UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

EDUARDO NUNES DE SOUZA

Single Event Upset mitigation for FPGA-based Low-Density Parity-Check decoder

Monograph presented in partial fulfillment of the requirements for the degree of Bachelor in Computer Engineering.

Advisor: Prof. Dr. Gabriel Luca Nazar

Porto Alegre
2018

“The development of technology will leave only one problem:
the infirmity of human nature.”

Karl Kraus

ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor Gabriel, my advisor, for his patient guidance and assistance.

I would also like to extend my thanks to Marcos, for his help with the fault injection platform, and Geferson, for his assistance with the decoder.

Finally, I wish to thank my family and friends for their support and encouragement throughout my study.

ABSTRACT

With the increasing of data rates and physical limitation defined by channel capacity, communication systems have to be designed with high efficiency and reliability. LDPC codes have emerged over the last decades and became a key component of many commercialized systems as a benefit of their excellent performance and suitability to parallel hardware implementation. Under that scenario, FPGA-based decoders have been exploited since these devices offer rapid prototyping and high levels of parallelism. FPGAs, as any semiconductor device, have become sensitive to radiation due to the continual evolution of fabrication technology, such as device shrinkage, power supply reduction and increasing operating speeds. FPGAs' cells are especially susceptible to single event upsets (SEUs) and fault tolerance techniques must be applied in order to mitigate their effects. In this work, it is presented a study about the effects of SEUs in an FPGA-based LDPC decoder and it is proposed a selective technique to improve reliability in this specific application.

Keywords: Fault Tolerance. Low-density Parity-Check. Forward Error Correction. Field-Programmable Gate Array. Single event upset.

Mitigação de *single event upsets* em um decodificador LDPC implementado em FPGA

RESUMO

Com o aumento das taxas de dados e limitações físicas definidas pela capacidade do canal, os sistemas de comunicação devem ser projetados com alta eficiência e confiabilidade. Os códigos LDPC emergiram nas últimas décadas e se tornaram um componente-chave de vários sistemas comerciais, como resultado de seu excelente desempenho e possibilidade de paralelismo. Nesse contexto, implementações em FPGAs vêm sendo exploradas, uma vez que esses dispositivos oferecem prototipagem rápida e altos níveis de paralelismo. Os FPGAs, como qualquer dispositivo semicondutor, tornaram-se sensíveis à radiação devido à evolução contínua da tecnologia de fabricação, como encolhimento do dispositivo, redução da voltagem de alimentação e aumento das velocidades de operação. As células dos FPGAs são especialmente suscetíveis a *single event upsets* (SEUs) e técnicas de tolerância a falhas devem ser aplicadas para atenuar seus efeitos. Neste trabalho, é apresentado um estudo sobre os efeitos de SEUs em um decodificador LDPC implementado em FPGA e uma técnica seletiva para aumentar a confiabilidade nesta aplicação específica é proposta.

Palavras-chave: Tolerância a falhas. *Low-density Parity-Check*. *Forward Error Correction*. *Field-Programmable Gate Array*. *Single event upset.*

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BP	Belief Propagation
BPSK	Binary Phase-Shift Keying
BUBs	Bit Update Blocks
CLB	Configurable Logic Block
CNs	Check Nodes
DMR	Double Modular Redundancy
DSP	Digital Signal Processor
FEC	Forward Error Correction
FF	Flip-Flop
FPGAs	Field Programmable Gate Arrays
HDL	Hardware Description Language
IC	Integrated Circuit
ICAP	Internal Configuration Access Port
IDS	Informed Dynamic Scheduling
IOB	Input/output Block
LBP	Layered Belief Propagation
LDPC	Low-Density Parity-Check
LLR	Logarithmic-Likelihood Ratio
LUT	Lookup Table
MOS	Metal Oxide Semiconductor
PCUBs	Parity Check Update Blocks
QC	Quasi-cyclic
RAM	Random Access Memory
SEE	Single Event Effect
SEFIs	Single Event Functional Interrupts
SEL	Single Event Latch-up
SET	Single Event Transient
SEU	Single Event Upsets
SRAM	Static Random Access Memory

STMR	Selective Triple Modular Redundancy
TID	Total Ionizing Dose
TMR	Triple Modular Redundancy
VNs	Verification Nodes

