability of Masking SEUs in configuration memory bits			**Action after one module is faulty**	**Resynchronization of modules after scrubbing**
	Single faulty modules	Two faulty modules	High number of accumulated upsets provoking multiple faulty modules		
XTMR	Yes	Low chance, it depends on faulty signals and bits voted by the voters	No	Scrubbing is needed	Usually no synchronization is needed as majority voters are used in the flip-fliops with feedback voter.
Coarse grain TMR	Yes	No	No	Scrubbing is needed	Reset of the modules is needed
Coarse grain nMR	Yes	Yes	Yes up to the capability of the majority voter	No need scrubbing until n-2 modules are faulty	Reset of the modules is needed

3.5 Testing the radiation effects in SRAM-based FPGA

3.5.1 Sea level radiation test

TID experiments aim to characterize the maximum ionizing radiation dose absorbed by the device in which functional and parametric features remain proper. The most common radiation source used to perform these destructive experiment is the γ ray from Co-60 source. Since the effect on the device is the degradation in electrical parameters, propagation delays, supply current, I/O current, and functional errors are usually monitored (REZGUI et al., 2012; KASTENSMIDT et al., 2011).

Faults induced by SEUs are the main concern in SRAM-based FPGA. At sea level, experiments are performed in specialized laboratories that commonly use protons, heavy ions or neutrons particles as source of radiation to induce SEUs in the target device. As the goal of these experiments is to describe the susceptibility of the target circuit to SEU effects, functional errors as the state of the configuration bits must to be constantly monitored.

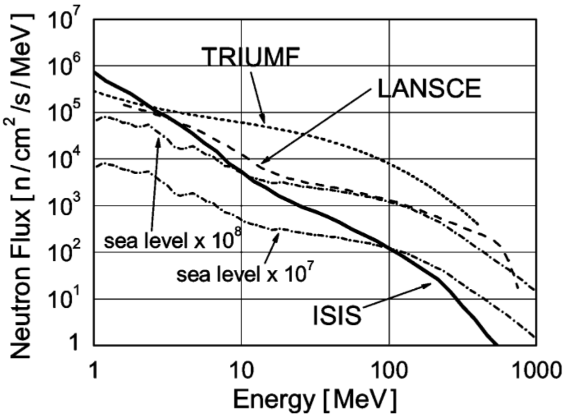
SEU rate caused by protons and heavy ions (charged particles) are usually higher than obtained by neutron experiments, but also experiments with protons and heavy ions are more expensive and complex process. Since at sea level neutrons are the main source of SEUs, neutron accelerators try to replay the same characteristics than in the atmosphere but with higher flux energy to obtain more faults in short time. Experiments with neutrons were performed to evaluate fault tolerant techniques, and also to characterize component for aerospace applications (NORMAND; DOMINIK, 2010; ZHU; SONG; PAN, 2013; XILINX, 2014a).

There are few facilities that provide neutron beams matching the terrestrial flux. ISIS (in Rutherford Appleton Laboratory - Didcot, U.K) (ISIS, 2014), LANSCE (Los Alamos Neutron Science Center - New Mexico, USA) (LANSCE, 2014), TRIUMF (Canada's national laboratory for particle and nuclear physics - Vancouver, Canada) (TRIUMF, 2014), and RCNP (China Institute of Atomic Energy - Beijing, China) (RCNP, 2014) are examples of neutron beams that feature an energy spectrum that is similar to the terrestrial but considering high acceleration factor (VIOLANTE et al., 2007). Since the neutron flux spectrum of facilities aims the survey of SEUs effects caused by atmosphere neutrons, it is possible to compare the results of different experiments performed in such facilities.. (PLATT et al., 2008; XILINX, 2011a). Figure 3.13 shows the spectrum comparison between ISIS, LANSCE, TRIUMF and atmospheric neutron flux. The scheme of ISIS facility is shown in Figure 3.14a. Neutrons produced at ISIS follow the spallation process that consists on the bombarding of a heavy-metal target (tungsten) with pulses of highly energetic protons, generating neutrons from the nuclei of the target atoms. The energy of the produced neutrons is reduced through a moderator, which can be of different types. The resulting neutron beam reaches 26 different lines, including the VESUVIO irradiation chamber depicted in Figure 3.14b. The ISIS spectrum integrated above 10 MeV yields $7.86 \cdot 10^4 n \cdot cm^{-2} \cdot s^{-1}$ on the irradiated device(VIOLANTE et al., 2007).

3.5.2 SEU emulation by bitstream manipulation

Fault injection by bitstream manipulation is an important methodology to inject faults in an SRAM-based FPGA to predict the SEUs and MBU effects in the design. The emulation of SEUs and MBUs in the configuration memory are performed by flipping the configuration bits on an FPGA. The main goal of this approach relies on the fact that it allows fast injection campaigns in configuration memory, once the circuit under test

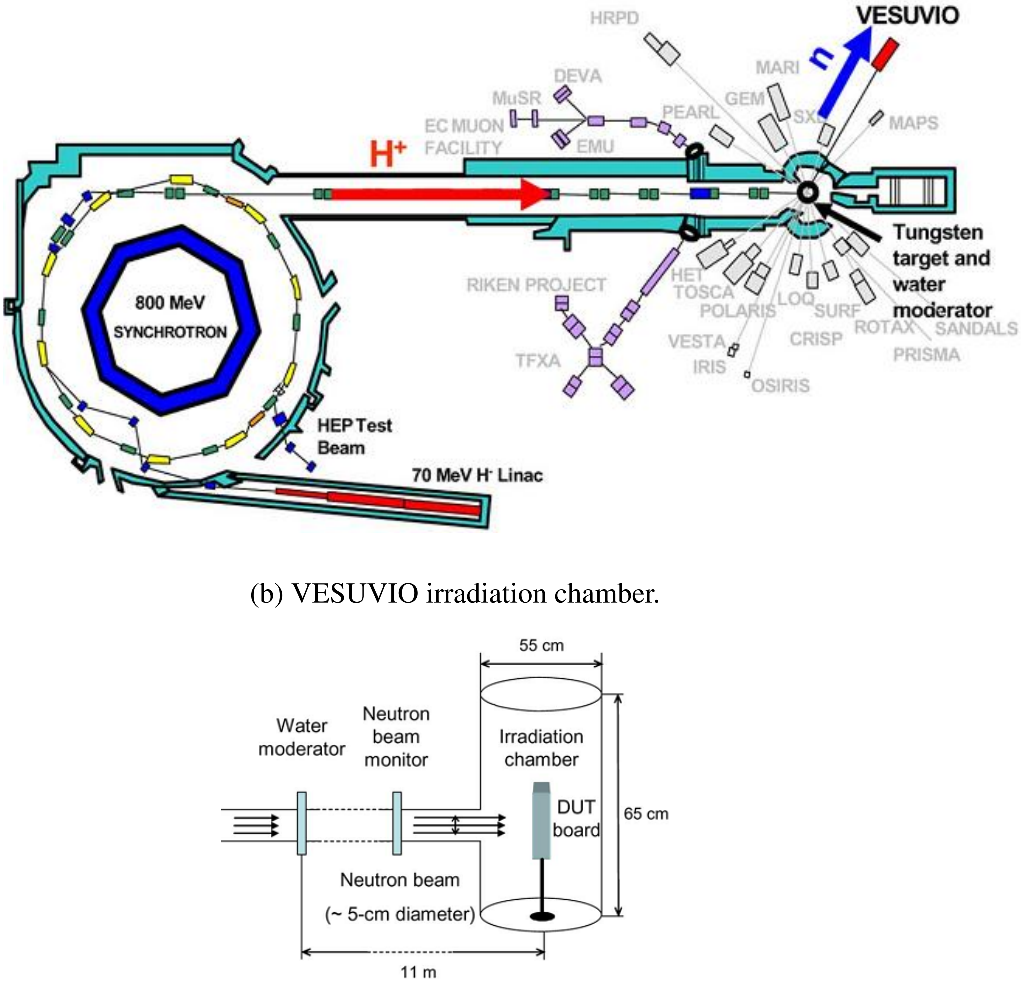
Figure 3.13: Neutron spectrum comparison between the ISIS, LANSCE and TRIUMF facilities and to the terrestrial one at sea level multiplied by 10^7 and 10^8.



(VIOLANTE et al., 2007)

Figure 3.14: ISIS facility and VESUVIO scheme.

(a) ISIS neutron facility.



(b) VESUVIO irradiation chamber.

(VIOLANTE et al., 2007)

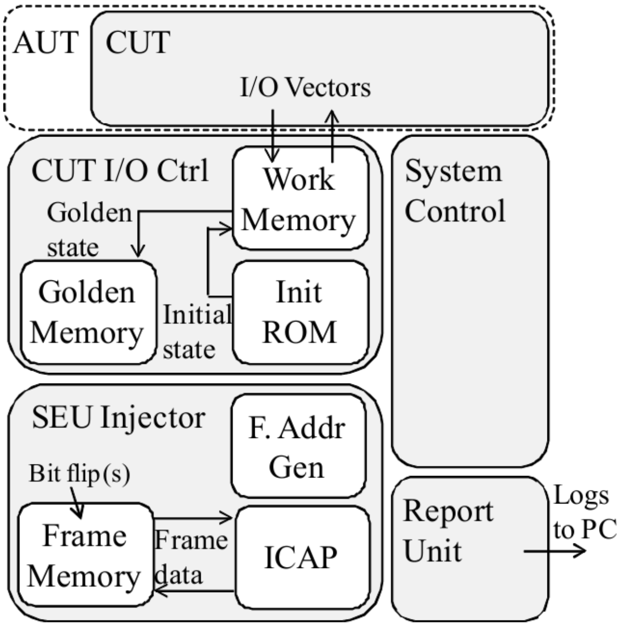
(CUT) executes at the full FPGA speed and not on simulation software which only emulate the SEUs effects on LUTs and user flip-flops. Moreover, comparing to radiation tests on particles accelerators, the amount of injected faults per unit of time (upset rate) is much higher, since a bit-flip is directly injected in the memory cell, not depending on the possibility of a particle flips or not a bit in the configuration memory. The control of the test is also superior comparing to a radiation test, since a precise location is flipped (a known bit), which allows the user reproducing a real radiation test.

The fault injection can be performed by an external or internal (depending of the configuration resources of the device) programmable port of the FPGA. FLIPPER fault injection platform (ALDERIGHI et al., 2009) is based on a mother control board (based on XC2VP20 device) which controls the fault injection process of a DUT board (based on (XQR2V6000 device) by means of the external configuration port of the FPGA. The experiment setup and control process are made through a software application running in a host PC that interacts with ModelSim simulation tool, at each clock edge. SEUs are injected by active partial reconfiguration into a randomly chosen configuration memory location successively, accumulating bitflips (SEU) in the configuration memory until a functional fault is detected. FT-SHADES (AGUIRRE et al., 2007) also uses partial configuration to inject faults in microprocessors implemented in FPGAs.

Some FPGA devices allows the configuration of their elements trough internal configuration ports. For example, Virtex FPGAs from Xilinx provide an internal configuration access port (ICAP) primitive. (XILINX, 2012c) which makes possible to reconfigure frame by frame without the necessity of using input/output pins. In (STERPONE; VIOLANTE; REZGUI, 2006), designs protected by TMR are availed by a fault injector based on ICAP. The fault locations are defined in a Fault List Manager(FLM) that is used by a Fault Injection Manager (FIM) to perform the injection. In the same paper, the FLM was composed by 5000 fault locations randomly selected. The Virtex-5 SEU Controller (CHAPMAN, 2010b) from Xilinx take advantage of the ICAP to inject bit-flips in random way by means of soft-core processor PicoBlaze. Virtex-5 SEU Controller can inject one SEU or two SEUs in contiguous configuration bits emulating an MBU. However, since such bit-flips are injected in random configuration bit of the device, the injector can also be affected by the fault injected.

In order to avoid the possibility to the inject faults in the same injector, in the injector platform proposed in (NAZAR; CARRO, 2012), an area under test (AUT) is defined to constrain the candidate configuration bits to be flipped and belong to the circuit under test (CUT). Such platform is implemented in a Virtex-5 FPGA (XC5VLX110T component) and uses the ICAP primitive to perform the fitflip, a CUT I/o control to manage the input/output of the CUT and detect errors according to golden information, and the SEU injector where the ICAP is controlled and the bit-flip position is selected. Moreover, a report unit control is used to send the experiment logs to an external PC. Figure 3.15 depicts the fault injector components implemented in the FPGA. In (NAZAR et al., 2013), authors use neutron radiation and fault injector proposed in (NAZAR; CARRO, 2012) to evaluate the detection capabilities of dual modular redundancy (DMR) technique implemented in coarse grain (DMR-CG) and fine grain (DMR-FG). Results presented in Table 3.5 are classified in 3 categories: *Detect only* category represents the number of events where each technique detected errors but the output results were right; *Detect & PO* category represents the number of errors detected by each technique and errors detected in the output of each circuit; PO Only techniques represents the number of errors detected at the output of the circuits but not detected by any technique. As shown, a high number of faults injected were implemented, and the discrepancy is low when a high number of experiments are implemented (between 2.86% and 3.87%) but is high when few number of events are detected: 71.83%.

Figure 3.15: Fault injection system proposed in (NAZAR; CARRO, 2012).



(NAZAR; CARRO, 2012)

Table 3.5: Comparison of results obtained by the fault injector proposed in (NAZAR; CARRO, 2012) and by neutron experiments, testing fine and coarse grain of DMR technique.

	Radiation			**Fault Injecion**			
	DMR-FG	DMR-CG	Ratio	DMR-FG	DMR-CG	Ratio	Ratio variation
Detect Only	396	245	1.62	89872	5775	1.56	3.87%
Detect & PO	287	223	1.29	69701	55706	1.25	2.86%
PO Only	5	6	0.83	571	193	2.96	-71.83%
Total	688	474	1.45	160144	113654	1.41	3.01%

(NAZAR et al., 2013)

4 PROPOSED SELF-ADAPTIVE N-MODULAR REDUNDANCY TECHNIQUE

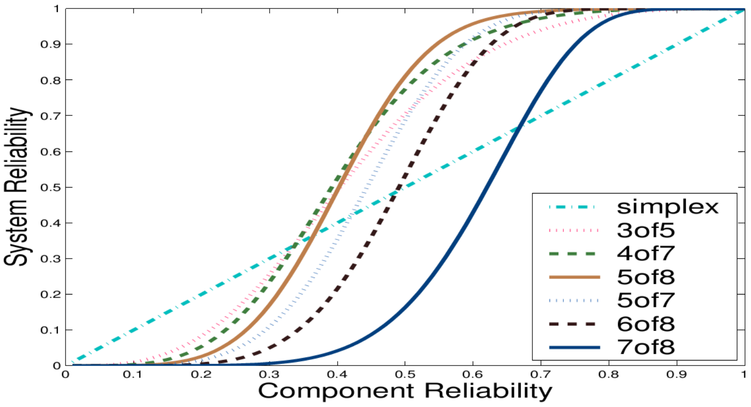
The use of modular redundancies allows to mask the effects of some faulty modules by voting the outputs (comparing them) to know the correct output. For example, in the case of $n = 3$ (TMR) the voter compares the three results of each module. If one of them is faulty, the majority voter selects the output of the two results agree (2-out-3). Moreover, we can increment the number of redundancies to allow more masking of faulty modules. For example, if $n = 5$, the system can mask until three fault modules getting more masking capabilities.

On the other hand, the increment of redundancy modules will increment the area used into the FPGA. From the point of view of available resources, technology trends indicate that each generation of FPGAs, devices offer more and more resources to use. Then, the overhead in resources is each time a minor problem. However, the use of more resources increments the probability of one energized particle heats the design, and consequently, the design is more susceptible to fail. Moreover, the reliability of the components decrease in the time. As demonstrated in (SHOOMAN, 2002), the reliability of a system with a high number of redundancies is high just at the beginning of its operation life, and as the system still working, systems with less redundancy modules are more reliable. This could be explained by the fact that the system reliability decreases exponentially with the time, and then, there are more modules that can cause the fault of the system. One example of the application of nMR is discussed in (SATORI; SLOAN; KUMAR, 2009) where authors propose the use of a fluid nMR computers framework to work with applications with inherent algorithmic error tolerance (property of soft computations to absorb errors in the form of degraded system outputs). Figure 4.1 is coherent with the reliability analysis of n redundancies: the impact in the reliability of the system depends on the number of redundancies and on the reliability of each element (SHOOMAN, 2002).

In the case of SRAM-based FPGAs exposed to radiation, the accumulated faults increment the possibility to have a faulty module. Then, it is expected that in the beginning it is better to use the highest as possible number of redundancies, but that number must to be reduced in the time according to the reliability feature of each element.

Our proposal is based on the possibility to use an self-adaptive nMR capable to change the voting policy as the modules fail. Figure 4.2 shows the probability to have a correct output of an nMR system (reliability) for different policy voting. For example, in an 11MR system the voting policy is 6-out-11 correct module outputs, in a 9MR system the voting policy is 5-out-9 correct module outputs, in a 7MR system the voting policy is 4-out-7 correct module outputs, in a 5MR system the voting policy is 3-out-5 correct module outputs, and finally in a 3MR (TMR) system the voting policy is 2-out-3 correct

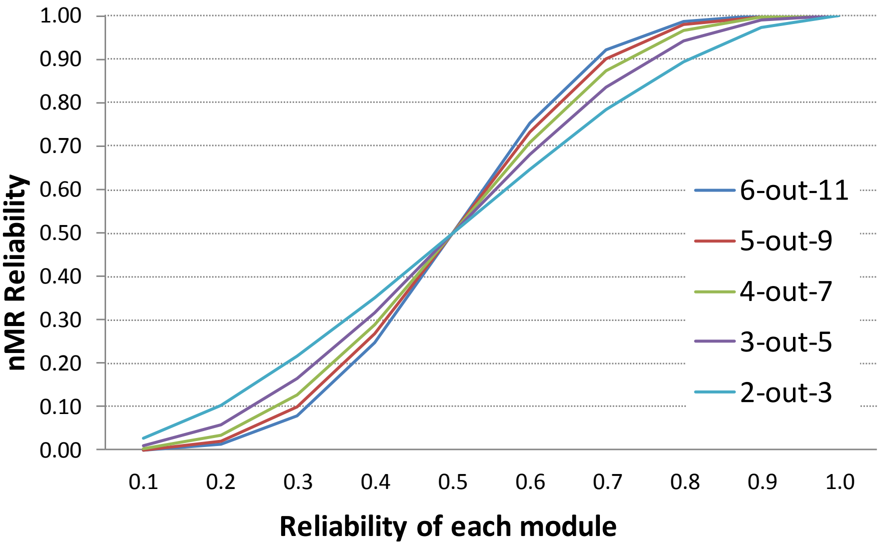
Figure 4.1: Reliability characteristics of nMR depending on the voting policies and the reliability of each elements which can recompute the same operation until 8 times.



(SATORI; SLOAN; KUMAR, 2009)

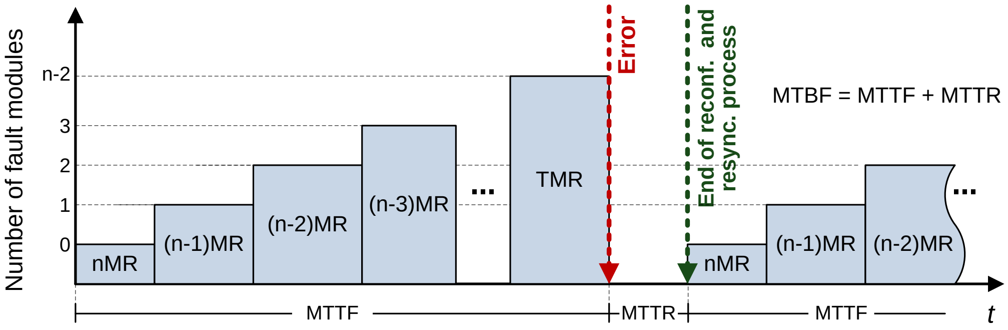
module outputs. In the beginning, considering that the FPGA does not have accumulated upsets, the reliability of each module p is close to 1, so according to the Figure 4.2 it is better to have more number of redundancy modules. In our propose, when one module faults the system is not any more a 11MR, but it is a 10MR. After that if another module fails, the system is a 9MR which follows the policy 5-out-9. The system still working until 2 fault-free modules remain working, in such case the reconfiguration is required.

Figure 4.2: Reliability of m-out-n policy voter according to Equation 1.1.



On the other hand, the possibility to use nMR aims the increase of the MTBF and with this, the reduction of scrubbing rate as shown in Figure 4.3, and consequently the possibility to reduce the power penalties for the use of scrubbing. The main challenge of our propose is the develop of a majority voter able to modify the voting policy according to the number of fault-free modules. In this Chapter we present the architectures of the system and the voter named Self-adaptive voter that allows the change of policy voting according to the number of fault-free modules.

Figure 4.3: MTBF for a self-adaptive nMR system.

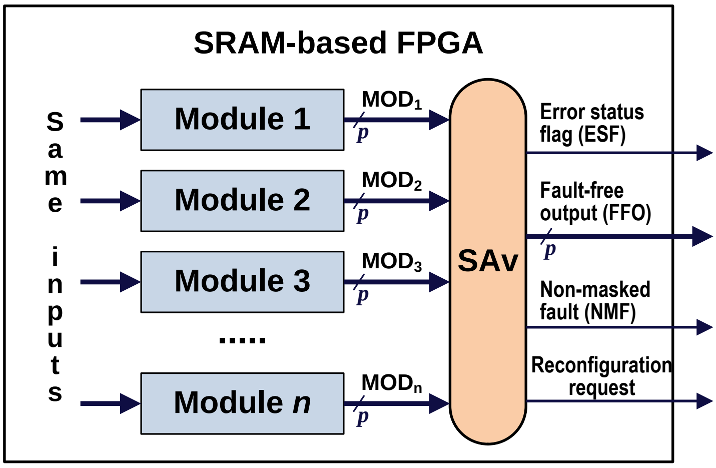


4.1 NMR system architecture proposal

The proposed nMR is composed by n identical modules that receive identical inputs and deliver p-bits output to the Self-Adaptive voter (SAv) as shown in Figure 4.4. The SAv receives $n \times p$ bits from all modules and generates the Fault-Free p-bits output (FFO), n-bits error status flags (ESF), a non-masked fault signal (NMF), and the reconfiguration request. FFO is selected by the SAv depending of the current voting policy. ESF indicates the error flag status of each module. NMF is set when the voter can not decide the correct output due there are an even number of redundancies (n), and two different results as output of the same number of modules. The reconfiguration request is set when it only remains 3 fault-free modules and one module fails. Note that from that time, if an extra module fails the voter will not be able to define the correct output.

The SAv and interconnections path are critical because a single fault in that structure will produce the overall system failure. However, SAv represents a very small area compared to the redundant modules (scalability will be discussed on following sections) and it can also be replicated.

Figure 4.4: Scheme of nMR technique with Self-adaptive voter.

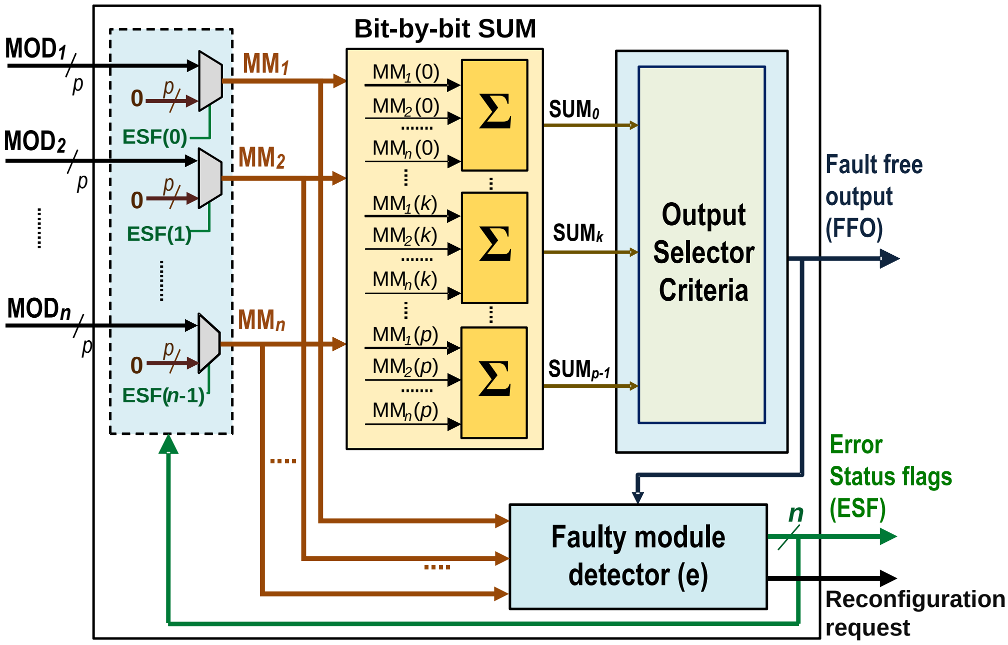


4.2 Self-adaptive Voter

Voter is a critical function in nMR techniques since decides the output value. Reliability of majority voters for computational structures was studied in (HAN et al., 2011). In (SIMEVSKI et al., 2012) a programmable and scalable voter for n redundancies implemented in ASIC is proposed. For TMR designs, a voter with high reliability was presented in (BAN; NAVINER, 2011).

SAv considers as population the output values of each healthy module. As represented in Figure 4.5, the SAv has n inputs (MOD$_{1-n}$) of p bits. Notice that the signal ESF$_i$, $\forall i = 0, 1, ..., n-1$ selects which input will be considered in the vote (MM$_i$). At the beginning, it is assumed that all inputs are coming from healthy modules, so ESF$_i$ = 0 and NMF = 0.

Figure 4.5: Self-adaptive voter.



The voting is realized bit-by-bit in the *Output Selector* block, which considers the sum of each bit of all masked inputs ($\sum_{i=1}^{N} MM_i[k]$, where $k = 0, 1, 0..., p-1$) and the number of fault-free modules. Defining the fault-free output bit as FFO[k] with $k = 0, 1, ..., p-1$, and the number of fault modules as e, each fault-free bit output FFO[k] will be defined as presented in Figure 4.6. Notice that if there is an even number of fault-free modules, it is possible to have the same number of fault-free and fault modules. In that case, could be impossible to select and fault-free output, and consequently NMF is set.

Once the fault-free output defined, its bits are compared in the *fault module detector* against each masked input MM$_i$ and any fault module can be detected and isolated. Notice that the comparison is performed bit by bit, and then if at least one bit of any module does not match with its fault-free value, the module is blocked.

Although an even number of redundant modules may incur in equal and in not electable majority situation, we consider that this is a very uncommon situation since each module commonly has more than one signal as output and the voting is performed signal by signal, then, multiple votes are always performed. However, in case this situation happens, the voter considers as a non-correctable situation and a reconfiguration of the

Figure 4.6: Output Selector criteria.

If $(N-e)$ is odd and $e<N-2$:

$$\text{SUM}_K > \frac{N-e-1}{2} \Rightarrow FFO(K)=1, \qquad \text{NMF=0}$$

$$\text{SUM}_K \leq \frac{N-e-1}{2} \Rightarrow FFO(K)=0, \qquad \text{NMF=0}$$

If $(N-e)$ is even and $e<N-2$:

$$\text{SUM}_K > \frac{N-e}{2} \Rightarrow FFO(K)=1, \qquad \text{NMF=0}$$

$$\text{SUM}_K \leq \frac{N-e}{2} \Rightarrow FFO(K)=0, \qquad \text{NMF=0}$$

$$\text{SUM}_K = \frac{N-e}{2} \Rightarrow FFO(K)=0, \qquad \text{NMF=1}$$

If $e \geq N-2$:

$$FFO(K)=0, \qquad \text{NMF=1}$$

system is needed. In order to guarantee the correct output, the reconfiguration and resynchronization of the system will be requested when only remain two free-fault modules. Figure 4.7 explains the SAv process in a flow diagram starting in the bit-by-bit sum function. Notice that XTMR technique use majority voters in the feedback path of flip-flops to correct and resynchronize the faulty flip-flops. Self-adaptive voter (SAv) is much more complex than a standard TMR majority voter and the inclusion of SAv in the feedpath of flip-flops may be impractical for complex circuits.

As an example, consider a 4MR system with all modules working correctly, where the results of each modules are ‘11001’. Then, $e=0$, $n=4$, $p=5$, ESF=‘00000’, MOD$_1$=‘11001’, MOD$_2$=‘11001’, MOD$_3$=‘11001’, and MOD$_4$=‘11001’. The outputs of bit-by-bit SUM block are: SUM$_0$=4, SUM$_1$=0, SUM$_2$=0, SUM$_3$=4, and SUM$_4$=4. According to the Output Selector Criteria, $(N-e)=4$ is even, and consequently FFO(0)=1, FFO(1)=0, FFO(2)=0, FFO(3)=1 and FFO(4)=1, or FFO= ‘11011’. Finally, as FFO match with all modules output, Faulty module detector block will set $e=0$ and input selectors ESF=’00000’, then for all cases MM=MOD.

On the other hand, if the second module fails and its result is MOD$_2$=‘10101’, the outputs of bit-by-bit SUM block are: SUM$_0$=4, SUM$_1$=0, SUM$_2$=3, SUM$_3$=3, and SUM$_4$=4. One more time, according to the Output Selector Criteria, $(N-e)=4$ is even, and consequently FFO(0)=1, FFO(1)=0, FFO(2)=0, FFO(3)=1 and FFO(4)=1, or FFO= ‘11011’. However, this time FFO does not match with all modules output: faulty module detector block will compare FFO with each module result MM, finding a discrepancy in Module 2: FF0 $\neq$ MM$_2$, consequently $e=1$ and input selectors ESF=’00010’. From now, since ESF$_1$ $=$ ‘1$'$, MM$_2$ $=$ ‘00000$'$, the output of Module 2 is unconsidered, and the voter working just with $n-e=3$ inputs, the system turns in a classical TMR system. Figure 4.8 shows this example.

Figure 4.7: Self-adaptive voter process.

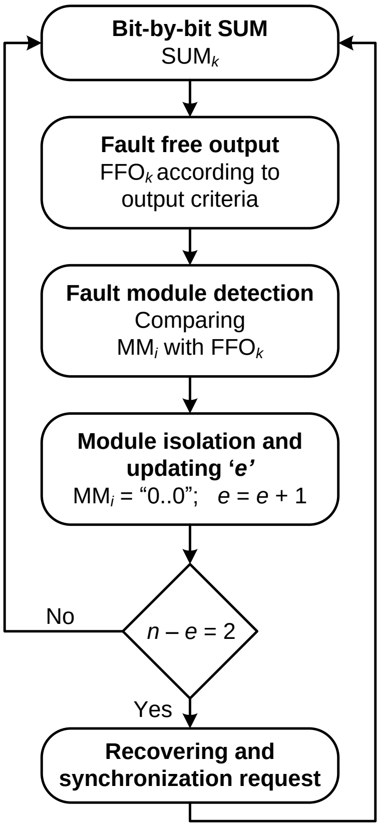
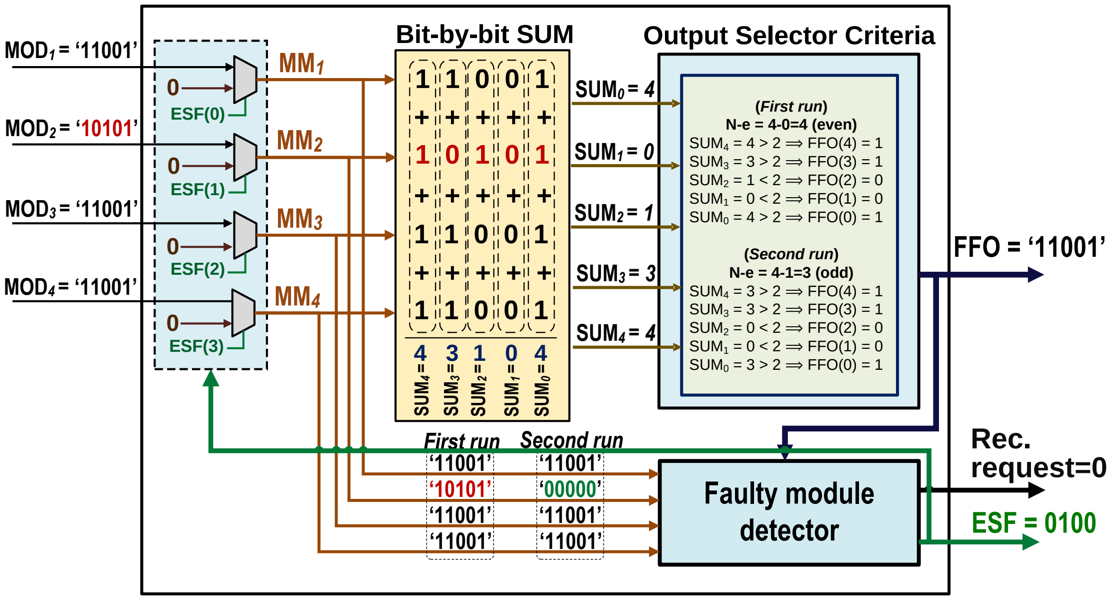

Bit-by-bit SUM
SUM$_k$
Fault free output
FFO$_k$ according to output criteria
Fault module detection
Comparing MM$_i$ with FFO$_k$
Module isolation and updating 'e'
MM$_i$ = "0..0"; $e = e + 1$
$n - e = 2$
No
Yes
Recovering and synchronization request

Figure 4.8: Example of Self-Adaptive voter. First, *n*=4 and Module 2 is fault (first run). The fault is masked and in the second run, *n*=3 (TMR) and second module is not considered in follow votes.



4.3 Scalability of SAv

Since in an nMR system the reliability of the voter is critical, some authors propose the triplication of the voter. However, it is also recommended to use the least amount of resources as possible. The SAv proposed is based on the sum of all input bits, so it is expected that it uses more resources that a standard TMR majority voter and also will depends on the number of input bits.

The Figure 4.9 shows a diagram of the SAv implemented design. We can note that flip-flops are only used in the input and output of the voter, and with this we can know in advance the number of flip-flops needed to implement the voter depending on the number of modules used and output word width. Hence, in a nMR system with n modules of p-bits output, the SAv will use $n \times p$ flip-flops in the input, plus p flip-flops for FFO, plus n flip-flops for ESF and finally one extra register for ENC output which can be used as configuration request signal. The Equation 4.1 defines this value. For example, considering 7 modular redundancies where each module has 8-bits output word, we expect to have $7 \times 8 + 8 + 7 + 1 = 72$ flip-flops.

$$\#Flip-flops = n \times p + p + n + 1 \tag{4.1}$$

On the other hand, the amount of LUTs used in the SAv used not only depend on the number of bits at the input of the SAv, but also the type of LUTs available in the device and the algorithm used by the synthesis tool. As an example, Table 4.1 shows the resources occupation for the SAv in a 7MR system considering different width of outputs. Results were taken from synthesis report of ISE Xilinx Tool considering Virtex-5 LX50T FPGA (XC5VLX50T-1FF1136 device). Notice that LUTs and flip-flops increase exponentially with the number of input bits to be voted.

Figure 4.9: Diagram of SAv implementation.

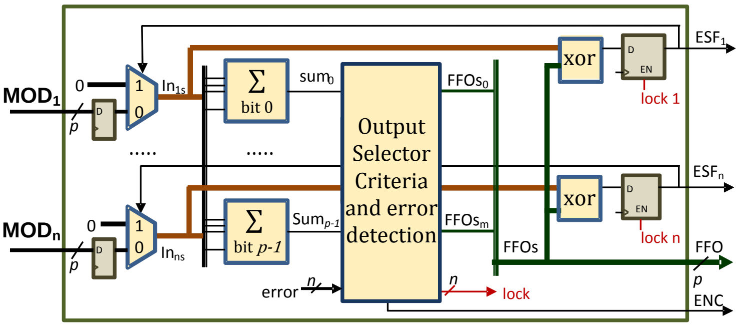


Table 4.1: Relation of SAv occupation for 7MR to the number of bits voted.

Module output width (bits)	Flip-flops		6-LUTs	
	#	%	#	%
8	72	0.33	154	0.70
16	136	0.62	265	1.20
32	264	1.20	514	2.34
64	520	2.36	1007	4.58
128	1032	4.69	1985	9.02

5 PROPOSED FAULT INJECTOR PLATFORM

The proposed multiple fault injector platform helps to emulate SBU and MBU and their accumulation effects in the configuration memory bits of a SRAM-based FPGA quickly, maintaining good control of the experiment and inexpensive compared with radiation experiments. Our goal is to replicate the effects of radiation to validate protection techniques and improve the radiation test methodologies and test plans under accumulated multiple faults.

The main differences of the available platforms (GUZMAN-MIRANDA; TOMBS; AGUIRRE, 2008; STERPONE; VIOLANTE; REZGUI, 2006; VIOLANTE, 2007; NAZAR; CARRO, 2012) and the one presented here is that the proposed platform aims to inject multiple faults in order to repeat neutron radiation test experiments based on the observed and collected flux of particles and bit-flips. In this way, it is possible to verify and test designs in the laboratory before radiation ground testing, having a better prediction of the mitigation technique efficiency to cope with multiple and accumulated faults, and also a validation of the test setup.

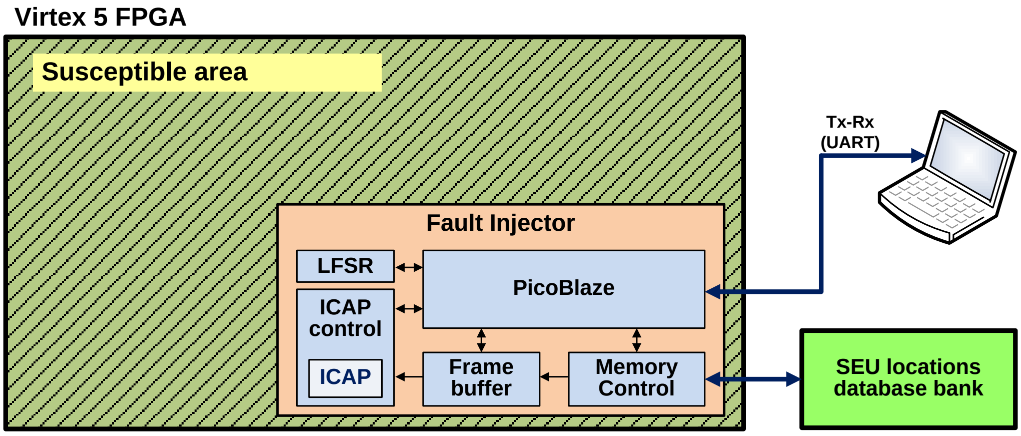
5.1 Fault Injector Architecture

The proposed multiple fault injection platform uses the SRAM-based FPGA Virtex-5 and the internal configuration port ICAP to partial reconfigure the bitstream to inject faults (however, it can be implemented in other Xilinx FPGA that contains ICAP primitive). The ICAP is responsible to access the configuration memory through each frame address. Frames are the smallest addressable segments of the FPGA configuration memory bits and are composed by 41 words of 32 bits (1312 bits) in case of Virtex-5. This approach can also be applied to other Xilinx FPGA that have ICAP.

The configuration bit position to be flipped can be selected through the control block from an in-chip random generator (implemented by a linear feedback shift register - LFSR), or from an SEU location database stored in an external flash memory (if it is available on board). In order to have more realistic results, the SEU database is composed by pre-collected real bitflips location detected from previous neutron accelerated experiments in ISIS facilities to replicate SEUs induced by radiation. Also, customized SEU distribution can be used as SEU location database. Figure 5.1 depicts the injector platform. Bitflip rate can be defined by the tester according to project specification. Fault injector control is implemented by the 8-bit soft-processor Picoblaze, provided by Xilinx (XILINX, 2005). Picoblaze allows the communication to the external PC, as well as controls the LFSR, ICAP control, frame buffer, and memory control blocks.

The injector controller considers two zones in the floorplane: *susceptible area*, where the injector controller can flip any bit, and *forbidden area*, where no bitflip is generated.

Figure 5.1: Architecture of fault injector proposed.



Consequently, the circuit under test must be placed in the susceptible area, and all components of the fault injector (and other in which we do not want to inject faults) must be placed in the forbidden area. Clock lines and connection lines between the control circuit that send the data to the Host PC must be taken into account to avoid faults and consequently the lost of connection of the system. SEUs are injected consecutively one by one until the user needs are achieved, after that, the flipped bit locations are sent to the Host PC. The user can define specific susceptible or forbidden area.

The injector was implemented into XC5VLX50T on Genesys Digilent board and in XC5VLX110T Virtex5 FPGA on ML505 Evaluation Platform board. Synthesis result of the injector controller module is detailed in Table I.

5.2 Modeling MBUs

The injected faults can be modeled mainly in two different approaches:

- By using the randomization based on different distribution models in time and location using a Linear feedback shift register (LFSR),
- By using a radiation database from previous radiation experiments or customized database.

5.2.1 Linear feedback shift register (LFSR)

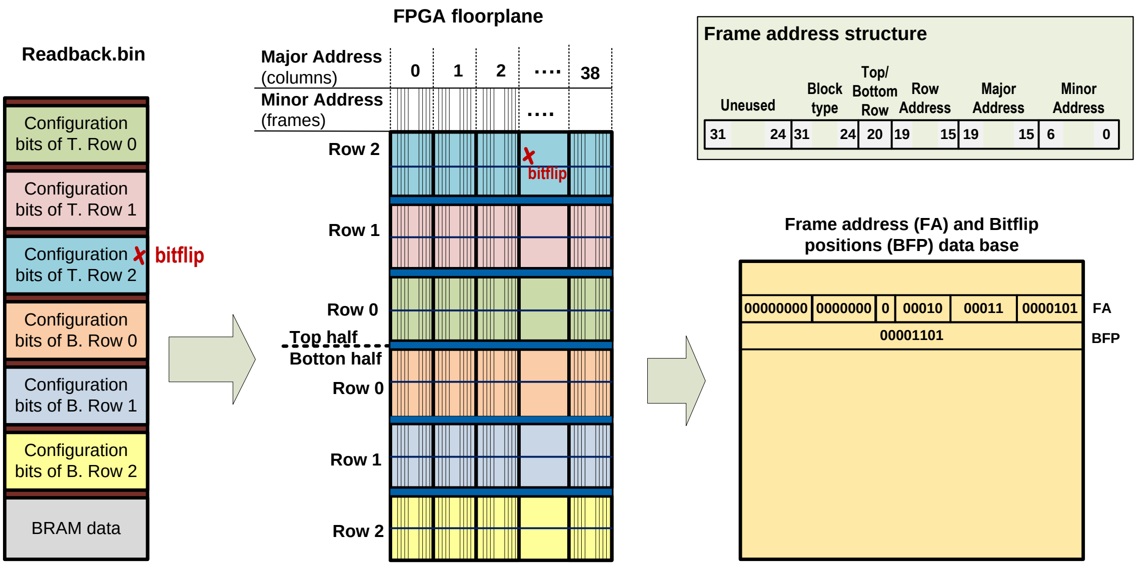
A pseudo-random generator circuit was used aiming at supplying random addresses to the injection control and then tuned to simulate the behavior of a real neutron test. Several LFSR structures and seeds were implemented and tested to generate a good random dispersion and to obtain the effects nearest to the produced ones for the radiation experiments. The selected one is a 25-bits LSFR and is based on the frame address structure (XILINX, 2012c). Each frame address is divided into 5 main parts: type (4 bits, in our case always ”0001”), top/bottom (1 bit), row (5 bits), major address (8 bits) and minor address (7 bits). Then, the LFSR is composed by 4 sub-groups of smaller LFSRs . The selection of the bit position inside the frame is selected by the 11 first bits of the LFSR. The forbidden frame addresses (frames of injector platform or defined by the user) and nonexistent frame addresses are filtered by the injector control.