SUMMARY

1 INTRODUCTION

1.1 Motivation

The main purpose of a digital communication system is to transmit data from one extremity of the system to the other efficiently and with an acceptable level of quality and reliability. The quality of the signal is commonly expressed in terms of Bit Error Rate (BER), i.e., the probability of bit errors measured in the receiver side. The power of the transmitted signal and the channel's bandwidth are the most fundamental parameters of a system and, along with the noise spectral density, they determine the energy per bit to noise power spectral density ratio (E_b/N_0). Practical restrictions frequently limit the value of E_b/N_0 and the modulation scheme used may not capable of providing an acceptable BER. In such cases, the best approach to guarantee data integrity is to encode the message transmitted by applying error control, which may be performed with Forward Error Correction (FEC) codes (HAYKIN, 2004).

Low-Density Parity-Check (LDPC) codes represent a powerful class of FEC codes, i.e., they may be employed for correcting transmission errors in communication systems. They were conceived by Gallager (1962) in his doctoral dissertation, but at the time, they were impractical to implement due to the lack of the necessary hardware technology. Thirty-four years later, Mackay and Neal (1996) verified that the performance of LDPC codes are equivalent to that of Turbo codes and could approach Shannon's theoretical limit. Since their rediscovery, they have been extensively used in several commercial standards like WiFi, WiMAX, DVB-S2, CCSDS and ITU G.hn (HAILES et al., 2015).

LDPC codes are defined by a parity-check matrix, in which the rows are the coefficients of the parity-check equations. A code is considered regular if the number of nonzero elements does not vary among each row or each column of the parity-check matrix. Irregular codes, on the other hand, do not respect this property. In general, irregular codes have a better performance than regular codes (CARRASCO, 2009). Matrices $\mathbf{H_1}$ and $\mathbf{H_2}$, as follows, are parity-check matrix for an irregular code and a regular code, respectively. The corresponding parity-check equations are listed aside.

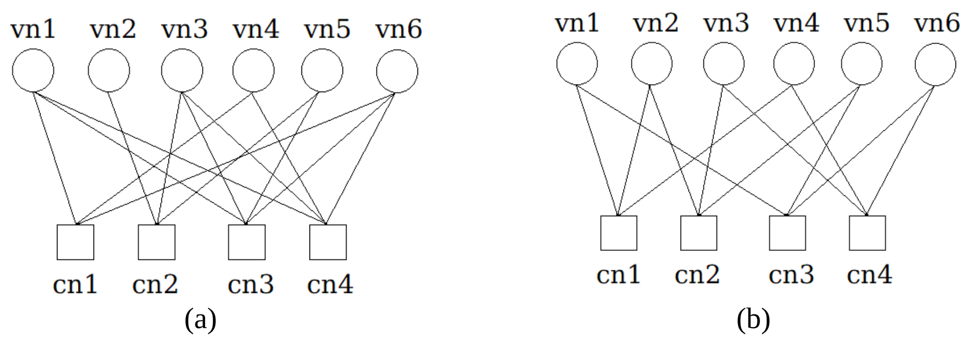
In the meantime, between the creation and effective usage, LDPC codes were not heavily investigated. One exception is the work of Tanner (1981), in which he generalized LDPC codes and proposed a graphical representation for them. Each row represents a parity check equation and each column of the matrix represents a codeword bit. The so-called factor graphs introduce the concepts of Check Nodes (CN), each of them is attached with a row of the

$$\mathbf{H_1} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad \begin{array}{l} c_1+c_4+c_6 = 0, \\ c_2+c_3+c_5 = 0, \\ c_1+c_3+c_5+c_6 = 0, \\ c_1+c_3+c_4 +c_6= 0. \end{array}$$

$$\mathbf{H_2} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad \begin{array}{l} c_1+c_2+c_4 = 0, \\ c_2+c3+c5 = 0, \\ c_1+c5+c6 = 0, \\ c_3+c_4+c_6 = 0. \end{array}$$

parity-check matrix, and the Variable Nodes (VN), which are related with the columns. A connection between a CN and a VN represents a bit ‘1’ in the parity-check matrix. Figure 1.1 illustrates the corresponding factor graph to parity-check matrices $\mathbf{H_1}$ and $\mathbf{H_2}$.

Figure 1.1 – Factor graph representations for matrices $\mathbf{H_1}$ (a) and $\mathbf{H_2}$ (b)

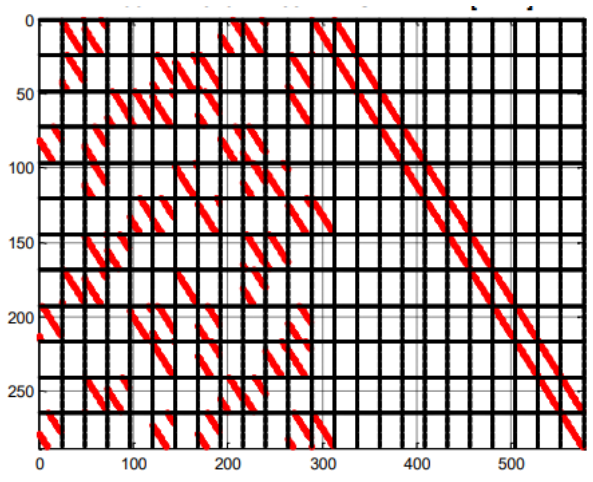


Source: the author

The parity-check matrix of an LDPC code is often sparse, i.e., it has few ‘1’s and many ‘0’s, which allows the decoding to be partly-parallelized. Quasi-cyclic (QC) LDPC code is a class of construction code whose structure facilitates low-complexity memory addressing and routing for the hardware implementation. It is based on a matrix, in which each element represents an equally-sized square submatrix (*Z x Z*). If a particular element in the matrix has a value of ‘0’, then the corresponding submatrix is a null matrix. Otherwise, the submatrix is an identity matrix, which has been cyclically shifted a number of times according to the corresponding value (HAILES et al., 2015). Figure 1.2 shows the parity-check matrix for a QC-LDPC code implemented in IEEE 802.11ad standard with a code rate = $\frac{1}{2}$, i.e., half of the coded message is formed by redundant bits.

Besides the implementation of LDPC codes using low-complexity calculations, a high level of parallelism can be exploited, which makes them suitable to be implemented in Field

Figure 1.2 – Matrix H for QC-LDPC implemented in IEEE 802.11, with code rate = $\frac{1}{2}$



Source: Avaliant (2018)

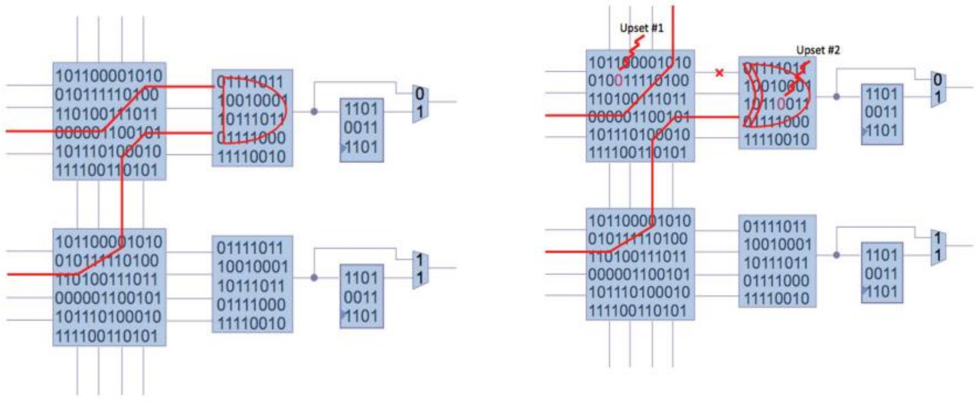
Programmable Gate Arrays (FPGAs). These devices are extremely versatile and can offer high degree of parallel processing. In addition, they offer rapid prototyping and are especially useful for measuring BER performance, due to the reduced time that the simulations take when compared, for instance, to a general-purpose processor (HAILES et al., 2015). Their usage has been increasing more and more, and they are not restricted to coding applications, FPGAs are present in other fields such as industrial, automobile and medical applications.

When these applications run over the presence of ionizing radiation, FPGAs (as any other semiconductor device) are susceptible to many effects that may either damage the device or compromise the operation of the circuit by changing its behavior. The immediate effects caused by radiation are called single event effects (SEEs). The particular type of SEE that we are interested are the single event upsets (SEUs). SEUs are soft errors, because the device itself is not permanently damaged by radiation, but the radiation event causes enough charge disturbance to reverse or flip the data of a memory cell, register, latch or flip-flop. SRAM-based FPGAs are particularly susceptible to SEUs within the configuration memory and fault tolerance techniques may be applied in order to guarantee the proper operation of the circuit, even in the presence of faults. Figure 1.3 illustrates the misbehavior of a circuit caused by a particle strike.