5.2.2 SEU Location Database

A database is composed of multiple and accumulated faults in Virtex-5 FPGA built from radiation experiments or by customized bitflip. The database has the radiation data of two Virtex-5 devices: XC5VLX50T and XC5VLX110T irradiated with a neutron spectrum that resemble the atmospheric one in the ISIS facilities of Rutherford Appleton Laboratory (Didcot, United Kingdom). The flux was about $4.3x10^4 neutrons/s/cm^2$.

Based on our knowledge of the FPGA bitstream, we can precisely determine the frame address and bit position of each SEU registered during the experiment as shown in Figure 5.2. The readback file (Readback.bin) obtained during the radiation experiment contains the configuration bit values at the moment of the readback. When compared with a ”golden” readback (before radiation experiment) and considering the mask file (which indicates the position of dynamic configuration bits) we can locate the bitflipped in the configuration memory. Since the readback is related to the floorplane, we also can locate the position of the bit-flip on the FPGA floorplan. Finally, using the frame address structure available in (XILINX, 2012c) we extract the frame address and bit position of the bit-flip. These informations are necessary to write into the configuration memory through the ICAP.

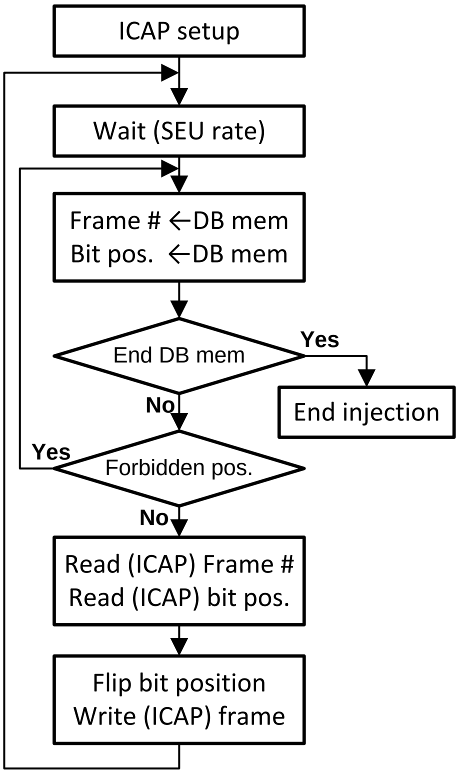
Figure 5.2: Getting the bitflip locations in Virtex-5 FPGAs.



In our neutrons experiments experiments more than 1,000 SEUs in the configuration memory were identified. This information is stored in the platform in a external flash memory. In the case of the Genesys board, it has a flash memory of 256 Mbit (organized as 16-bit by 16MBytes) for non-volatile storage of FPGA configuration files. We used three memory addresses to store the information of each SEU. The first two positions store the frame address and the last position store the bit position inside the frame. So, up to 5 million SEUs can be stored in this memory.

Figure 5.3 shows the flow diagram of the fault injector. The user configures the SEU rate, the memory position of the first SEU location (number of frame and bit position) and the number of faults to inject. Then, it is checked if the SEU location belongs to the forbidden position (by default it is the region where the fault injector is implemented) to read a new bit position from the data base memory (DB memory). In order to generate the bitflip, the entire frame is read and stored, the bit position is flipped and the entire frame is write again into the memory configuration.

Figure 5.3: Flow diagram of the proposed fault injector.



5.3 Fault Injection Campaign Results and comparisons

Since we are interested in repeating the effects of radiation effects in SRAM-based FPGAs, we analyze the SEUs distribution and its effects in some circuit under test exposed to neutron irradiation. The used FPGAs were XC5VLX110T on ML505 board, and XC5VLX50T on Genesys board. The fault injector uses 687 LUTs, 289 flip-flops and 2 BRAMs which represent 2.4%, 1% and 3.3% of the LUTs, flip-flops and BRAMs available in a XC5VLX50T FPGA.

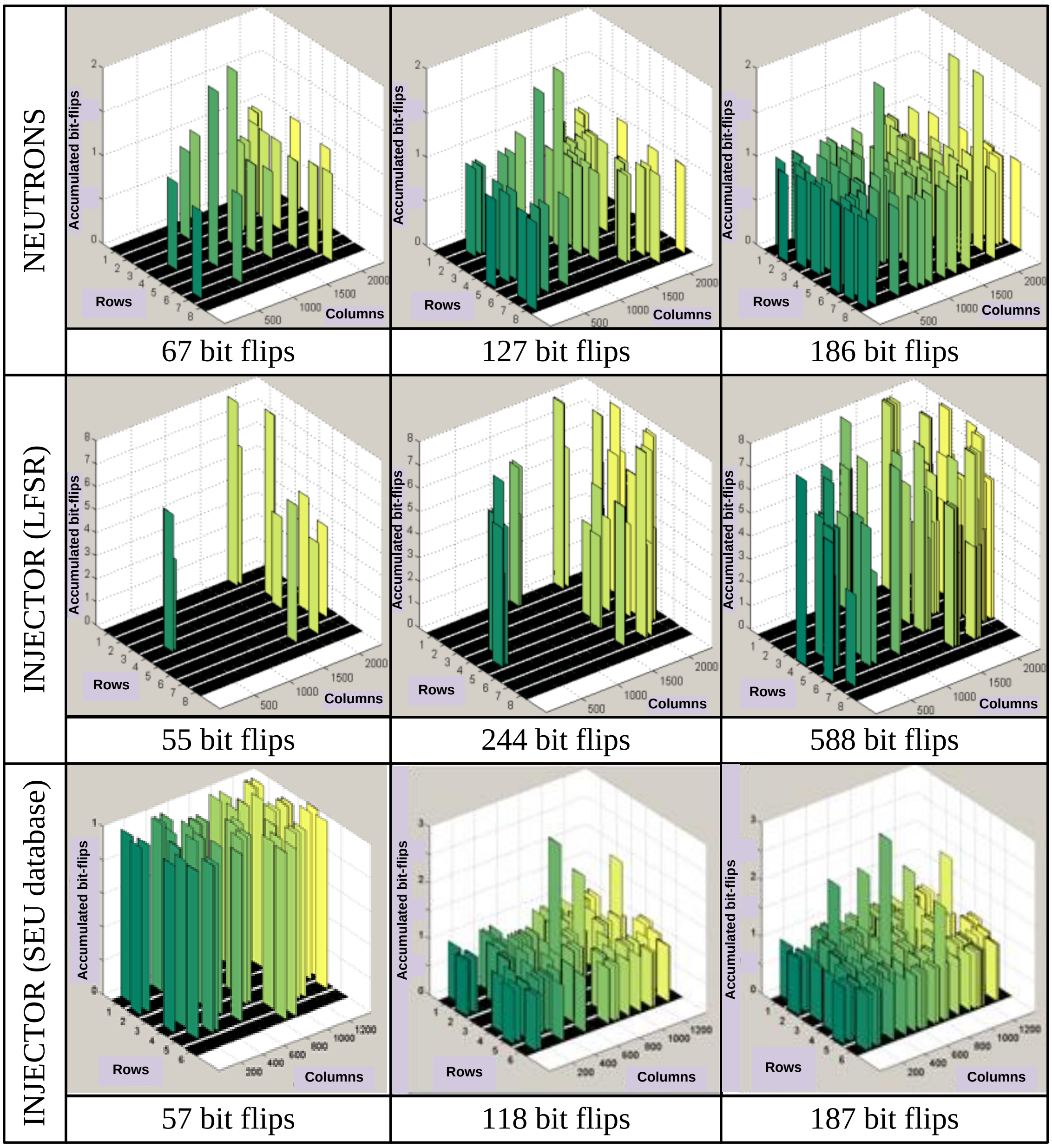
First, we compared the bit-flips distribution generated by both LFSR and by the fault injector platform with the SEUs induced by the energized neutrons. Then, we compare the masking capabilities of a DTMR technique as case study using the proposed fault injector and radiation experiments.

5.3.1 MBU Distribution Analysis in Time and Location

In order to verify the capability of the fault injector to replay the location of the bit-flips induced by radiation, we plotted and compared in Figure 5.4 two different bit-flip distributions generated by the fault injector with one generated by neutron radiation experiments. In the figure, NEUTRONS distributions bars represent the bit-flips distribution generated by radiation experiments, INJECTOR (LFSR) were generated by the LFSR of the injector, and finally, INJECTOR (SEU database) distribution were generated by the random function of Matlab and stored into the SEU locations database bank.

Each bar in the plots represents the number of accumulated SEUs per frame in configuration bits (no BRAMs data are considered). The total number of accumulated SEUs (bitflips) for each plot is also shown. Neutrons experiments commonly show one bitflip by frame, when the injector using LSFR show values between 4 and 8 per frame. On the other hand, the results from the SEU database are similar to the neutrons results as expected.

Figure 5.4: Comparing injected faults distribution. SEU data base is composed by random bitflip positions generated by Matlab.



5.3.2 Comparison between Fault Injection using the LFSR and Neutron Test

In order to verify the capability of the LSFR to mimic the effects of SEUs induced by radiation, we compared the fault tolerant capabilities of a circuit protected by diverse triple modular redundant modules (DTMR) and by TMR, by means of neutron radiation and fault injection (TAMBARA L.; RECH, 2013). In the DTMR, each redundant copy is implemented in a distinct way using different replicas and algorithms. The case study circuit presented in (TAMBARA L.; RECH, 2013) was an 8x8 matrix multiplication operation, implemented in its DTMR version by a finite state machine (FSM), a combinational circuit, and by a software version running in a miniMIPS processor. In the case of standard TMR, the matrix multiplication was performed by three miniMIPS processor. All circuits were implemented prototyped in XC5VLX110T FPGA.

Results are compared and shown in Table 5.1. In the case of neutron experiments, DTMR needs 190 accumulated SEUs to have an error against 86 for the case of TMR, then, it is necessary 2.21x times more SEUs. In the case of injector results using LSFR, DTMR needs 391 accumulated faults to have an error against 158 of TMR scheme. Although it is necessary almost the double of faults in fault injection to have an error in the design compared to the neutron test, the proportion between the designs (DTMR and TMR) is 2.47x times, which it is almost the same from the results of the neutron experiments. When comparing both results, the error is about 12.01%. The difference comes from the difficulty on modeling the randomization by using the LFSR once multiple faults are injected in the same frame compared to the radiation experiment results.

Results show that faults injected using the LSFR circuit induced similar effects in circuit under test when compared to radiation effects. However, the number of accumulated upsets is different due the random capability of LSFR circuit.

Table 5.1: Comparison of radiation and fault injection experiments for DTMR-MIPS.

	Neutron Experiment			**Fault Injector using LSFR**			
	TMR	**DTMR**	**Increase factor**	**TMR**	**DTMR**	**Increase factor**	**Error (%)**
# Accumulated SEUs to provoke an error (average)	86	190	2.21	158	391	2.47	12.01
# Runs	26	23		500	500		
Time (aprox.) in minutes	634	634		8	8		

6 POWER ANALYSIS IN NMR SYSTEMS IN SRAM-BASED FPGAS

The nMR technique has been used at design and system level to cope with multiple faults. However, the main drawback is the extensive use of resources overhead, such as area and power. In application-specific integrated circuit (ASIC), all resources are carefully projected to implement a target design. Consequently, the amount of transistors in the circuit is optimized to each particular design and the power consumption is specific for that particular ASIC with a determined static and a dynamic part. Therefore, in case of an ASIC, the replication of a design will have a high impact on the power overhead. On the other hand, FPGAs are designed to have a suitable size configurable matrix that can fit many types of designs projected by the user. So, the amount of transistors of a FPGA is the same for all implemented designs, and the static power consumption is almost independently to the implemented design (KUON; ROSE, 2007). Moreover, despite being used 100% of logic blocks and user flip-flops, about 35% of the static power is dissipated in the unused transistors of unused interconnect switches (TUAN; LAI, 2003). The dynamic power of the customized design is the one that plays the main difference among designs but it represents a small overhead in the majority of the cases. So, for FPGAs, the use of nMR technique does not imply necessarily into n times increase in power, as it is observed in ASICs. As it will be present, in many cases the use of nMR in FPGAs implies in only 1.57 times higher power dissipation, while providing a high making effect capability.

In this chapter, we present a generic model to estimate the power penalty in nMR designs synthesized into SRAM-based FPGA. The goal is to use the model to help to predict in early stages of the design process the power overhead when using nMR. The target FPGA family in this section is Virtex-5 from Xilinx (XILINX, 2009b), but this work can be extended to other families of the same fabricant. We discuss the proposal model in terms of number of redundancies (n) in the nMR technique, the relation between static and dynamic power (r) and the size of the FPGA matrix. Then, we provide a power consumption analysis of a corner case circuit, one synthetic circuit (chain of adders), and a microprocessor (running a matrix multiplication application) using nMR, where n varies from 3 to 7. All implemented designs were synthesized into different sizes of Virtex-5 SRAM-based FPGAs. Comparisons between the power consumption estimated by XPower tool and the model are presented. The obtained results and the proposed model are very important and innovative because they point out the main differences when estimating power in fault tolerant designs in FPGAs compared to ASICs. The model can guide designers to predict the impact of a design protected by nMR in SRAM-based FPGAs. And the low overhead power results may impulse designers to use more often nMR in high reliability applications when using SRAM-based FPGAs.

6.1 Modeling power consumption in SRAM-based FPGAs

Total power consumption is composed by static power P_{STAT} and dynamic power P_{DYN} defined by Equation 6.1.

$$P_{TOT} = P_{STAT} + P_{DYN} \tag{6.1}$$

In CMOS devices, the static power is linearly related to the voltage level (V_{CC}), and to the leakage current of the device (I_{CC}), as defined in Equation 6.2. The leakage current of the device is the sum of the transistor leakage currents, which depends of the voltage and operational temperature of the transistor.

$$P_{STAT} = V_{CC} \times I_{CC} \tag{6.2}$$

On the other hand, the dynamic power is related to the switching activity of transistors, and the capacitance and voltage level that powers the device, as defined in the Equation 6.3. Notice that if all transistors are powered with the same voltage level V_{CC} and the same frequency, the Equation 6.3 can also be written as Equation 6.4

$$P_{DYN} = \sum_{i=1}^{n} \alpha_i C_i f V_{CC}^2 \tag{6.3}$$

Where:
n= number of toggling nodes
α_i = switching activity
C_i = load capacitance of the node i
f = clock frequency
V_{CC} = transistor source voltage

$$P_{DYN} = f V_{CC}^2 \sum_{i=1}^{n} \alpha_i C_i \tag{6.4}$$

Both equations 6.2 and 6.4 are valid for designs implemented as ASIC or into FPGAs. However, the total power consumption of a design depends on the specific design characteristics of target circuit. In ASIC, the number of transistors is optimized for area and performance and interconnections are implemented directly by metal traces. Consequently, the static power consumption is designed to be as minimum as possible, and the dynamic power is the main contributor for the total power consumption. On the other hand, SRAM-based FPGA devices are composed by fix number of transistors, which comprise the arrangement of logical blocks, configurable interconnects and special blocks as internal RAMs and DSP modules. These elements are the key of the versatility, which is the main feature of the SRAM-based FPGA, but also all these resources are the cause of extra static power consumption.

As it is well known, the same design implemented in ASIC and into a FPGA using the same process technology will has much less power consumption when implemented as ASIC (KUON; ROSE, 2007). Moreover, it is expected that in ASIC implementations, the power overhead caused by the use of redundant modules to be increased in the same factor of the number of redundancies. In case of FPGAs, this proportion may not be true due to the fact that the static power play an important task in the total power consumption.

In order to minimize static power in FPGA, vendors offer devices with different number of configurable resources for every family. For example, in the case of Virtex-5 LXT, the number of slices (each one contains 4 LUTs and 4 flip-flops) for LX20T, LX30T, LX50T, LX85T, LX110T, LX155T, LX220T, LX330T are 3120, 4800, 7200, 12960, 17280, 24320, 24560 and 51840 respectively (XILINX, 2009b). In addition, to have a better optimization of power consumption, FPGAs use diverse supply voltage lines for powering their internal components (XILINX, 2010a) as presented in the Table 6.1.

Table 6.1: Maximum and recommended voltage levels in supply voltage lines of Virtex-5 FPGA (65 nm).

Symbol	Description	Absolute maximum voltages (V)	Recommeneded operating volages (V)
V_{CCINT}	Internal supply voltage relative to GND	-0.5 to 1.1	0.95 to 1.05
V_{CCAUX}	Auxiliary supply voltage relative to GND	-0.5 to 3.0	2.375 to 2.625
V_{CCO}	Output drivers supply voltage relative to GND	-0.5 to 3.75	1.14 to 3.45
V_{BATT}	Key memory battery backup supply	-0.5 to 4.05	1.0 to 3.6

(XILINX, 2010a)

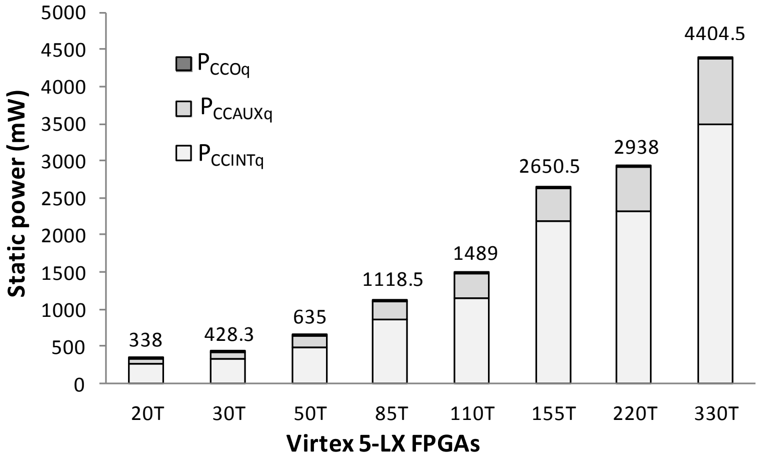
In order to determine the static power of a FPGA device, it is possible to calculate it by multiplying the typical quiescent supply current at 85° junction temperature (T_j) with the correspondent voltage supply (XILINX, 2010a). In order to determine the total power consumption, a tool provided by Xilinx called XPower can be used. It considers the current and power consumption for each voltage line, since different FPGA families have multi voltage power line for internal core, input/output pins, and other elements. XPower is an accurate power estimation tool because it relies in the libraries from the vendor with specific technology and fabric information used in the target FPGA. Static power results are presented in Figure 6.1, where P_{CCINTq}, P_{CCAUXq} and P_{CCOq} are the static power consumption in lines V_{CCINT}, V_{CCAUX} and V_{CCO} respectively. Note that the size of the device impacts drastically the static power consumption P_{CCINTq} that powers the internal configurable elements.

6.1.1 Power considerations for nMR FPGA implementation

Since all the transistors of the FPGA are turned on independently to the design synthesized into the configurable matrix, it is expected that the static power (P_{STAT}) is almost constant when compared to the total power consumed. On the other hand, dynamic power consumption (P_{DYN}) depends on the characteristics of the designed circuit and operating frequency. Then, we define the power consumption of the original module as: $P_1 = P_{STAT} + P_{DYN}$.

In order to have an estimative of the power penalties, we assume that the use of n redundancies will only impact in the dynamic power component. Each original module

Figure 6.1: Typical static power consumption for LX Virtex-5 FPGAs by supply line calculated from the typical quiescent supply current values at 85°C T_j according to (XILINX, 2010a) and XPOWER tool.



is composed by its function logic block, input and output ports. However, to implement the modular redundancy, we can only replicate the logic function block and maintain the original input and output ports, or replicating the entire module. In the last case, n logic modules are obtained, n input ports and n output ports.

Hence, the total power consumed by the nMR circuit (P_n) when inputs and outputs are replicated can be approximated defined by Equation 6.5. Note that we are not considering the impact of the power consumption of the voter, since ideally, the voter is very small compared to the redundant module.

$$P_{n-all} = P_{STAT} + n \cdot P_{DYN} \tag{6.5}$$

Consequently, the power overhead (P_{OV-all}) can be defined by:

$$P_{OV-all} = \frac{P_{n-all}}{P_1} = \frac{P_{STAT} + n \cdot P_{DYN}}{P_{STAT} + P_{DYN}} \tag{6.6}$$

Note that the corners of P_{OV-all} are determined by the relation between dynamic and static power consumption, as shown:

- If $P_{STAT} >> P_{DYN} \Rightarrow P_{OV-all} \approx 1$
- If $P_{STAT} << P_{DYN} \Rightarrow P_{OV-all} \approx n$

Then, $P_{OV-all} \in]1, n[$

Moreover, considering r as the rate between dynamic and static power of the original module, the Equation 6.6 can be rewritten as:

$$P_{OV-all} = \frac{P_{OV-all}}{P_1} = \frac{n \cdot r + 1}{r + 1} \tag{6.7}$$

Where $r = \frac{P_{DYN}}{P_{STAT}}$, and P_{DYN} and P_{STAT} correspond to the original module.