1.2 Goals

Several works in literature propose fault tolerance mechanisms to mitigate SEUs in LDPC decoders, but none of them has the specific concern with FPGAs implementations. Li

Figure 1.3 - (a) Configuration bits used to a certain logic and routing
(b) Logic and routing are modified due to upsets caused by radiation



(a) (b)

Source: Wirthlin (2015)

et al. (2016), for example, propose to protect parts of the control logic with a Hamming decoder while May et al. (2008) propose to protect some subsets of the circuit with Triple Modular Redundancy (TMR). Both of them are concerned in ASIC implementations, which have a different model of failures. On the other hand, there are several publications aiming to protect FPGAs from SEUs, as in Foster et al. (2010), which propose a technique to define and protect critical subsets of the circuit. Samudrala et al. (2004) describes a technique to predict which cells of the circuit are more sensitive to SEUs by the probability of signal's inputs, while Pratt et al. (2006), in a different manner, has a concern to protect cells that may lead the circuit to a complete faulty state, even when these cells are repaired with configuration scrubbing. All of these publications are focused on general-purpose applications.

This work presents an overview on the effects of radiation in semiconductor devices and FPGAs, FEC codes, LDPC codes as well as a study with the purpose of providing protection from radiation on an FPGA-based LDPC decoder. The Check Nodes (CNs) are considered the most critical element of LDPC decoders, since they execute all arithmetic operations, occupy most of the total area, and have the greatest workload of the circuit. Initially, it was made a coarse-grained analysis in the architecture, evaluating the severity of each CN to the overall performance of the decoder. A fine-grained analysis in the internal structure of the CNs was then made in order to determine which subsets of the circuit are most relevant to the overall sensitivity. The fault injection simulations were executed in a fault injection platform adapted from Leipnitz (2015) to Kintex-7 devices. The architecture evaluated is presented in Hess (2016), who proposes an HDL implementation and a software-based model. The results and the method used to perform fault injections will be shown as well as the LDPC decoder architecture used as reference.

1.3 Structure

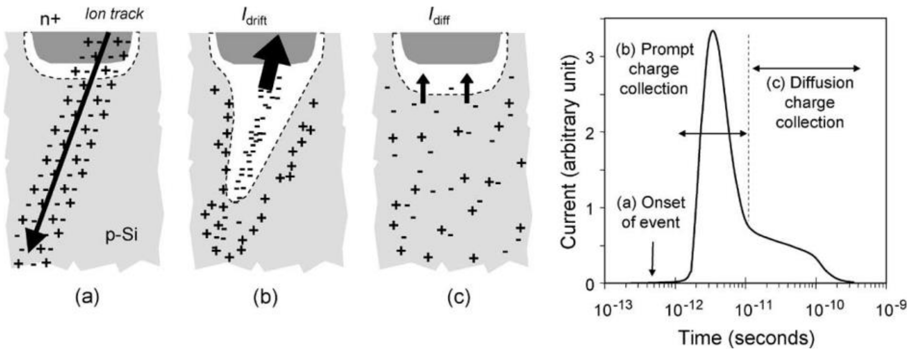
This work is structured as follows: section 2 explains the basic concepts, such as radiation effects on semiconductor devices (2.1), radiation effects on FPGAs (2.2), Forward Error Correction (2.3), LDPC codes (2.4) and related works (2.5). Section 3 details the architecture of the decoder used as reference. Section 4 presents the fault injection method and platform used. Section 5 presents the coarse-grained analysis among CNs and section 6 presents the fine-grained approach, which exploits the internal structure of CNs. Finally, section 7 shows the conclusions of this work.

2 BACKGROUND

2.1 Radiation effects on semiconductor devices

Ionizing radiation can be defined as the transmission of energy through atomic and subatomic particles with very high kinetic energy. It is a natural phenomenon and it is generated from materials on earth, the sun and other cosmic sources (WIRTHLIN, 2015). In space, there is high flow of protons, neutrons, alpha particles and heavy ions that affect semiconductor devices. At ground level, on the other hand, neutrons are the most frequent cause of failures (BAUMANN, 2005). When a single heavy ion strikes the silicon, it loses its energy through the production of free electron-hole pairs, resulting in a dense ionized track in the local region (KASTENSMIDT et al., 2004). Figure 2.1 illustrates the current pulse disturbance caused by this event in a reverse-biased junction.