Following the same logic, we can model the expected overhead power of nMR when only the functional logic block is replicated. In such case we must to subtract the power consumed by the replicated input and outputs ports. Considering the dynamic power consumed in each input/output pin as P_{DYN-IO}, we can model the power overhead of a system that replicates only the functional logic block P_{OV-flb} with:

$$P_{OV-flb} = \frac{n\dot{r} + 1 - (n-1) \cdot P_{DYN-IO}/P_{STAT}}{r+1} \tag{6.8}$$

We can also rewrite the Equation 6.8 as a function of P_{OV-all}

$$P_{OV-flb} = P_{OV-all} - \frac{(n-1)}{r+1} \cdot P_{DYN-IO}/P_{STAT} \tag{6.9}$$

Hence, the power overhead of a nMR system which replicates all input and outputs P_{OV-all} can be predicted by the Equation 6.7, and by the Equation 6.9 when only the internal logic blocks P_{OV-flb} are replicated . Both equations are based on the number of redundancies, and the dynamic and static power rate characteristics of the original module.

However, the number of modular redundancies is limited by the amount of available resources into the target FPGA. Hence, designers may have two different project scenarios: when the original FPGA has enough available sources to implement n redundant modules and when it does not and a larger FPGA device of the family must be used.

6.1.1.1 Option 1: target FPGA is capable to implement the nMR technique

In this case, the FPGA part is the same independently of the number of the redundant modules selected, consequently the P_{STAT} is almost constant for all n cases. The power overhead model presented in Equation 6.7. is plotted in Figure 6.2, for 4 different values of r (ratio between dynamic and static power) and for n redundant modules. Notice that for designs with $r < 0.5$ ($P_{DYN} < 0.5P_{STAT}$), the power overhead is very low: for example, for 11 redundancy modules and $r = 0.5$, the expected overhead $P_{11}/P_1 = 4.33$ times larger. Such overhead is considerable very much lower than in the case of an ASIC implementation, when nMR with 11 redundant modules would present an expected overhead in power consumption of approximately 11 times larger power.

6.1.1.2 Option 2: target FPGA is not capable to implement the nMR technique

If the resources required to implement more redundant modules are not available in the original target FPGA device, a larger FPGA must be selected to fit the n redundancies. In such case, r will be different according to the FPGA selected. Considering FPGAs belonging to the same family product, the main difference will be the number of configurable logics available in the device, and consequently, P_{STAT} will be greater for larger FPGAs. Since r is equal to P_{DYN}/P_{STAT}, it is expected that the power overhead will increase more smoothly as presented in Figure 6.3.

6.2 Estimating power in case-study circuits implemented in SRAM-based FPGA

In order to analyze the power overhead in nMR designs and compare it with the proposed model, we estimate the dynamic and static power consumption using XPower

Figure 6.2: nMR power overhead penalties as function of the number of redundant modules n, and the ratio r between dynamic and static power considering the Equation 6.7.

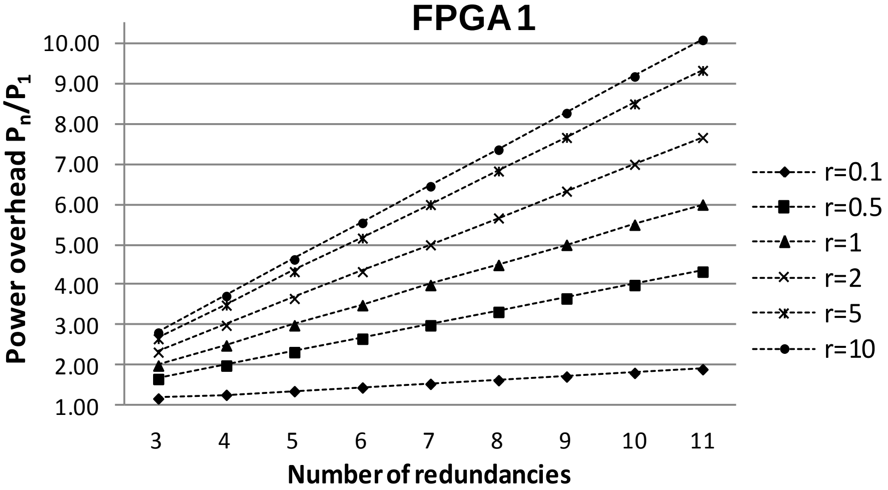
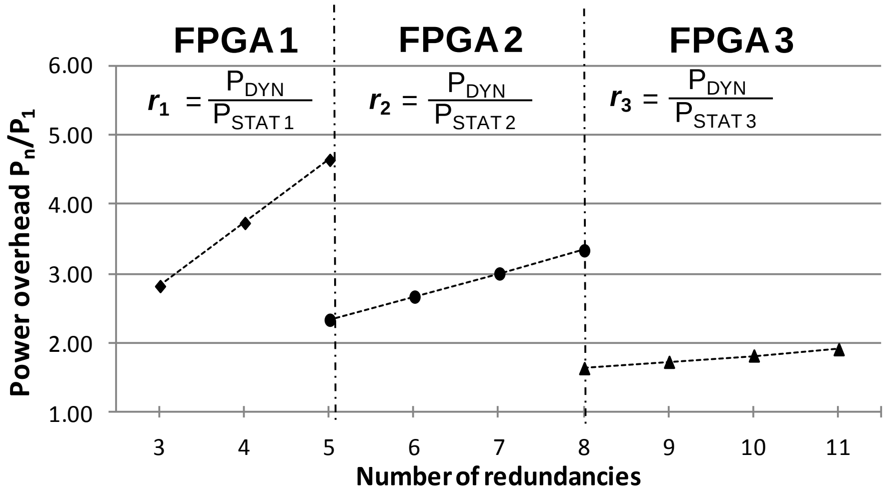


Figure 6.3: Example of different expected power overheads depending on the target FPGA device capable of implementing the selected nMR considering the Equation 6.7. Since $size_{FPGA1} > size_{FPGA2} > size_{FPGA3}$, then $P_{STAT1} > P_{STAT2} > P_{STAT3}$, and $r_1 < r_2 < r_3$.



Xilinx tool (XILINX, 2011b) for two case study circuits synthesized into Virtex-5 family FPGAs (XILINX, 2009b). The first case study circuit is a miniMIPS soft-processor (HANGOUT; JAN, 2009) running a 6x6 matrix multiplication. The last one is a chain of adders implemented by only LUTs and flip-flop slices (no DSP blocks are considered). Although it does not represent a typical application circuit, this circuit allows the exploration of corner case due its high switch activity representing a very high r.

6.2.1 Case-study circuit 1: miniMIPS

MiniMIPS is a soft-core version of MIPS 32-bit microprocessor. The nMR system was implemented in 4 different versions: $n = 1$ (the original module), $n = 3$, $n = 5$ and $n = 7$, where each miniMIPS runs a 6x6 matrix multiplication algorithm and results are delivered in 12 bits. The system uses the SAv as voter as shown in Figure 6.4.

Figure 6.4: Diagram of 7MR 16-bit adders for power test.

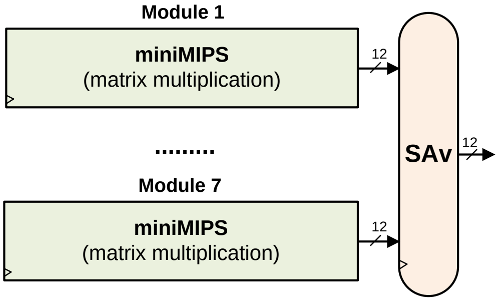


Table 6.2 shows the synthesis results for Virtex-5 LX50T, Virtex-5 LX30T and Virtex-5 LX20T FPGA in terms of occupation resources. As shown, if we are looking for the smallest device of Virtex-5 LX family, Virtex-5 LX20T can only be implemented tree modular redundancies. If we need to use 4MR system, the smallest FPGA is Virtex-5 LX30T. If we have a Virtex-5 LX50T, it is possible to implement until 7 redundancies (7MR). The SAv voter uses 0.30% and 0.21% of available LUTs and flip-flops in a Virtex-5 LX50T. These values are very small compared with the size of the original module.

Table 6.2: Resources used by miniMIPS-nMR in three Virtex-5 devices.

	Virtex-5 LX50T			**Virtex-5 LX30T**			**Virtex-5 LX20T**		
n	LUTs (%)	Reg. (%)	BRAM (%)	LUTs (%)	Reg. (%)	BRAM (%)	LUTs (%)	Reg. (%)	BRAM (%)
1	12.18	5.21	5	18.27	7.81	8.3	28.10	12.02	5
3	34.17	15.83	15	51.26	23.75	25	79.53	36.54	15
4	–	–	–	68.36	31.63	33.3	–	–	–
5	56.88	26.34	25	–	–	–	–	–	–
7	79.76	36.85	35	–	–	–	–	–	–

Figure 6.5 shows the dynamic and static power distribution for each case obtained from XPower tool. Notice that static power is constant for all the cases as the FPGA has the same size for all nMR and frequencies, while the dynamic power increases with the number of redundant modules *n* and the frequency.

Considering the Option 1, we analyze the effect of power consumption in the nMR designs of miniMIPS. For our analyzes propose, we present in Table 6.3 total power consumed for the processor running at 25 Mhz, 33 Mhz, 50 Mhz and 66 Mhz estimated by the XPower, the *r* obtained using the XPower results, the power overhead P_{OV-flb} obtained by XPower and by the model defined in Equation 6.9, and the error of the model proposed respect to XPower results. We highlight that *r* values are far lower than 1, and consequently we expect that power overhead will be low as shown in Figure 6.2. According to Tables 6.3, the highest overhead obtained by XPower is 1.57 times the higher power of the original module, for the 7MR working at 66 Mhz (r = 0.217). As shown, the overhead obtained from the Equation 6.9 is very close to results obtained from XPower tool. Notice that the maximum error is 6.54% for $f = 66\ Mhz$ and n=7, and lower errors are obtained for lower *r* and *n* values. Results of power overhead obtained from XPower tool and the model proposed in Equation 6.9 are plotted in Figure 6.6.

Figure 6.5: Measured static and dynamic power using XPower of a miniMIPS processor implemented using three different nMR (*n*=3, *n*=5 and *n*=7) synthesized into the same XC5VLX50T FPGA.

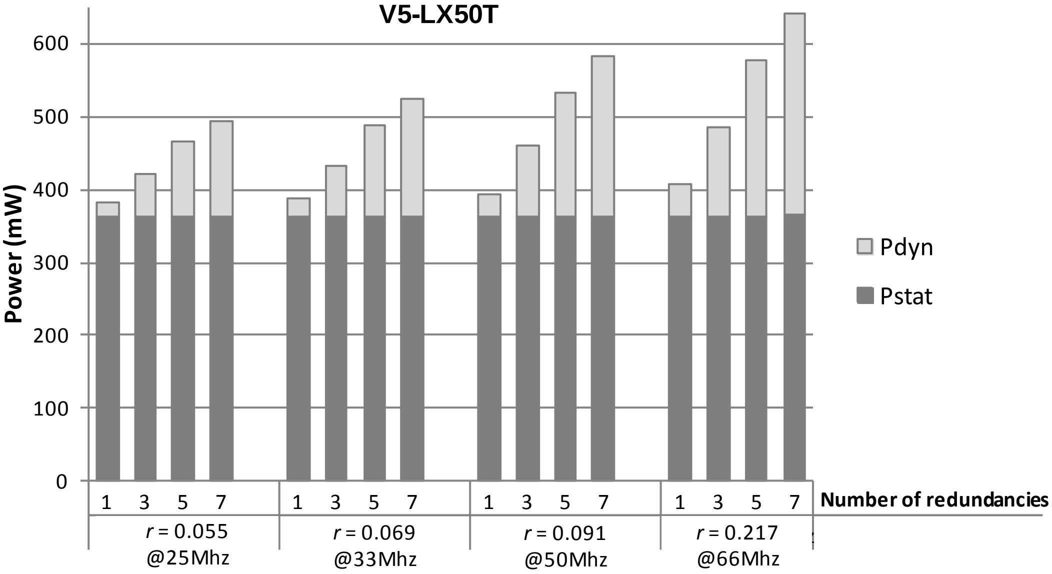
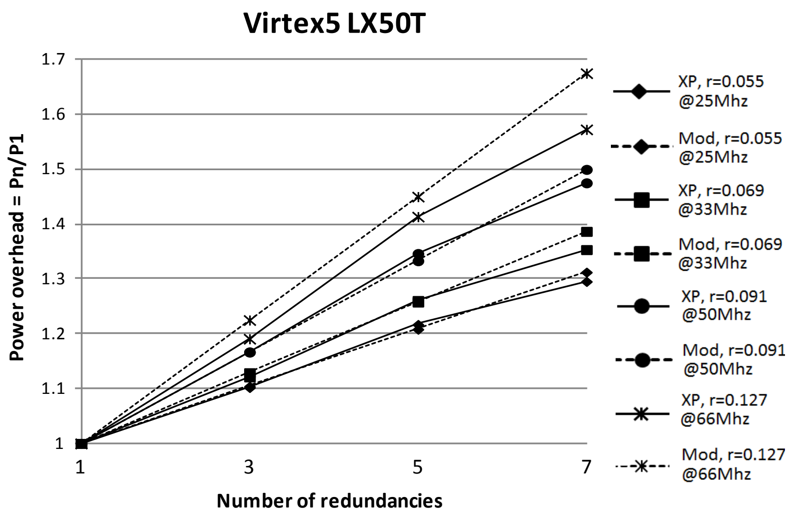


Table 6.3: Power consumption estimated by XPower and by the model proposed in the Equation 6.9 for the miniMIPS-nMR running at 25Mhz, 33Mhz, 50Mhz and 66Mhz in XC5VLX50T FPGA.

	25 Mhz				**33 Mhz**			
	XPower		Equation 6.9		XPower		Equation 6.9	
n	P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error (%)	P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error (%)
1	382	1	1	0	387	1	1	0
3	421	1.10	1.11	0.24	434	1.12	1.13	0.69
5	465	1.22	1.21	0.65	488	1.26	1.26	0.20
7	495	1.30	1.31	1.41	524	1.35	1.39	2.48
r	0.055		–	–	0.069		–	–

	50 Mhz				**66 Mhz**			
	XPower		Equation 6.9		XPower		Equation 6.9	
n	P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error (%)	P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error (%)
1	395	1	1	0	408	1	1	0
3	461	1.17	1.17	0	486	1.19	1.23	2.88
5	532	1.35	1.33	0.94	577	1.41	1.45	2.60
7	583	1.48	1.50	1.72	642	1.57	1.68	6.54
r	0.091		–	–	0.127		–	–

Figure 6.6: Power overhead of nMR of miniMIPS obtained by XPower (XP) and by the proposed model from Equation 6.9 for XC5VLX50T FPGA.



Now, considering the Option 2, we analyze the effect of power in the nMR designs of the miniMIPS when the target FPGA is not capable to implement the selected nMR cases and a larger FPGA is selected. Aiming the use of the maximum resources in each device, the FPGAs selected were V5LX20T, V5LX30T and V5LX20T. Similar to previous case, Figure 6.7 shows the power distribution for all nMR circuits implemented. Note that in this case, the static power is not constant for all nMR as the FPGA device changes and *n* increases, but we can observe that the main contribution of the power comes also from the static power.

Figure 6.7: Measured Static and Dynamic Power using XPower of a miniMIPS processor implemented using three different nMR synthesized into the three different FPGAs (XC5VLX20T, XC5VLX30T, XC5VLX50T).

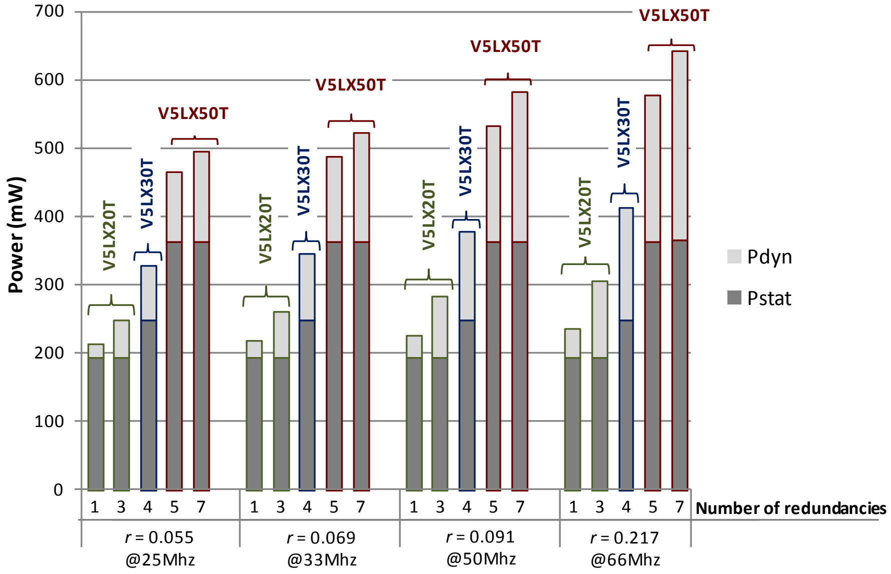


Table 6.2 shows the resources used by nMR implementation for n = 3, 4, 5 and 7, and their power characteristics in Table 6.4. The highest power overhead obtained by XPower is 1.42 times the higher power of the original module, for the 7MR working at 66 Mhz (r=0.178). As expected in Figure 6.3, larger FPGAs have lower r values, and consequently the power overhead increases smoothly. About the error, notice that Equation 6.9 is pessimistic for all cases. According to the results, the maximum error is always lower than 5%. Figure 6.8 shows the power overhead obtained from XPower tool for all implemented circuits in three selected devices.

Table 6.4: Power consumption estimated by XPower and by the model proposed in the Equation 6.9 for the miniMIPS-nMR running at 25Mhz and 33Mhz, 50Mhz and 66Mhz in XC5VLX30T and XC5VLX20T FPGAs (Option 2).

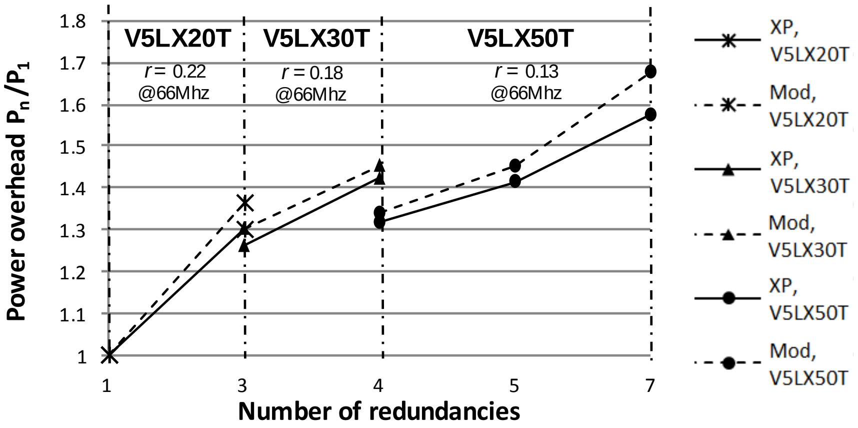
V5LX	***n***	**25 Mhz**				**33 Mhz**			
		XPower		Equation 6.9		XPower		Equation 6.9	
		P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error (%)	P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error
30T	1	268	1	1	0	272	1	1	0
	3	307	1.15	1.16	-0.98	319	1.17	1.18	-0.94
	4	329	1.23	1.24	-0.61	346	1.27	1.28	-0.29
	r	0.085		–	–	0.101		–	–
20T	1	213	1	1	0	218	1	1	0
	3	248	1.16	1.19	-2.02	260	1.19	1.23	-3.08
	r	0.104		–	–	0.130		–	–

V5LX	***n***	**50 Mhz**				**66 Mhz**			
		XPower		Equation 6.9		XPower		Equation 6.9	
		P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error (%)	P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error
30T	1	281	1	1	0	292	1	1	0
	3	344	1.22	1.24	-1.46	368	1.26	1.3	-3.27
	4	380	1.35	1.35	-0.79	415	1.42	1.45	-2.17
	r	0.138		–	–	0.178		–	–
20T	1	227	1	1	0	236	1	1	0
	3	283	1.25	1.30	-4.24	307	1.3	1.36	-4.89
	r	0.176		–	–	0.223		–	–

6.2.2 Case-study circuit 2: Adders chain

Considering Equations 6.7 and 6.9, a large power overhead is reached when dynamic power is very high too. Since dynamic power is related to the switching activity, any circuit switching a large number of flip-flops and LUTs can be considered as a bad case from the point of view of power overhead.

Figure 6.8: Power overhead of nMR of miniMIPS obtained by XPower (XP) and by the proposed model from 6.9 synthesized into the different FPGA Virtex-5 devices (XC5VLX20T, XC5VLX30T, XC5VLX50T).



A synthetic adder chain circuit composes by 190 16-bit adders was selected to explore the power overhead of a circuit with high dynamic power consumption. Then, the nMR system analyzed is composed by 7 adder chain circuit (basic module) working with a SAv as shown in Figure 6.9. The number of redundancies and adders aimed to use the more amount of resources of a Virtex-5 LX50T considering a dedicated placement. Each module has the same inputs sourced by a generator pattern based on a 32-bit LFSR to guarantee a high and random switching activity. The switching activity file (vsd file) was created using the post routing model.

Figure 6.9: Diagram of 7MR 16-bit adders used in the power analysis.

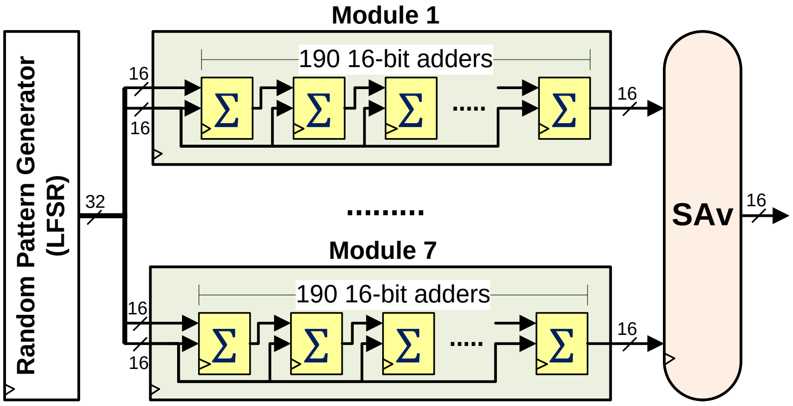


Table 6.5 shows the synthesis results for Virtex-5 LX50T FPGA for 3, 5 and 7 redundancies. The total power and power overhead estimated by XPower and by the proposed model presented in Equations 6.9, running at 25Mhz, 50Mhz, 100Mhz and 200Mhz are presented in Table 6.6. The maximum operational frequency is 260 MHz, and the average static power (obtained from XPower tool) is 211.6 mW. Using the dynamic and static power consumption obtained from XPower Tool, the *r* values for 7MR are 0.153, 0.297, 0.572, and 1.121, for the system running at 25Mhz, 50Mhz, 100Mhz and 200Mhz respectively. We want to to highlight that although replicating 7 times the original circuit, using

almost the totality of LUTs and flip-flops of the FPGA and having a high switching activity, the higher *r* that we got is 1.120 with a power overhead of 3.32. We interpret these results as the fact that for common circuits, the penalty for using *n* modular redundancies in SRAM-based FPGA is much lower than *n*.

Table 6.5: Resources used by Adder chains nMR in three Virtex-5 devices.

n	**Virtex-5 LX50T**		
	LUTs (%)	Reg. (%)	BRAM (%)
1	10.56	10.83	0
3	32.48	32.23	0
5	53.60	54.84	0
7	74.96	76.62	0
SAv	0.72	0.86	0

Powers overhead estimated by XPower and by the Equation 6.9 are plotted in Figure 6.10. Table 6.6 also presents the power overhead error of the model presented in Equations 6.9 respect to XPower results. We can notice the good accuracy of the model. According to the results, the Equation 6.9 estimate the power overhead with a maximum of error of 2.11%.

Figure 6.10: Power overhead of nMR of adder chains obtained by XPower (XP) and by the proposed model (Mod) from Equation 6.9 for Virtex-5 LX50T FPGA.

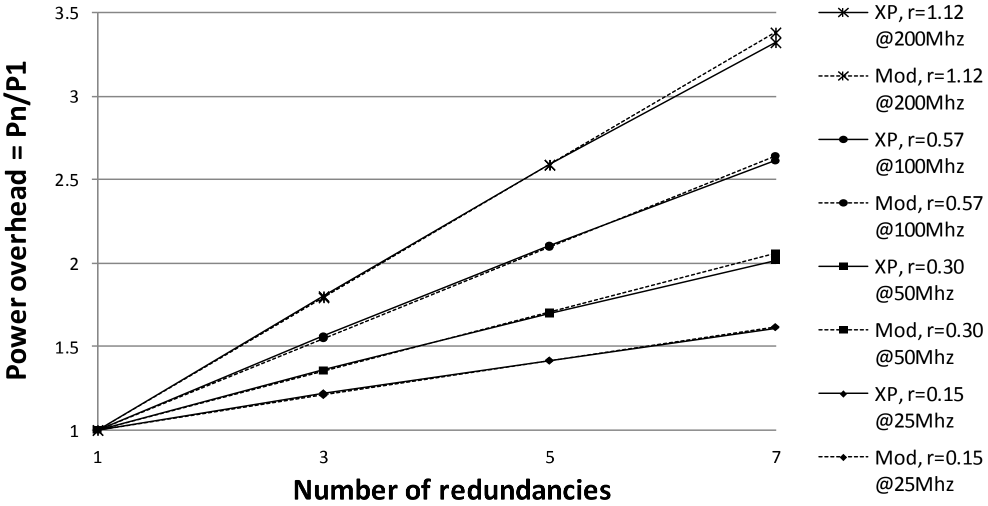


Table 6.6: Power consumption estimated by XPower and by the model proposed in the Equation 6.9 for the Adder chain nMR running at 25Mhz, 50 Mhz, 100Mhz and 200 Mhz in XC5VLX50T FPGA.

n	**25 Mhz**				**50 Mhz**			
	XPower		Equation 6.9		XPower		Equation 6.9	
	P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error (%)	P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error (%)
1	498	1	1	0	562	1	1	0
3	608	1.22	1.21	1.23	762	1.36	1.35	0.27
5	706	1.42	1.41	0.41	953	1.70	1.71	-0.52
7	803	1.61	1.62	-0.33	1132	2.02	2.06	-2.11
r	0.153		–	–	0.297		–	–
n	**100 Mhz**				**200 Mhz**			
	XPower		Equation 6.9		XPower		Equation 6.9	
	P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error (%)	P_{TOT} (mW)	P_{OV}	P_{OV-flb}	Error (%)
1	684	1	1	0	931	1	1	0
3	1068	1.56	1.55	0.90	1678	1.80	1.79	0.44
5	1440	2.11	2.09	0.50	2412	2.59	2.59	0.08
7	1787	2.61	2.64	-1.13	3094	3.32	3.38	-1.80
r	0.572		–	–	1.120		–	–

7 RELIABILITY ANALYSIS OF NMR SYSTEMS IN SRAM-BASED FPGAS

The assessment of the capability to tolerate faults of the proposed technique was conducted through fault injection emulation and radiation experiments in two case study circuits. Both circuits were implemented in a Virtex-5 FPGA, specifically the XC5VLX50T component which is part of the Genesys board from Digilent company. Finally, we analyze the cost of the nMR implementations in terms of power consumed.

7.1 Case-study circuits

The criterion of selection of the first circuit prioritizes the low logic masking of faults and the facility to climb it, so that its implementation uses the widest possible area of the device. These criteria guarantee a high susceptibility to errors caused by faults in the configuration memory of the FPGA which improves the statistic analysis. The adders chain circuit meets these criteria.

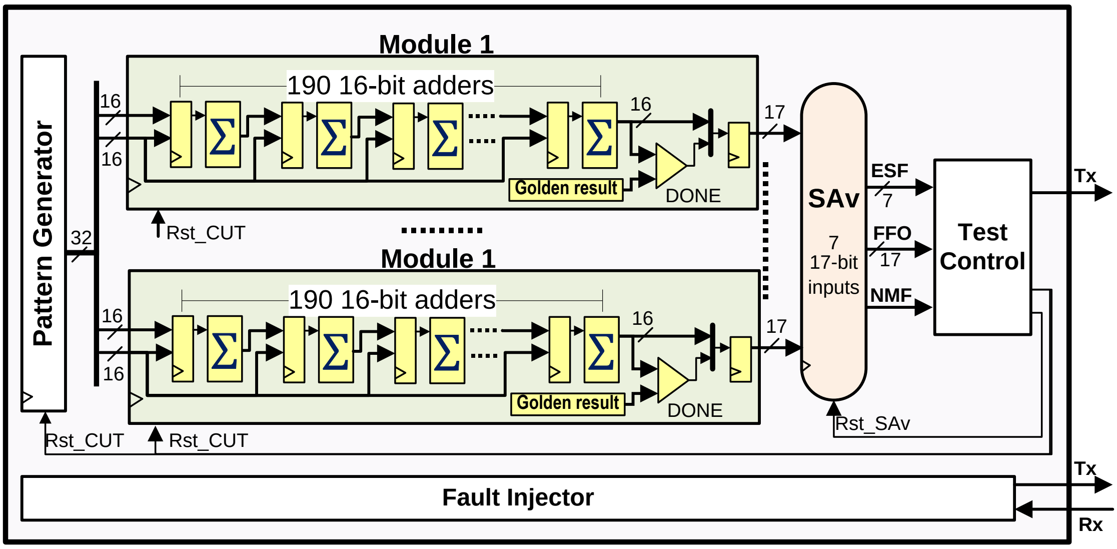
The second case study circuit aims to analyze the use of nMR in a wide used application. We selected a miniMIPS soft-processor running a 6x6 matrix multiplication, although this application has inherent masking features.

7.1.1 Adder chain

The Virtex-5 XC5VLX50T has 28,000 LUTs and 28,000 registers, however, it is not possible to use all resources due the complexity of routing. Moreover, the synthesis tool is configured to respect the hierarchy of the design to avoid the share of the same resources by more than one module. This synthesis strategy seeks to prevent that a single event upsets causes the fault of more than one module, but also since one CLB has 8 LUTs and 8 registers , there are LUTs and registers not used by the design.

Considering the implementation of a 7MR adder chain as basic module, a SAv of 7 17-bits inputs, a pattern generator to source the inputs of each adder block, the test control block and also the fault injector, the maximum number of adders of each module is 190, and flip-flops are used between each adder. Figure 7.1 shows a block diagram of the circuit under test (CUT). The generator block is implemented by a 32-bit counter which is initialized in an specific value. The 16-bit most significant bits compose the first adder operator, and the rest 16-bit less significant bits compose the second operator. In this way the adder chain block result is deterministic and we can expect the correct result known as ‘golden’ result in an specific time. In our experiments, after 37163 clock cycles the expected ‘golden’ result is X‘5ACE’. Each adder chain block set a flag ‘DONE’ when the output is the golden result. DONE signal is also voted by the SAv, so, there are 17 bits

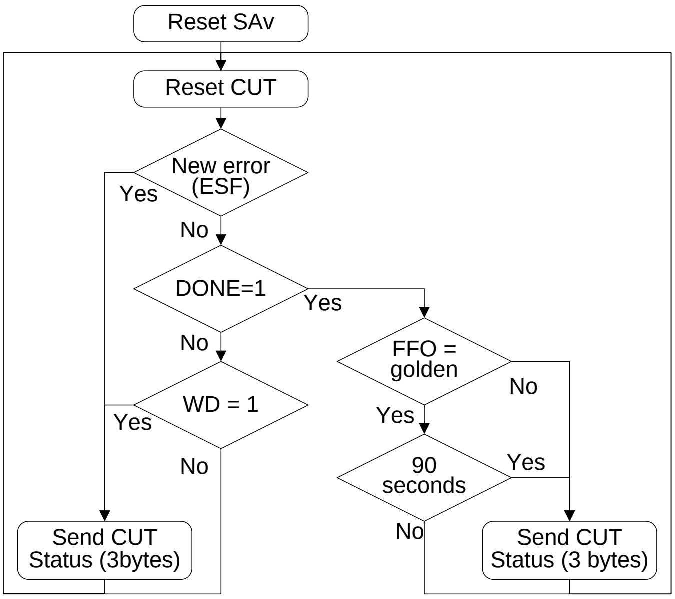
Figure 7.1: Block diagram of 7MR of adders chain circuit.



(16 bits from the last adder and one from DONE) sent to the SAv.

Test control block implement a watchdog circuit to signalize if the expected DONE signal is not set in 37200 clock cycles. Hence, this block can detect errors in the SAv circuit: wrong output, if the FFO is different to X‘5ACE’ when DONE is set, and stop working if DONE is not asserted in at least 37200 clock cycles. At the beginning of the experiment, all redundancy modules, pattern generator and SAv module is reseted by the test control. Since a complete run of the CUT is considered when DONE signal is set, test control reset the pattern generator and redundancy modules are reset when a watchdog error is detected or when DONE is asserted. The flow diagram of the test control is shown in Figure 7.2. The test control block sends the state of the experiment to an external PC

Figure 7.2: Flow diagram of test control.



approximately each 90 seconds or when a new error is detected, in three consecutive bytes:

- First byte, the header: ‘01010101’.
- Second byte, error status of each module from SAv: (NMF bit) & (ESF).
- Third byte, voter errors (VE) due wrong output or watchdog error (WD), and end code : (VE) & (WD) & ‘001010’.

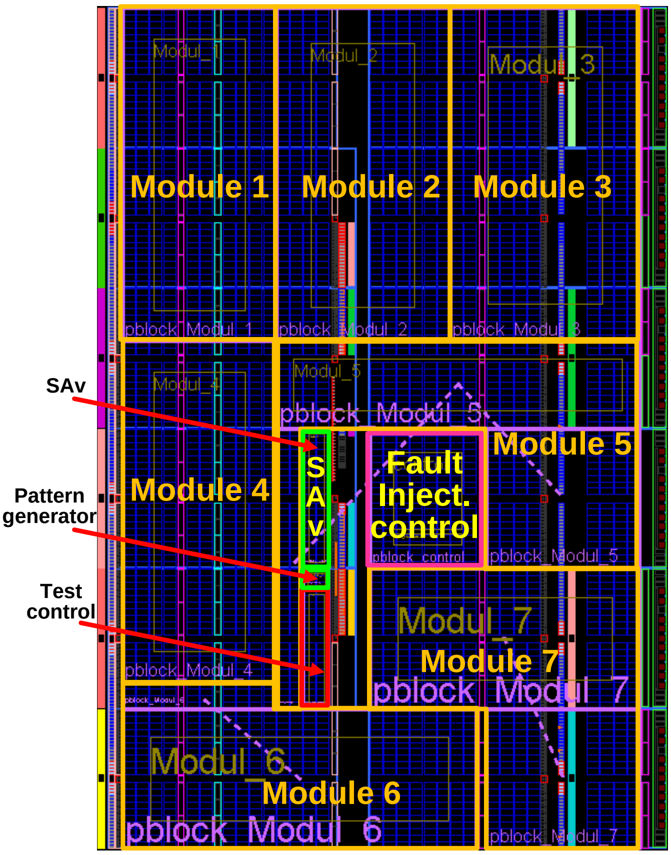
In order to validate the system before the experiment radiations, the fault injector platform was also implemented in the same design. The CUT and fault injector of Figure 7.1 was implemented considering dedicated floorplan as shown in Figure 7.3.

Table 7.1 shows the resources used by each module of the CUT and the fault injector in absolute number (#) and proportional to the constrained placement block (PBlock) and the device. Notice that SAv block uses less than 1% of LUTs and registers of the device, and is also less than 10% of the resources used by each adder chain module. This fact reduces the possibility of errors in the voter caused by radiation. Pattern generator and test control circuit are also small compared to the CUT, which is ideal for testing purposes due our goal is to evaluate the reliability of the nMR technique in radiation conditions. Fault injector block uses almost 3% of LUTs and registers, and is larger than the test control. However this block is only used during fault injection campaigns and has not influence in the CUT.

Table 7.1: Used resources for adders chain case-study circuit implemented in XC5VLX50T FPGA.

	LUTs			**Registers**			**BRAMs**	
Resources	#	% (PBlock)	% (device)	#	% (PBlock)	% (device)	#	%
Module 1	3,044	78.44	10.57	3,072	79.17	10.67	0	0
Module 2	3,044	78.44	10.57	3,072	79.17	10.67	0	0
Module 3	3,044	78.44	10.57	3,072	79.17	10.67	0	0
Module 4	3,044	78.44	10.57	3,072	79.17	10.67	0	0
Module 5	3,044	78.44	10.57	3,072	79.17	10.67	0	0
Module 6	3,044	78.44	10.57	3,072	79.17	10.67	0	0
Module 7	3,044	78.44	10.57	3,072	79.17	10.67	0	0
Generator	15	31.25	0.05	16	33.33	0.06	0	0
SAv	247	77.19	0.86	144	45.00	0.50	0	0
Test control	152	55.88	0.53	121	44.49	0.42	0	0
Total CUT	**21,722**	**–**	**75.42**	**21,785**	**–**	**75.64**	**0**	**0**
Fault injector	851	66.48	2.95	643	50.23	2.23	2	3.33
Total	**22,573**	**–**	**78.38**	**22,428**	**77.88**	**–**	**2**	**3.33**

Figure 7.3: Floorplan of the adder chains 7MR in XC5VLX50T FPGA.



7.1.2 miniMIPS

The second case study circuit is based on the 32-bit MIPS processor in softcore version (implemented using the configurable resources of the FPGA) named miniMIPS (HANGOUT; JAN, 2009). We selected a processor as case study due it is very useful in systems on chip application. In order to use the more quantity of the common configurable resources of the FPGA as LUTs and registers, we modified the original sources and removed the DSP elements. We also optimized the size of BRAMS according to the application program. The maximum number of miniMPIS that we got to implement in the XC5VLX50T FPGA were 6. The used algorithm was a 6x6 matrix multiplication implemented in assembler code. When complied, tis program uses 2,337 32-bit words, and since the data bus uses 32 bits, we required a 12-bit address bus.

The test control has the same functionality of the previous case. Notice that in this case we do not need of any pattern generator. The test control reset all processors when the DONE is asserted. The expected time to get the 'golden' result is 3,375 clock cycles when the 'golden' result is X'A80'. watchdog error signal is set if DONE signal is not asserted in 3,500 clock cycles. Then the flow diagram of the test control is the same shown in Figure 7.2. The test control block sends the state of the experiment to an external PC approximately each 90 seconds or when a new error is detected, in three consecutive bytes:

- First byte, the header: '01010101'.
- Second byte, error status of each module from SAv: '0' & (NMF bit) & (ESF).

- Third byte, voter errors (VE) due wrong output or watchdog error (WD), and end code : (VE) & (WD) & ‘001010’.

The CUT and fault injector of Figure 7.4 was implemented considering dedicated floorplan as shown in Figure 7.5.

Figure 7.4: Block diagram of 6MR of miniMIPS circuit.

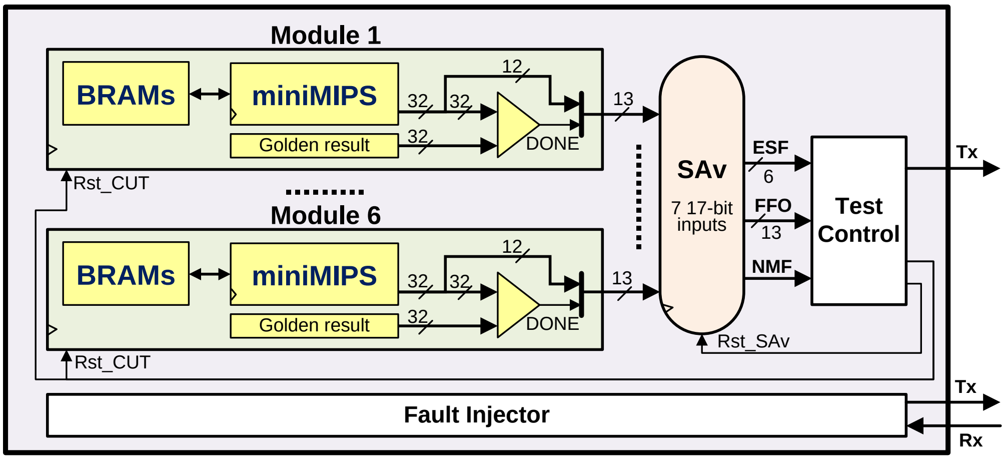


Figure 7.5: Floorplan of miniMIPS 6MR in XC5VLX50T FPGA.

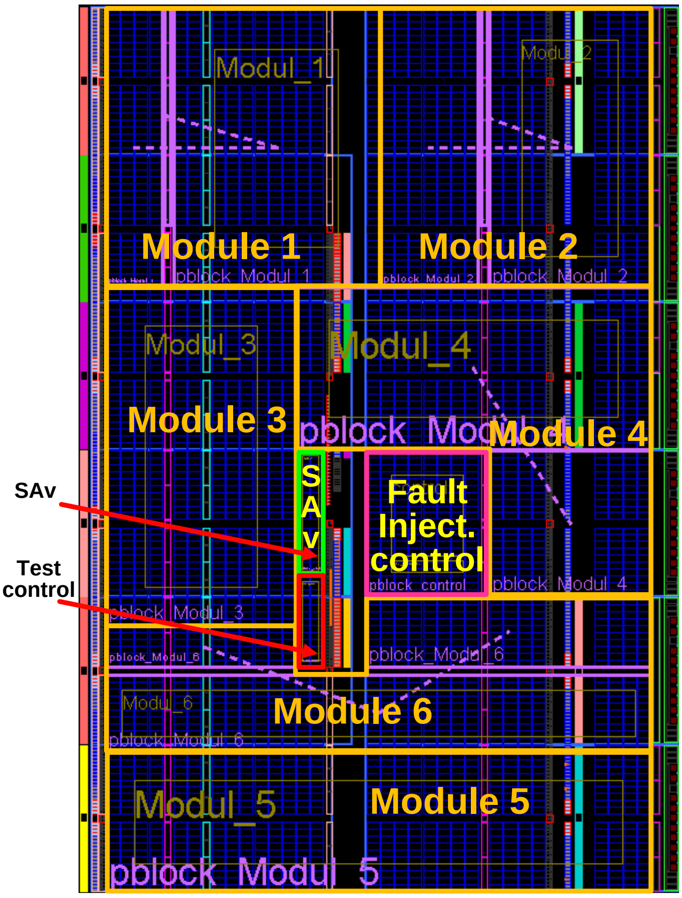


Table 7.2 shows the resources used by each module of the CUT and the fault injector in absolute number (#) and proportional to the constrained placement block (PBlock) and the device. Notice that SAv block uses less than 1% of LUTs and registers of the device, and is also less than 10% of the resources used by each adder chain module. This fact reduces the possibility of errors in the voter caused by radiation. Test control circuit are also small compared to the CUT, which is ideal for testing purposes due our goal is to evaluate the reliability of the nMR technique in radiation conditions. Fault injector block uses almost 3% of LUTs and registers, and is larger than the test control. However this block is only used during fault injection campaigns and has not influence in the CUT.

Table 7.2: Used resources for miniMIPS case-study circuit implemented in XC5VLX50T FPGA.

	LUTs			Registers			BRAMs		
Resources	#	% (PBlock)	% (device)	#	% (PBlock)	% (device)	#	% (PBlock)	% (device)
Module 1	3,514	77.06	12.20	1,500	32.89	5.21	3	50.00	5.00
Module 2	3,514	77.06	12.20	1,500	32.89	5.21	3	50.00	5.00
Module 3	3,514	79.57	12.20	1,500	33.97	5.21	3	37.50	5.00
Module 4	3,514	79.00	12.20	1,500	33.72	5.21	3	75.00	5.00
Module 5	3,514	77.06	12.20	1,500	32.89	5.21	3	50.00	5.00
Module 6	3,514	78.16	12.20	1,500	33.36	5.21	3	42.86	5.00
SAv	193	70.96	0.67	98	36.03	0.34	0	0	0
Test cntrl.	174	83.65	0.60	117	56.25	0.41	0	0	0
Total CUT	**21,451**	**–**	**74.48**	**9,215**	**–**	**32.00**	**0**	**0**	**0**
Fault inj.	851	66.48	2.95	643	50.23	2.23	2	50.00	3.33
Total	**22,302**	**–**	**77.44**	**9,858**	**–**	**34.23**	**2**	**–**	**33.33**

7.2 Fault injection campaigns results

Before irradiation experiments, the accumulation fault effects in both adder chains 7MR and miniMIPS 6MR were availed using the fault injector described in Chapter 4. The fault injection test campaign takes advantage of 1,550 faults collected from previous neutron ground test experiments that are stored into the SEU position memory. There also were used around 3,000 SEU locations generated randomly by Matlab tool.

Figure 7.6 shows the number of flipped bits to provoke 1, 2, 3, 4, 5 and 6 faulty modules of adder chains, with a 95% confidence interval for 10 campaigns.

In the case of miniMIPS study case circuit, Figure 7.7 shows the number of flipped bits to provoke 1, 2, 3, 4, and 5 faulty modules of miniMIPS, with a 95% confidence interval. We can notice that it is necessary to have a higher number of upsets to one module fails, nevertheless this behavior was expected due processors have more masking capability than the adder chains implemented.

The SAv of 7 17-bits inputs was evaluated by fault injection campings. The 1,550 collected faults fomr radiation experiments plus 1,000 SEUs positions generated randomly with Matlab tool were used to emulate SEUs. SAv was tested in similar way to the adders chain experiment, but SAv inputs were sourced by the pattern generator instead of the adder chain blocks. After the experiment no faults in the SAv were detected. This means that the susceptibility of the voter is very low compared to the other elements.

Figure 7.6: Number of accumulated faults needed to provoke multiple faulty modules under fault injection in the adder chain case-study implemented in XC5VLX50T FPGA.

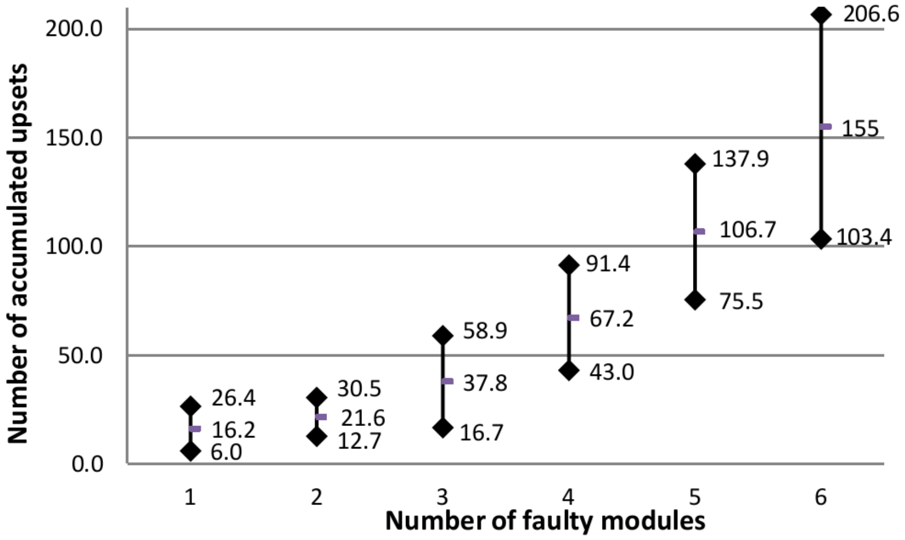
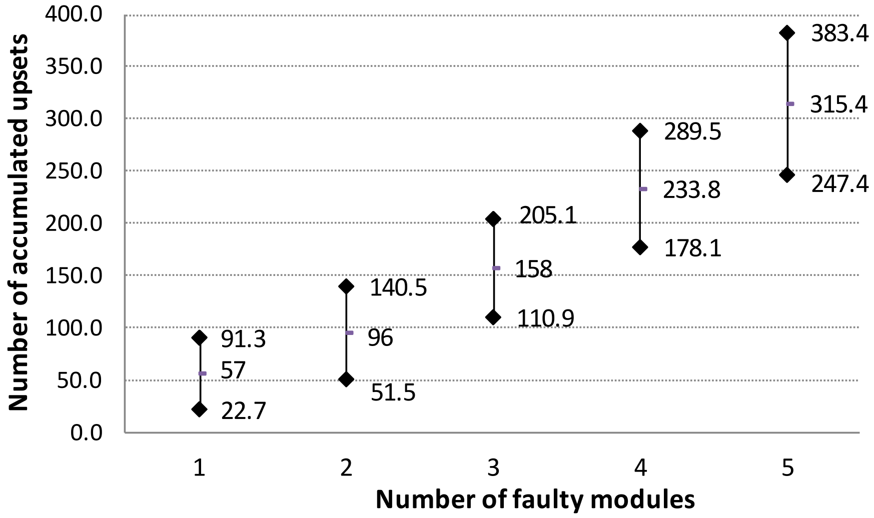


Figure 7.7: Number of accumulated faults needed to provoke multiple faulty modules under fault injection in the miniMIPS case-study implemented in XC5VLX50T FPGA.



7.3 Neutron radiation results

Case study circuits were implemented in the Virtex-5 XC5VLX50T FPGA and irradiated with the neutron spectrum available in the ISIS facility in the CCLRC Rutherford Appleton Laboratory, Didcot, UK, which resembles the atmospheric one.

The device was irradiated with neutrons produced at ISIS by the spallation process: a heavy-metal target (tungsten) is bombarded with pulses of highly energetic protons, generating neutrons from the nuclei of the target atoms (VIOLANTE, 2007). Figure 7.8 the experiment setup mounted inside the VESUVIO irradiation chamber at ISIS. The available neutron flux was of about 3.7x10^4 neutrons/s/cm^2, and the overall irradiation time time was 3,500 minutes.

Figure 7.8: Virtex-5 testing in the VESUVIO irradiation chamber.

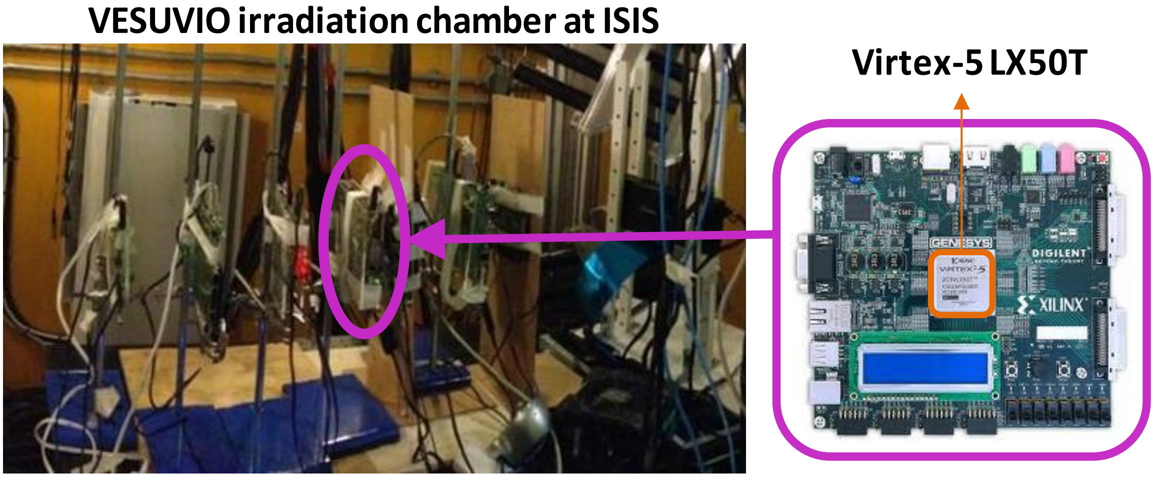


Figure 7.9 shows depicts the test setup. The PC can reconfigure the FPGA through JTAG interface when a no-correctible error is detected (NMF=1) or when no message is received from FPGA in one minute. The PC also runs a C program which generates a log file of the test process based on information received from SAv and the bistream values obtained by readback process. Figure 7.10 shows a flow methodology of radiation test process.

Figure 7.9: Test setup of the nMR the system under radiation.

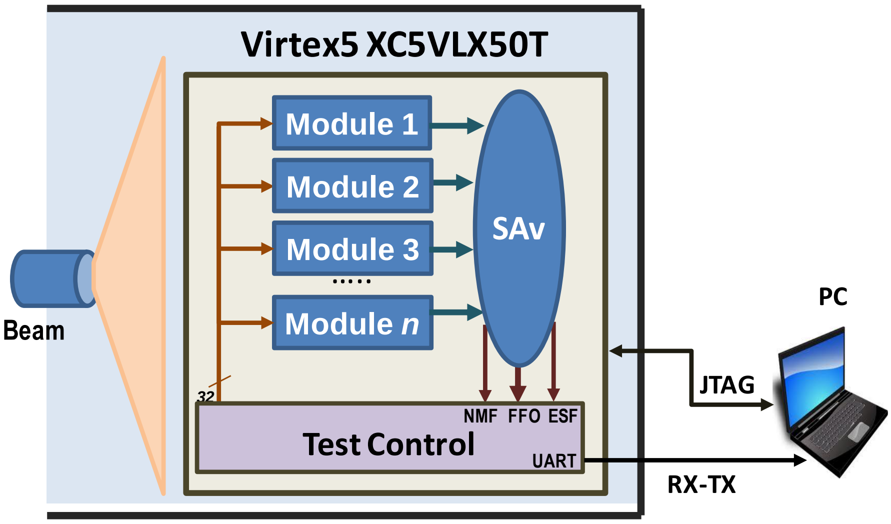
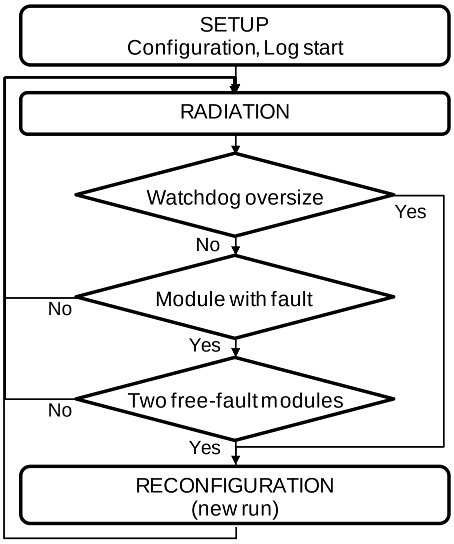


Figure 7.10: Radiation test flow methodology.



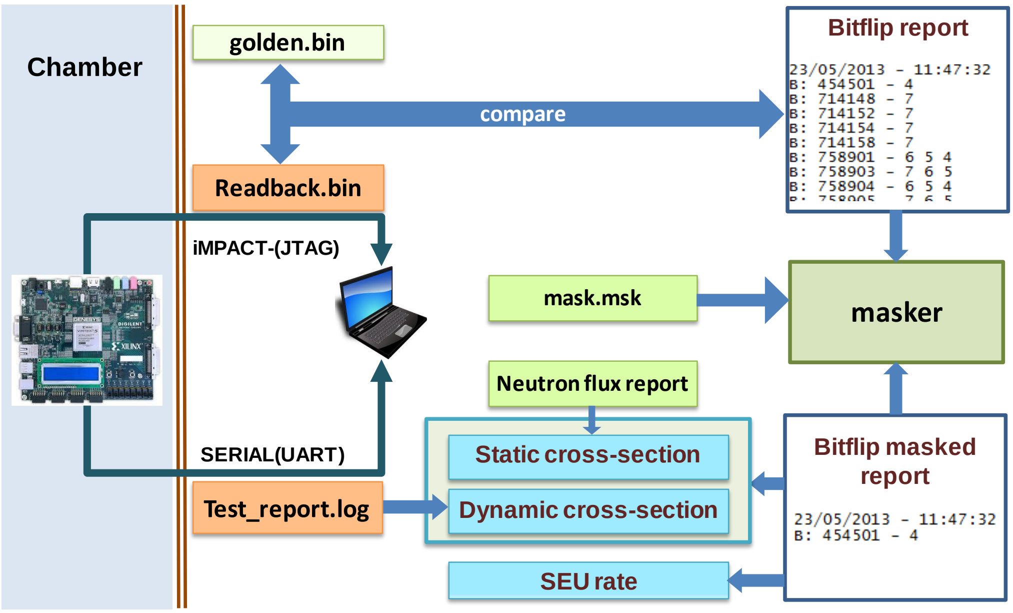
Radiation experiment has the following goals:

- Determine the SEU rate of the device for the neutron flux irradiated.
- Determine the static bit cross-section and device cross-section.
- Determine the dynamic cross-section of the nMR system considering different number of redundancies.

In order to get the cross-section and SEU rate from radiation experiments, the readback is performed by Impact Xilinx tool each 90 seconds or when a new error is detected. When a readback is implemented, a readback.bin file is generated automatically containing the state of the configuration memory including the BRAM data and LFSR implemented by the CLBs eventually. The detection of bit-flip is performed by comparing the current readback.bin file with a 'golden.bin' readback file which is obtained before the radiation experiment. Nevertheless, BRAM data and LFSR data (known also as dynamic configuration bits) changes according to the dynamic of the application and can be confused as a bit-flip. ISE synthesis tool may generate a 'mask.msk' file which contains the position where the dynamic configuration bits are located in the bin file. A 'masker' program described in C language is used to obtain the report of the bit-flips caused by radiation according to the time of the experiment. The report of bit-flips and the neutrons flux is used to obtain the SEU rate and the static cross-section On the other hand, the report generated by the PC test program with the information received from the test control block is used to determine the dynamic cross-section. The Figure 7.11 shows the methodology to obtain the cross-section and SEU rate. Both the bit-flips report as the Test_report.log are obtained on-line with the experiment, while bit-flip masked report is generated off-line with the experiment.

Similar to fault injection campaign, the number of accumulated upsets that provoke fault modules was taken into account in each run. Figure 7.12 shows the average of accumulated upsets for all runs, with 95% confidence intervals. According to experimental results, similar number of accumulated faults was needed to provoke modules to fail when compared results from the fault injection and from the neutron radiation test experiment

Figure 7.11: Radiation analysis methodology.

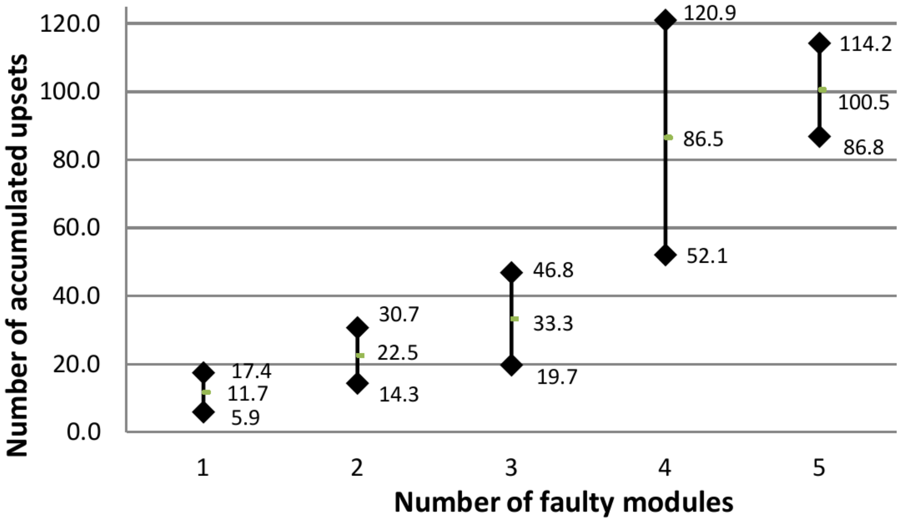


```
23/05/2013 - 11:47:32
B: 454501 - 4
B: 714148 - 7
B: 714152 - 7
B: 714154 - 7
B: 714158 - 7
B: 758901 - 6 5 4
B: 758903 - 7 6 5
B: 758904 - 6 5 4
B: 758905 - 7 6 5
```

```
23/05/2013 - 11:47:32
B: 454501 - 4
```

as shown in Figure 7.13. This is because the injected SEU locations in the injection fault campaigns were obtained from previous radiation experiments in ISIS facilities and using similar neutron flux.

Figure 7.12: Radiation results: Number of accumulated faults needed to provoke multiple faulty modules in the adder chain case-study circuit implemented in XC5VLX50T FPGA.



The experimental cross-section (σ) was obtained by dividing the number of observed errors by the fluence (number of particles hitting the device per unit area). The static cross-section of the tested device was measured to be 9.17x10^{-08} cm^2/device with a 95% confidence interval of (8.26x10^{-08} cm^2/device, 1.01x10^{-07} cm^2/device). The observed upset rate was 0.20 upset/min with a 95% confidence interval of (0.18 upset/min, 0.222 upset/min).

Figure 7.13: Comparison between fault injection and radiations results of adder chain tests.

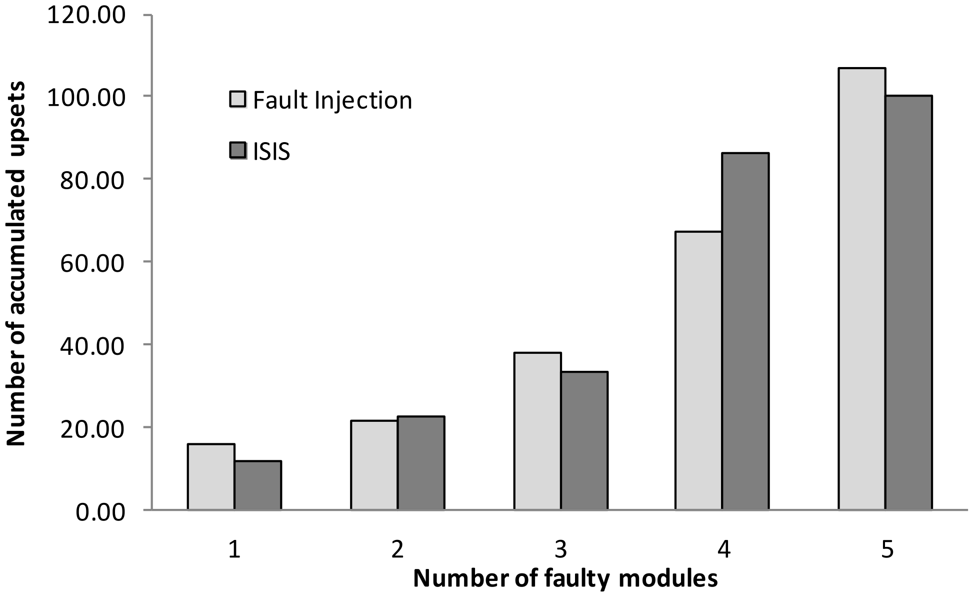


Figure 7.14 shows average cross-section values and their confidence intervals for several nMR systems: n = 3, 4, 5, 6 and 7. Despite the long duration of the experiments, the confidence intervals are large due to the small number of runs. Nevertheless, results show the reduction trend of the cross-section when the number of redundancies is incremented. As shown, cross-section falls off significantly from n=3 to n=4 and keeps falling smoothly for n greater than 4. Despite of this, the proportion of such cross-section reduction is 4.8 times from n = 3 to n = 4, 2.81 times from n = 4 to n = 5, 1.95 times from n = 5 to n = 6, and 1.94 from n = 6 to n = 7.

Figure 7.14: Radiation results: Neutron cross-section for nMR adder chain case-study implemented in XC5VLX50T FPGA for *n* = 3 to *n* = 7.

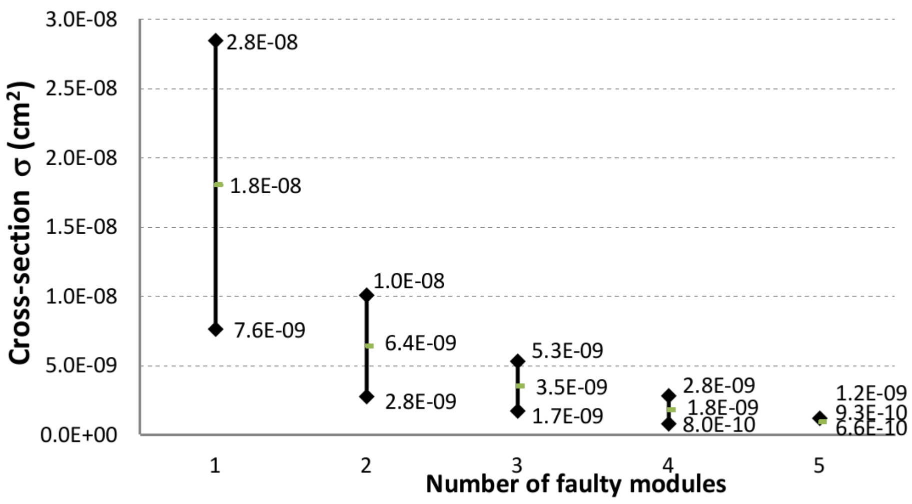


Table 7.3 shows the reliability results in terms of MTTF. The second column of the Table represents the average values, while the next column shows the confidence interval with a 95% of confidence. Considering that the The cross-section at ISIS resemble the cross-section at sea level, and at sea level the neutron flux is around 13neutrons/cm^2/h, Table 7.4 presents the average SER in FITs for ISIS experiments and the expected MTTF at sea level.

Table 7.3: MTTF in seconds of adder chains nMR according to neutron radiation results.

nMR	**MTTF (average)**	**Confidence intervale (with 95%)**
3MR	3770 s.	[1318.8, 6221.2] s.
4MR	6750 s.	[3536.8, 9963.2] s.
5MR	10425 s.	[6199.1, 14650.9] s.
6MR	29070 s.	[16379.1, 41760.9] s.
7MR	33120 s.	[24797.3, 41442.7] s.

Table 7.4: Average MTTF from ISIS experiments and expected MTTF at sea level considering 13neutrons/cm^2/h at sea level.

nMR	**SER-ISIS**	**SER-sea level**
3MR	9.55×10^8 FITs	234.64 FIT
4MR	5.33×10^8 FITs	83.42 FIT
5MR	3.45×10^8 FITs	45.66 FIT
6MR	1.24×10^8 FITs	23.45 FIT
7MR	1.09×10^8 FITs	12.06 FIT

7.4 Reliability and Power analysis

As discussed, the hardening solution area overhead increases proportionally with n. However, since the amount of resources available in an FPGA increases in each new generation, the area (resources) may be considered as a minor constraint in some common designs. The operating frequency does not change too much as the redundant modules work in parallel. Power consumption, on the contrary, is a critical parameter in FPGAs devices as static power increases linearly with the amount of resources. So, considering new technologies, one can decide whether it is worth to use more than three redundancy modules when high reliability is required but consuming slightly more power. In Table 7.5, one can notice that when comparing TMR with 7MR of adder chains, cross-section reduces in 19.46 times and power increases only 1.31 times. Moreover, 7MR can also save power in the scrubbing technique. If considering blind scrubbing, since a 7MR system allows until 5 faulty modules in the system while the TMR allows only 1 faulty module, the number of accumulated upsets observed in 7MR reaches 4 times higher than in

Table 7.5: Average power overhead versus cross-section reduction for adder chains case study.

	Power	σ	Comparison with TMR	
System	(mW)	(cm^2)	Increase in Power	Reduction in σ
3MR	409	180.5E-10	1.00x	1.00x
4MR	445	64.2 E-10	1.09x	2.81x
5MR	476	35.1 E-10	1.16x	5.14x
6MR	511	18.0 E-10	1.25x	10.01x
7MR	535	9.28E-10	1.31x	19.46x

TMR. So the scrubbing rate would be approximately 4 times lower. However, the final scrubbing power consumption depends on the implementation strategy. External scrubbing (HEINER; COLLINS; WIRTHLIN, 2008) requires the use of dedicated input/output pins, which uses higher voltage than internal elements and consequently higher power consumption. Internal scrubbing (BERG et al., 2008) avoids the use of external components but requires the use of extra internal blocks as scrubbing control circuit and internal configuration access port, which represent a power overhead.

8 CONCLUSIONS AND DISCUSSIONS

In this Thesis, we have proposed the use of a multiple redundancy system composed of *n* modules, known as n-modular redundancy (nMR), to cope both with a high number of accumulated upsets between sparse scrubbings and multi-bits upset in SRAM-based FPGAs. In the proposed hardening technique, *n* identical modules operate in tandem and an innovative self-adaptive majority voter elects the modules' outputs, masking multiple-bit upsets. The main drawbacks in the use of nMR systems, are the area and power overhead. In the first case, technology trend makes FPGAs have more and more resources, and consequently, we consider that resources overhead as a minor issue. Power consumption penalties are analyzed and a predictable model based on the power characteristics of a single module have been presented in this work. The reliability and MTTF of our proposal have been analyzed by means of two case studies which were subjected to fault injection by a given platform, and neutron radiation.

In the following subsections, the contributions of this theses are summarized, the results are discussed and the future works are presented. Finally, the publications of the author during the development of this work are listed.

8.1 Contributions

8.1.1 A novel Self-Adaptive voter

In this work, a novel Self-Adaptive voter (SAv) used to nMR systems has been presented. When an nMR system implemented in a SRAM-based FPGA is exposed to radiation, bit-flips in the configuration memory of the FPGA may affect the functionality of the redundancy modules. Classical majority voters used in TMR systems (nMR with *n*=3) have a fixed voting policy of 2-out-3, since just one fault redundancy module is tolerated. Nevertheless, in nMR systems the number of tolerated redundancy modules depends on the value of *n* and consequently the voting policy may change according to the evolution of the radiation effects in the nMR system. SAv takes into account the *n* value as the number of the current faulty modules in the system to select on-the-fly the appropriate voting policy. In this work, we have implemented a SAv used to vote 7 inputs of 17 bits each, and other one used to vote 6 inputs of 13 bits each. The amount of resources used by the SAv in the systems presented are very lower compared to the resources used by the redundancy modules. The scalability of the SAv has been also studied.

8.1.2 Power penalty model for redundancy systems in SRAM-based FPGA

Power overhead is a main concern for the application of modular redundancy in SRAM-based FPGAs, however, it is not easy to find a power consumption analysis in the literature. In order to analyze the power overhead penalty due the increasing number of redundancy modules, in this work a mathematical model based on the power characteristics of a single module has been proposed. The power penalty model considers the replication of the functional logic such as their input and output signals. Nevertheless the most nMR implementations do not consider inputs and outputs replication, a variation of the model which do not consider the replication of inputs and outputs is also proposed. Power penalty model was applied to two case study circuits implemented in an XC5VLX50T FPGA and compared to results obtained from the estimation power tool of the vendor, where the discrepancy obtained was less than 10%. Results show that power overhead in nMR systems depends on the n used, but also depends on the relationship between the dynamic and static power consumption of a single module r. Moreover, according to the results, the power overhead of a nMR system is far lower than n for a typical design, since in SRAM-based FPGAs exist a big amount of transistors that are not used by the design but consume static power.

8.1.3 Radiation test methodology

In this work, radiation experiments were performed in ISIS facilities of Routherford laboratories in England. Radiation experiments are usually slow and expensive, so it is necessary to take as much data as possible in a reliable way. Whereas the aim of our experiments is to obtain the static effects (SEU rate, bit and device cross-section) and dynamic effects (dynamic cross-section) of radiation in the proposed technique, the methodology used in this work allows to collect in automated process the information of the position and the time of the bit-flips produced, besides the state of the voting nMR system. During the experiments, the readback of the configuration bitstream is performed from an external PC each 90 seconds (taking into account that SEU rate for similar experiments is usually more than 3 minutes) or when a new error is detected by the test control circuit implemented in the irradiated FPGA. If the external PC does not receive any information from the FPGA or detects that the SAv can not mask any other fault, PC performs the reconfiguration of the FPGA.

8.1.4 Fault injection platform

A novel fault injector platform to analyze the radiation effects in the memory configuration of a SRAM-based FPGAs is also presented in this work. The novelty of the proposed fault injector is based on the use of results from previous radiation experiments (neutron beam during more than 10 days) to select the location of the bits to be flipped which are stored in an on-board flash memory. Moreover, bitflips locations can also be defined by a pseudo random generator pattern or by another customized locations stored in the on-board flash memory. The fault injector control is composed by a soft-processor to perform the communication with an external PC and to control the injection process through the use of an ICAP primitive, buffer memory and a memory controller. Injector control uses less than 3% of the LUTs and flip-flops resources of an XC5VLX50T, and the desired number of bitflips is selected through an external host PC. The fault injector using the bitflip locations from previous experiments was used to predict the effects of radiation on two case study circuits in few minutes, and their results were compared with those obtained by radiation experiments in many days, noting the closeness of their results.

8.2 Discussions and future works

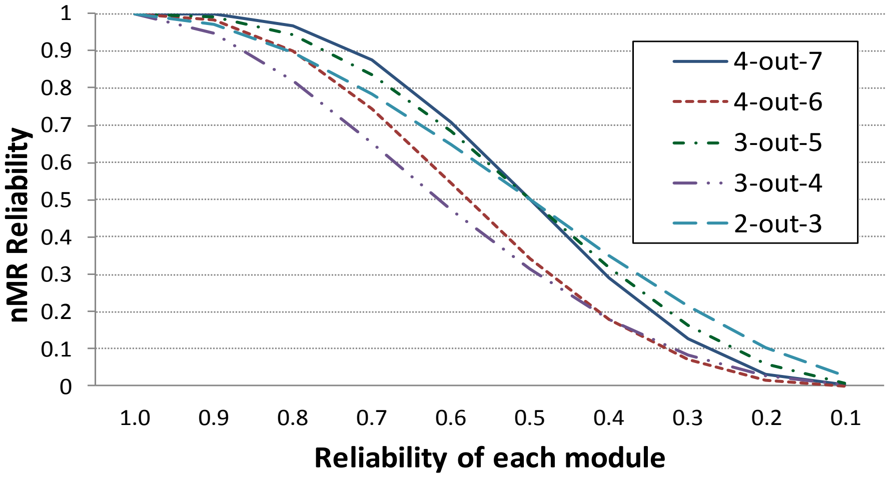
8.2.1 Exploring the voting policies

In this work, the voter policies used by the SAv changes whenever any module fails. This policy was implemented by the SAv and tested by fault injection and neutron radiation in two different case study circuits, proving the increase of reliability when compared with the traditional and new techniques. Nevertheless, others policies can be used looking for an optimal trade-off between reliability, MTTF, MTTR, availability, amount of resources used by the voter, and power consumption overhead.

Figure 8.1 shows how the reliability (according to the Equation 1.1) of the system changes when the SAv changes its policy. Notice that after the first failed module is detected, the SAv changues his policy from 4-out-7 to 4-out-6. However, 4-out-6 policy has a lower reliability than 3-out-5, and then, such change may be unnecessary. Although it is not possible to predict the exactly moment when the module fails (this curves only shows the probability of fault), Figure 8.1 suggests that other voting policies may be explored. For example, the following policies may be explored:

- The change of policy can be performed when n/2+1 modules remains fault free.
- The change of policy can be performed when some number among n and n/2+1 remains fault free.
- It is not necessary to wait until 2 modules are working properly to perform the scrubbing. For example, the fault module can be corrected at the moment the fault is detected. This new approach involves the modification of the policy voting since the SAv must consider the possibility of enable a module flagged as faulty.

Figure 8.1: Reliability of nMR systems according to the voting policy used in this work.



8.2.2 Power consumption model

Although the overhead power model proposed in this Thesis uses the dynamic and static power of the original module guarantying the generality of the proposed, it was verified for an specific component and no BRAMs neither DSP blocks were used. More complex designs and other FPGA components must to be used to survey the proposed model.

On the other hand, although almost the full device was used, designs with different sized and different placement may be explored. Clock regions split the FPGA in sectors, then, depending on how these regions are used to implement the redundancy modules, the power consumption will not follows a linear function as the proposed model defines. Nevertheless, the studied cases uses represent a pessimistic situation since almost the full resources were used.

8.2.3 Internal fault correction

Despite this work is related to the masking capability of nMR systems, it is also necessary to perform the scrubbing of the FPGA. In a previous work published in (TARRILLO et al., 2014), we proposed the use of an small module to perform partial reconfiguration called DPR manager. The advantage of the DPR manager is that requires a reduced amount of resources, then the reliability of such module may cope with radiation environments. Future works may integrate the nMR system to the DPR manager, and consequently, a full protection of the SRAM-based FPGA may be achieved.

8.2.4 Exploring the optimal power, number of redundancies, synchronization, and module correction trade-off space

According to the experimental results and the power consumption overhead, it is possible to increment significantly the MTBF of the circuit implemented in a SRAM-based FPGA using nMR technique with power overhead far low than *n* factor. However, the number of redundancies, voting policies, and resynchronization sequence may affect the reliability, MTBF, availability and power consumption. It is necessary to explore all parameter combinations to reach with diverse project goals.

8.3 Publications

8.3.1 Journals

TARRILLO, J.; KASTENSMIDT, F. L; RECH, P.; FROST, C.; VALDERRAMA, C. Neutron Cross-Section of N-Modular Redundancy Technique in SRAM-based FPGAs. **IEEE Transactions on Nuclear Science**, accepted for publication, 2014.

TARRILLO, J.; AZAMBUJA, J. R.; KASTENSMIDT, F. L.; Junior, E.; VAZ, R. G.; GONCALEZ, O. L. Analyzing the Effects of TID in an Embedded System Running in a Flash-Based FPGA. **IEEE Transactions on Nuclear Science**, v. 12, p. 1-8, 2011.

8.3.2 Conferences and workshops

TARRILLO, J.; KASTENSMIDT, F. L. Estimating Power Consumption of Multiple Modular Redundant Designs in SRAM-based FPGAs for High Dependable Applications. Acepted in: IEEE Power And Timing Modeling, Optimization and Simulation (PATMOS), **Proceedings...** [S.l.: s.n.], 2014.

TARRILLO, J.; ESCOBAR, F. A.; KASTENSMIDT, F. L.; VALDERRAMA, C. Dynamic partial reconfiguration manager. In Circuits and Systems. In: IEEE 5th Latin American Symposium on Circuits and Systems (LASCAS), **Proceedings...** IEEE 2013. p. 1-4.

TARRILLO, J.; RECH, P.; FROST, C.; VALDERRAMA, C.; KASTENSMIDT, F.

L. Neutron Cross-section of N-Modular Redundancy Technique in SRAM-based FPGAs. In: 14th European Conference on Radiation and Its Effects on Components and Systems (RADECS). **Proceedings...** [S.l.: s.n.], 2013.

TARRILLO, J. ; TONFAT, J.; KASTENSMIDT, F. L.; REIS, R.; BRUGUIER, F.; BOURREE, M.; BENOIT, P.; TORRES, L. Using Electromagnetic Emanations for Variability Characterization in Flash-Based FPGAs. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI). **Proceedings...** IEEE 2013. p. 109-114.

TARRILLO, J.; ALTIERI, M. ; KASTENSMIDT, F. L. Improving error detection capability of a SpaceWire router IP. In: 12th European Conference on Radiation and Its Effects on Components and Systems (RADECS). **Proceedings...** IEEE 2011. p. 501-506.

TARRILLO, J. ; CHIELLE, E. ; CHIPANA, R. ; KASTENSMIDT, F. L. Design and Verification of a SpaceWire Router IP under SEE Effects. In: 12th IEEE Latin American Test Workshop (LATW). **Proceedings...** [S.l.: s.n.] 2011.

AZAMBUJA, J. R. ; TARRILLO, J. ; Junior, E. ; GONCALEZ, O. L. ; KASTENSMIDT, F. L Analyzing the Effects of TID in an Embedded System Running into a Flash-Based FPGA. In: IEEE Nuclear and Space Radiation Effects Conference (NSREC). **Proceedings...** [S.l.: s.n.] 2011.

REFERENCES

ADELL, P.; ALLEN, G. **Assessing and Mitigating Radiation Effects in Xilinx FPGAs**. Jet Propulsion Laboratory, California Institute of Technology, California: [s.n.], 2008.

AGUIRRE, M.; TOMBS, J.; MUOZ, F.; BAENA, V.; GUZMAN, H.; NAPOLES, J.; TORRALBA, A.; FERNÁNDEZ-LEÓN, A.; TORTOSA-LÓPEZ, F.; MERODIO, D. Selective protection analysis using a SEU emulator: testing protocol and case study over the leon2 processor. **Nuclear Science, IEEE Transactions on**, [S.l.], v.54, n.4, p.951–956, 2007.

ALDERIGHI, M.; CASINI, F.; CITTERIO, M.; D'ANGELO, S.; MANCINI, M.; PASTORE, S.; SECHI, G. R.; SORRENTI, G. Using FLIPPER to predict proton irradiation results for VIRTEX 2 devices: a case study. **Nuclear Science, IEEE Transactions on**, [S.l.], v.56, n.4, p.2103–2110, 2009.

ANGHEL, L.; NICOLAIDIS, M. Cost reduction and evaluation of a temporary faults-detecting technique. In: DESIGN, AUTOMATION, AND TEST IN EUROPE. **Proceedings...** [S.l.: s.n.], 2008. p.423–438.

ASADI, G.; TAHOORI, M. B. An accurate SER estimation method based on propagation probability. In: DESIGN, AUTOMATION AND TEST IN EUROPE-VOLUME 1. **Proceedings...** [S.l.: s.n.], 2005. p.306–307.

ASHRAF, R. A.; MOURI, O.; JADAA, R.; DEMARA, R. F. Design-for-Diversity for Improved Fault-Tolerance of TMR Systemson FPGAs. In: RECONFIGURABLE COMPUTING AND FPGAS (RECONFIG), 2011 INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2011. p.99–104.

ATHAN, S. P.; LANDIS, D. L.; AL-ARIAN, S. A. A novel built-in current sensor for I DDQ testing of deep submicron CMOS ICs. In: VLSI TEST SYMPOSIUM, 1996., PROCEEDINGS OF 14TH. **Proceedings...** [S.l.: s.n.], 1996. p.118–123.

AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. **Dependable and Secure Computing, IEEE Transactions on**, [S.l.], v.1, n.1, p.11–33, 2004.

AZAMBUJA, J. R.; SOUSA, F.; ROSA, L.; KASTENSMIDT, F. L. Evaluating large grain TMR and selective partial reconfiguration for soft error mitigation in SRAM-based FPGAs. In: ON-LINE TESTING SYMPOSIUM, 2009. IOLTS 2009. 15TH IEEE INTERNATIONAL. **Proceedings...** [S.l.: s.n.], 2009. p.101–106.

BAN, T.; NAVINER, L. A. Optimized robust digital voter in tmr designs. In: COLLOQUE NATIONAL GDR SOC-SIP. **Proceedings...** [S.l.: s.n.], 2011.

BARNABY, H. Total-ionizing-dose effects in modern CMOS technologies. **Nuclear Science, IEEE Transactions on**, [S.l.], v.53, n.6, p.3103–3121, 2006.

BARTH, J. L.; DYER, C.; STASSINOPOULOS, E. Space, atmospheric, and terrestrial radiation environments. **Nuclear Science, IEEE Transactions on**, [S.l.], v.50, n.3, p.466–482, 2003.

BAUMANN, R. C. Radiation-induced soft errors in advanced semiconductor technologies. **Device and Materials Reliability, IEEE Transactions on**, [S.l.], v.5, n.3, p.305–316, 2005.

BERG, M. Fault tolerance implementation within SRAM based FPGA designs based upon the increased level of single event upset susceptibility. In: ON-LINE TESTING SYMPOSIUM, 2006. IOLTS 2006. 12TH IEEE INTERNATIONAL. **Proceedings...** [S.l.: s.n.], 2006. p.3–pp.

BERG, M. **Complexity Management and Design Optimization Regarding a Variety of Triple Modular Redundancy Schemes through Automation**. [S.l.]: NASA, 2010.

BERG, M.; FRIENDLICH, M.; LAKEMAN, J.; WILCOX, T.; KIM, H.; LABEL, K.; PELLISH, J. Single Event Effects in Field Programmable Gate Array (FPGA) Devices: update 2012. In: NEPP ELEC. TECH. WORKSHOP, HTTP://RADHOME. GSFC. NASA. GOV/RADHOME/PAPERS/NEPP_ETW2012_ BERG. PDF. **Proceedings...** [S.l.: s.n.], 2012.

BERG, M.; POIVEY, C.; PETRICK, D.; ESPINOSA, D.; LESEA, A.; LABEL, K.; FRIENDLICH, M.; KIM, H.; PHAN, A. Effectiveness of internal versus external SEU scrubbing mitigation strategies in a Xilinx FPGA: design, test, and analysis. **Nuclear Science, IEEE Transactions on**, [S.l.], v.55, n.4, p.2259–2266, 2008.

BOLCHINI, C.; MIELE, A.; SANTAMBROGIO, M. D. TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs. In: DEFECT AND FAULT-TOLERANCE IN VLSI SYSTEMS, 2007. DFT'07. 22ND IEEE INTERNATIONAL SYMPOSIUM ON. **Proceedings...** [S.l.: s.n.], 2007. p.87–95.

BRIDGFORD, B.; CARMICHAEL, C.; TSENG, C. W. Correcting Single-Event Upsets in Virtex-II Platform FPGA Configuration Memory. In: XILINX APPLICATION NOTE, XAPP779 (V1.1). **Proceedings...** [S.l.: s.n.], 2007.

BRIDGFORD, B.; CARMICHAEL, C.; TSENG, C. W. **Single-event upset mitigation selection guide**. [S.l.]: Xilinx, 2008. (XAPP197(v1.0)).

CARMICHAEL, C. **Triple module redundancy design techniques for Virtex FPGAs**. [S.l.]: Xilinx, 2006. (XAPP197).

CARMICHAEL, C.; CAFFREY, M.; SALAZAR, A. **Correcting single-event upsets through Virtex partial configuration**. [S.l.]: Xilinx, 2000. (XAPP216(v1.0)).

CHAPMAN. **Virtex-5 SEU Critical Bit Information Extending the capability of the Virtex-5 SEU Controller**. [S.l.]: Xilinx, 2010.

CHAPMAN, K. **SEU Strategies for Virtex-5 Devices**. [S.l.]: Xilinx, 2010. (XAPP864(v2.0)).

CHAPMAN, K. **New Generation Virtex-5 SEU Controller**. [S.l.]: Xilinx, 2010. (2).

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

DODD, P. E.; SHANEYFELT, M. R.; FELIX, J. A.; SCHWANK, J. R. Production and propagation of single-event transients in high-speed digital logic ICs. **Nuclear Science, IEEE Transactions on**, [S.l.], v.51, n.6, p.3278–3284, 2004.

GUZMAN-MIRANDA, H.; TOMBS, J.; AGUIRRE, M. FT-UNSHADES-up: a platform for the analysis and optimal hardening of embedded systems in radiation environments. In: IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS. **Proceedings...** [S.l.: s.n.], 2008. p.2276–2281.

HAN, J.; BOYKIN, E. R.; CHEN, H.; LIANG, J.; FORTES, J. A. On the reliability of computational structures using majority logic. **Nanotechnology, IEEE Transactions on**, [S.l.], v.10, n.5, p.1099–1112, 2011.

HANGOUT, L.; JAN, S. The minimips project. **Available at opencores. org/projects. cgi/web/minimips/overview**, [S.l.], 2009.

HEINER, J.; COLLINS, N.; WIRTHLIN, M. Fault tolerant ICAP controller for high-reliable internal scrubbing. In: AEROSPACE CONFERENCE, 2008 IEEE. **Proceedings...** [S.l.: s.n.], 2008. p.1–10.

HEINER, J.; SELLERS, B.; WIRTHLIN, M.; KALB, J. FPGA partial reconfiguration via configuration scrubbing. In: FIELD PROGRAMMABLE LOGIC AND APPLICATIONS, 2009. FPL 2009. INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2009. p.99–104.

HERRERA-ALZU, I.; LÓPEZ-VALLEJO, M. Design techniques for Xilinx Virtex FPGA configuration memory scrubbers. **IEEE Transactions on Nuclear Science**, [S.l.], v.60, p.376–385, 2013.

HIARI, O.; SADEH, W.; RAWASHDEH, O. Towards single-chip diversity TMR for automotive applications. In: ELECTRO/INFORMATION TECHNOLOGY (EIT), 2012 IEEE INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2012. p.1–6.

IBE, E.; TANIGUCHI, H.; YAHAGI, Y.; SHIMBO, K.-i.; TOBA, T. Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule. **Electron Devices, IEEE Transactions on**, [S.l.], v.57, n.7, p.1527–1538, 2010.

ISIS. **Science and Technology Facilities Council**. http://www.isis.stfc.ac.uk: ISIS, 2014.

ITRS. **International Technology Roadmap for Semiconductors**. Available: http://www.itrs.net/Links/2011ITRS/2011Chapters/2011Design.pdf: ITRS, 2011.

KASTENSMIDT, F. L.; FONSECA, E. C. P.; VAZ, R. G.; GONÇALEZ, O. L.; CHIPANA, R.; WIRTH, G. I. TID in flash-based FPGA: power supply-current rise and logic function mapping effects in propagation-delay degradation. **IEEE Trans. Nucl. Sci**, [S.l.], v.58, n.4, p.1927–1934, 2011.

KASTENSMIDT, F. L.; STERPONE, L.; CARRO, L.; REORDA, M. S. On the optimal design of triple modular redundancy logic for SRAM-based FPGAs. In: DESIGN, AUTOMATION AND TEST IN EUROPE-VOLUME 2. **Proceedings...** [S.l.: s.n.], 2005. p.1290–1295.

KIM, E. P.; SHANBHAG, N. R. Soft N-modular redundancy. **Computers, IEEE Transactions on**, [S.l.], v.61, n.3, p.323–336, 2012.

KUON, I.; ROSE, J. Measuring the gap between FPGAs and ASICs. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.26, n.2, p.203–215, 2007.

LALA, J. H.; HARPER, R. E. Architectural principles for safety-critical real-time applications. **Proceedings of the IEEE**, [S.l.], v.82, n.1, p.25–40, 1994.

LANSCE. **Los Alamos Nuclear Science Center Los Alamos National Laboratory**. Available: http://www.lansce.lanl.gov: LANSCE, 2014.

LUO, P.; ZHANG, J. SEU mitigation strategies for SRAM-based FPGA. In: INTERNATIONAL SYMPOSIUM ON PHOTOELECTRONIC DETECTION AND IMAGING 2011. **Proceedings...** [S.l.: s.n.], 2011. p.81960N–81960N.

MACQUEEN, D.; GINGRICH, D.; BUCHANAN, N.; GREEN, P. Total ionizing dose effects in a SRAM-based FPGA. In: RADIATION EFFECTS DATA WORKSHOP. **Proceedings...** [S.l.: s.n.], 1999. n.24.

MAIZ, J.; HARELAND, S.; ZHANG, K.; ARMSTRONG, P. Characterization of multi-bit soft error events in advanced SRAMs. In: ELECTRON DEVICES MEETING, 2003. IEDM'03 TECHNICAL DIGEST. IEEE INTERNATIONAL. **Proceedings...** [S.l.: s.n.], 2003. p.21–4.

MANUZZATO, A.; GERARDIN, S.; PACCAGNELLA, A.; STERPONE, L.; VIOLANTE, M. Effectiveness of TMR-based techniques to mitigate alpha-induced SEU accumulation in commercial SRAM-based FPGAs. In: RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS, 2007. RADECS 2007. 9TH EUROPEAN CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2007. p.1–7.

MCMURTREY, D.; MORGAN, K.; PRATT, B.; WIRTHLIN, M. Estimating TMR reliability on FPGAs using markov models. **BYU Dept. Electr. Comput. Eng., Tech. Rep**, [S.l.], 2006.

MORGAN, K.; CAFFREY, M.; GRAHAM, P.; JOHNSON, E.; PRATT, B.; WIRTHLIN, M. SEU-induced persistent error propagation in FPGAs. **Nuclear Science, IEEE Transactions on**, [S.l.], v.52, n.6, p.2438–2445, 2005.

NAZAR, G. L.; CARRO, L. Fast single-FPGA fault injection platform. In: DEFECT AND FAULT TOLERANCE IN VLSI AND NANOTECHNOLOGY SYSTEMS (DFT), 2012 IEEE INTERNATIONAL SYMPOSIUM ON. **Proceedings...** [S.l.: s.n.], 2012. p.152–157.

NAZAR, G. L.; RECH, P.; FROST, C.; CARRO, L. Radiation and Fault Injection Testing of a Fine-Grained Error Detection Technique for FPGAs. **Nuclear Science, IEEE Transactions on**, [S.l.], v.60, n.4, p.2742–2749, 2013.

NAZAR, G.; SANTOS, L.; CARRO, L. Scrubbing unit repositioning for fast error repair in FPGAs. In: COMPILERS, ARCHITECTURE AND SYNTHESIS FOR EMBEDDED SYSTEMS (CASES), 2013 INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2013. p.1–10.

NIKNAHAD, M.; SANDER, O.; BECKER, J. Fine grain fault tolerance — A key to high reliability for FPGAs in space. In: AEROSPACE CONFERENCE, 2012 IEEE. **Proceedings...** [S.l.: s.n.], 2012. p.1–10.

NORMAND, E. Single event upset at ground level. **IEEE transactions on Nuclear Science**, [S.l.], v.43, n.6, p.2742–2750, 1996.

NORMAND, E. Correlation of inflight neutron dosimeter and SEU measurements with atmospheric neutron model. **Nuclear Science, IEEE Transactions on**, [S.l.], v.48, n.6, p.1996–2003, 2001.

NORMAND, E.; DOMINIK, L. Cross comparison guide for results of neutron SEE testing of microelectronics applicable to avionics. In: RADIATION EFFECTS DATA WORKSHOP (REDW), 2010 IEEE. **Proceedings...** [S.l.: s.n.], 2010. p.8–8.

OLDHAM, T. R.; MCLEAN, F. et al. Total ionizing dose effects in MOS oxides and devices. **IEEE Transactions on Nuclear Science**, [S.l.], v.50, n.3, p.483–499, 2003.

OSTLER, P. S.; CAFFREY, M. P.; GIBELYOU, D. S.; GRAHAM, P. S.; MORGAN, K. S.; PRATT, B. H.; QUINN, H. M.; WIRTHLIN, M. J. SRAM FPGA reliability analysis for harsh radiation environments. **Nuclear Science, IEEE Transactions on**, [S.l.], v.56, n.6, p.3519–3526, 2009.

PLATT, S.; TOROK, Z.; FROST, C. D.; ANSELL, S. Charge-collection and single-event upset measurements at the ISIS neutron source. **Nuclear Science, IEEE Transactions on**, [S.l.], v.55, n.4, p.2126–2132, 2008.

PRADHAN, D. K. **Fault-tolerant computer system design**. [S.l.]: Prentice-Hall, Inc., 1996.

PRATT, B.; CAFFREY, M.; GRAHAM, P.; MORGAN, K.; WIRTHLIN, M. Improving FPGA design robustness with partial TMR. In: RELIABILITY PHYSICS SYMPOSIUM PROCEEDINGS, 2006. 44TH ANNUAL., IEEE INTERNATIONAL. **Proceedings...** [S.l.: s.n.], 2006. p.226–232.

QUINN, H.; GRAHAM, P. Terrestrial-based Radiation Upsets: a cautionary tale. In: ANNUAL IEEE SYMPOSIUM ON FIELD-PROGRAMMABLE CUSTOM COMPUTING MACHINES (FCCM 05), 13. **Proceedings...** IEEE, 2005.

QUINN, H.; GRAHAM, P.; KRONE, J.; CAFFREY, M.; REZGUI, S. Radiation-induced multi-bit upsets in SRAM-based FPGAs. **IEEE Transactions on Nuclear Science**, [S.l.], v.52, n.6, p.2455–2461, 2005.

QUINN, H.; GRAHAM, P.; MORGAN, K.; BAKER, Z.; CAFFREY, M.; SMITH, D.; WIRTHLIN, M.; BELL, R. **Flight Experience of the Xilinx Virtex-4**. [S.l.]: IEEE-INST ELECTRICAL ELECTRONICS ENGINEERS INC 445 HOES LANE, PISCATAWAY, NJ 08855-4141 USA, 2013. 2682–2690p. v.60, n.4.

QUINN, H. M.; GRAHAM, P. S.; WIRTHLIN, M. J.; PRATT, B.; MORGAN, K. S.; CAFFREY, M. P.; KRONE, J. B. A test methodology for determining space readiness of Xilinx SRAM-based FPGA devices and designs. **Instrumentation and Measurement, IEEE Transactions on**, [S.l.], v.58, n.10, p.3380–3395, 2009.

QUINN, H.; MORGAN, K.; GRAHAM, P.; KRONE, J.; CAFFREY, M. A review of Xilinx FPGA architectural reliability concerns from Virtex to Virtex-5. In: EUROPEAN CONFERENCE ON RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS, 9. **Proceedings...** [S.l.: s.n.], 2007. p.1–8.

QUINN, H.; MORGAN, K.; GRAHAM, P.; KRONE, J.; CAFFREY, M.; LUNDGREEN, K. Domain crossing errors: limitations on single device triple-modular redundancy circuits in xilinx fpgas. **IEEE Transactions on Nuclear Science**, [S.l.], v.54, n.6, p.2037–2043, 2007.

RAINE, M.; HUBERT, G.; GAILLARDIN, M.; ARTOLA, L.; PAILLET, P.; GIRARD, S.; SAUVESTRE, J.-E.; BOURNEL, A. Impact of the radial ionization profile on SEE prediction for SOI transistors and SRAMs beyond the 32-nm technological node. **Nuclear Science, IEEE Transactions on**, [S.l.], v.58, n.3, p.840–847, 2011.

RAINE, M.; HUBERT, G.; GAILLARDIN, M.; PAILLET, P.; BOURNEL, A. Monte Carlo prediction of heavy ion induced MBU sensitivity for SOI SRAMs using radial ionization profile. **Nuclear Science, IEEE Transactions on**, [S.l.], v.58, n.6, p.2607–2613, 2011.

RAO, P.; EBRAHIMI, M.; SEYYEDI, R.; TAHOORI, M. B. Protecting SRAM-based FPGAs Against Multiple Bit Upsets Using Erasure Codes. In: THE 51ST ANNUAL DESIGN AUTOMATION CONFERENCE ON DESIGN AUTOMATION CONFERENCE. **Proceedings...** [S.l.: s.n.], 2014. p.1–6.

RCNP. **Research Center for Nuclear Physics.** 2014.

REZGUI, S.; WILCOX, E.; LEE, P.; CARTS, M.; LABEL, K.; NGUYEN, V.; TELECCO, N.; MCCOLLUM, J.; MANAZZA, L. R. Investigation of low dose rate and bias conditions on the total dose tolerance of a CMOS flash-based FPGA. In: IEEE NUCLEAR AND SPACE RADIATION EFFECTS. **Proceedings...** [S.l.: s.n.], 2012. p.134–143.

RITER, R. Modeling and testing a critical fault-tolerant multi-process system. In: FAULT-TOLERANT COMPUTING, 1995. FTCS-25. DIGEST OF PAPERS., TWENTY-FIFTH INTERNATIONAL SYMPOSIUM ON. **Proceedings...** [S.l.: s.n.], 1995. p.516–521.

SATORI, J.; SLOAN, J.; KUMAR, R. Fluid NMR-performing power/reliability tradeoffs for applications with error tolerance. In: WORKSHOP ON POWER AWARE COMPUTING AND SYSTEMS. **Proceedings...** [S.l.: s.n.], 2009.

SCHWIERZ, F. Graphene transistors. **Nature nanotechnology**, [S.l.], v.5, n.7, p.487–496, 2010.

SEIFERT, N.; GILL, B.; FOLEY, K.; RELANGI, P. Multi-cell upset probabilities of 45nm high-k+ metal gate SRAM devices in terrestrial and space environments. In: RELIABILITY PHYSICS SYMPOSIUM, 2008. IRPS 2008. IEEE INTERNATIONAL. **Proceedings...** [S.l.: s.n.], 2008. p.181–186.

SHOOMAN, M. L. **Reliability of Computer Systems and Networks**: fault tolerance, analysis, and design. [S.l.]: Wiley Online Library, 2002.

SIMEVSKI, A.; HADZIEVA, E.; KRAEMER, R.; KRSTIC, M. Scalable design of a programmable NMR voter with inputs' state descriptor and self-checking capability. In: ADAPTIVE HARDWARE AND SYSTEMS (AHS), 2012 NASA/ESA CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2012. p.182–189.

SMITH, F.; MOSTERT, S. Reconfigurable FPGA Computing to Mitigate for Total Ionizing Dose Effects. In: AEROSPACE CONFERENCE, 2007 IEEE. **Proceedings...** [S.l.: s.n.], 2007. p.1–13.

STERPONE, L.; ULLAH, A. On the optimal reconfiguration times for TMR circuits on SRAM based FPGAs. In: ADAPTIVE HARDWARE AND SYSTEMS (AHS), 2013 NASA/ESA CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2013. p.9–14.

STERPONE, L.; VIOLANTE, M.; REZGUI, S. An analysis based on fault injection of hardening techniques for SRAM-based FPGAs. **Nuclear Science, IEEE Transactions on**, [S.l.], v.53, n.4, p.2054–2059, 2006.

STRAKA, M.; KOTASEK, Z. High availability fault tolerant architectures implemented into fpgas. In: DIGITAL SYSTEM DESIGN, ARCHITECTURES, METHODS AND TOOLS, 2009. DSD'09. 12TH EUROMICRO CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2009. p.108–115.

TAHOORI, M. B.; MITRA, S. Automatic configuration generation for FPGA interconnect testing. In: IEEE 31ST VLSI TEST SYMPOSIUM (VTS), 2013. **Proceedings...** [S.l.: s.n.], 2003. p.134–134.

TAMBARA L.; RECH, P. K. F. F. C. Evaluating the Effectiveness of a Diversity TMR Scheme under Neutrons. In: RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS (RADECS), 2013 14TH EUROPEAN CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2013.

TARRILLO, J.; AZAMBUJA, J. R.; KASTENSMIDT, F. L.; FONSECA, E. C. P.; GALHARDO, R.; GONCALEZ, O. Analyzing the effects of TID in an embedded system running in a flash-based FPGA. **Nuclear Science, IEEE Transactions on**, [S.l.], v.58, n.6, p.2855–2862, 2011.

TARRILLO, J.; ESCOBAR, F.; LIMA KASTENSMIDT, F.; VALDERRAMA, C. Dynamic Partial Reconfiguration Manager. In: IEEE 5TH LATIN AMERICAN SYMPOSIUM ON CIRCUITS AND SYSTEMS., 2014. **Proceedings...** [S.l.: s.n.], 2014.

TRIUMF. **Canadaś national laboratory for particle and nuclear physics.** 2014.

TUAN, T.; LAI, B. Leakage power analysis of a 90nm FPGA. In: CUSTOM INTEGRATED CIRCUITS CONFERENCE, 2003. PROCEEDINGS OF THE IEEE 2003. **Proceedings...** [S.l.: s.n.], 2003. p.57–60.

VIOLANTE, L. S. M. A new partial reconfiguration-based fault-injection system to evaluate SEU effects in SRAM-based FPGAs. **Nuclear Science, IEEE Transactions on**, [S.l.], v.54, n.4, p.965–970, 2007.

VIOLANTE, M.; STERPONE, L.; MANUZZATO, A.; GERARDIN, S.; RECH, P.; BAGATIN, M.; PACCAGNELLA, A.; ANDREANI, C. et al. A new hardware/software platform and a new 1/E neutron source for soft error studies: testing fpgas at the isis facility. **Nuclear Science, IEEE Transactions on**, [S.l.], v.54, n.4, p.1184–1189, 2007.

WANG, X. Partitioning triple modular redundancy for single event upset mitigation in FPGA. In: E-PRODUCT E-SERVICE AND E-ENTERTAINMENT (ICEEE), 2010 INTERNATIONAL CONFERENCE ON. **Proceedings...** [S.l.: s.n.], 2010. p.1–4.

XILINX. **PicoBlaze 8-Bit Embedded Microcontroller User Guide**. [S.l.]: Xilinx, 2005. (UG129).

XILINX. **Virtex-II Pro and Virtex-II Pro X FPGA User Guide**. 2007.

XILINX. **Virtex-4 FPGA Configuration User Guide**. [S.l.]: Xilinx, 2009. (UG071 (v1.11)).

XILINX. **Virtex-5 Family Overview**. [S.l.]: Xilinx, 2009. (DS100 (v5.0)).

XILINX. **Virtex-5 FPGA Data Sheet**: dc and switching characteristics. [S.l.]: Xilinx, 2010. (DS202 (v5.3)).

XILINX. **Partial Reconfiguration User Guide**. [S.l.]: Xilinx, 2010. (UG702(v12.3)).

XILINX. **LogiCORE IP XPS HWICAP**. [S.l.]: Xilinx, 2010. (DS586(v5.00a)).

XILINX. **LogiCORE IP Soft Error Mitigation Controller**. [S.l.]: Xilinx, 2010. (UG764(v1.1)).

XILINX. **Continuing Experiments of Atmospheric Neutron Effects on Deep Submicron Integrated Circuits**. [S.l.]: Xilinx, 2011. (WP286 (v1.1)).

XILINX. **Xilinx Power Tools Tutorial**. [S.l.]: Xilinx, 2011. (UG733 (v13.1)).

XILINX. **Virtex-5 FPGA, User Guide**. [S.l.]: Xilinx, 2012. (UG190 (v5.4)).

XILINX. **Virtex-6 FPGA Configurable Logic Block, User Guide**. [S.l.]: Xilinx, 2012. (UG364 (v1.2).

XILINX. **Virtex-5 FPGA Configuration User Guide**. [S.l.]: Xilinx, 2012. (UG191 (v3.11)).

XILINX. **Virtex-6 FPGA Configuration, user guide**. [S.l.]: Xilinx, 2013. (UG360 (v3.7)).

XILINX. **7 Series FPGAs Configuration, User Guide**. [S.l.]: Xilinx, 2013. (UG470 (v1.7)).

XILINX. **Device Reliability Report, Fourth Quarter 2013**. [S.l.]: Xilinx, 2014. (UG116 (v9.8)).

XILINX. **7 Series FPGAs Overview**. [S.l.]: Xilinx, 2014. (UG180 (v1.15)).

XILINX. **Virtex-6 FPGA Clocking Resources**. [S.l.]: Xilinx, 2014. (UG362 (v2.5)).

ZHU, M.; SONG, N.; PAN, X. Mitigation and Experiment on Neutron Induced Single-Event Upsets in SRAM-Based FPGAs. **IEEE Transactions on Nuclear Science**, [S.l.], v.60, p.3063 – 3073, 2013.