Figure 2.1 – Collision of an ionized particle and the resulting current pulse



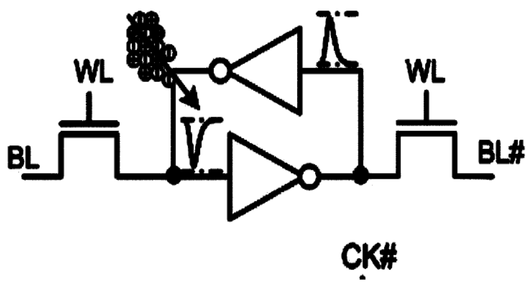
Source: Baumann (2005)

There are several radiation effects in semiconductor devices that vary in magnitude from data disruptions to permanent damage ranging from parametric shifts to complete device failure. Single event effects (SEEs), as the name implies, are device failures induced by a single radiation event. They can cause long-term effects to devices, compromising the operation of the circuit, or simply change the data state in a memory cell, register, latch or flip-flop. SEEs that do not permanently damage the device are called soft errors (BAUMANN, 2005).

Long-term effects include changes in device's electrical parameters, such as the threshold voltage, leakage current and the timing of the MOS transistors. High-energy particles can also displace atoms in the lattice of semiconductor materials, causing permanent damages. The maximum amount of radiation that a device can tolerate before failing its parameters is called total ionizing dose (TID) (BARNABY, 2006).

Single event upsets (SEUs), otherwise, are soft errors. They occur when an ionized particle causes enough charge disturbance to change the voltage level of critical nodes within a memory cell, causing the inversion of the original data stored (changing a logic “1” to a logic “0” or a logic “0” to a logic “1”). This effect occurs due to the feedback nature of these cells, as shown in Figure 2.2. If the radiation-induced voltage happens in a logic gate, that is, a glitch that propagates through combinational circuitry, it is called a single event transient (SET). Single event functional interrupts (SEFIs) are failures that change the internal state of important control registers within a device that control device-level functionality. SEFIs compromise the operation of the application, but they can be resolved by repowering the device and placing it in its initial state. As SEUs, SETs and SEFIs are soft errors (WIRTHLIN, 2015).

Figure 2.2 – Particle strike in a SRAM cell



Source: Monteanu & Autran (2008)

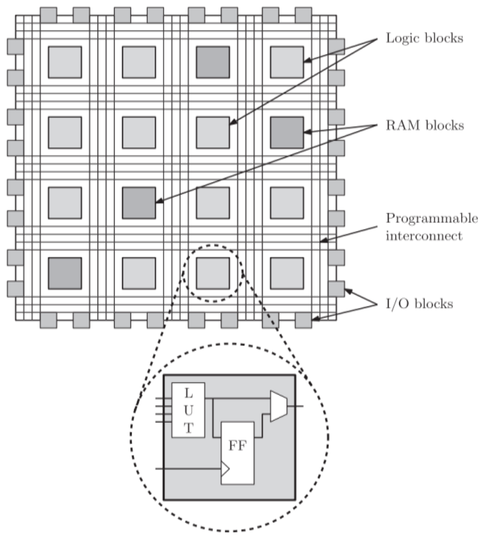
2.2 Radiation effects on FPGAs

With the request of consumer’s demand for device shrinkage, power supply reduction and increasing in the operating frequency, the circuits are more and more susceptible to the effects of radiation. Particularly, there is growing interest in using FPGAs in space and other extreme environments where high-energy radiation is more common than on earth (KASTENSMIDT et al., 2004).

FPGAs are semiconductor devices consisting of logic blocks, RAM blocks and I/O blocks. The most fundamental logic block of an FPGA is formed by a Lookup Table (LUT) and a Flip-Flop (FF). The I/O blocks surround the outer edge of the microchip, providing I/O access to the pins on the exterior of the FPGA package. Programmable routing is implemented so that it is possible to connect logic blocks and IOBs to logic blocks arbitrarily, as shown in Figure 2.3.

Xilinx devices quantify the logic resources in terms of “slices”, each of them contains several LUTs and FFs - the nature and quantity of the hardware resources available in each slice depends on the model and generation of the FPGA. The terminology given by Xilinx also introduces the concept of Configurable Logic Blocks (CLBs), which consists of multiple slices (BUELL et al., 2007).

Figure 2.3 – FPGA structure



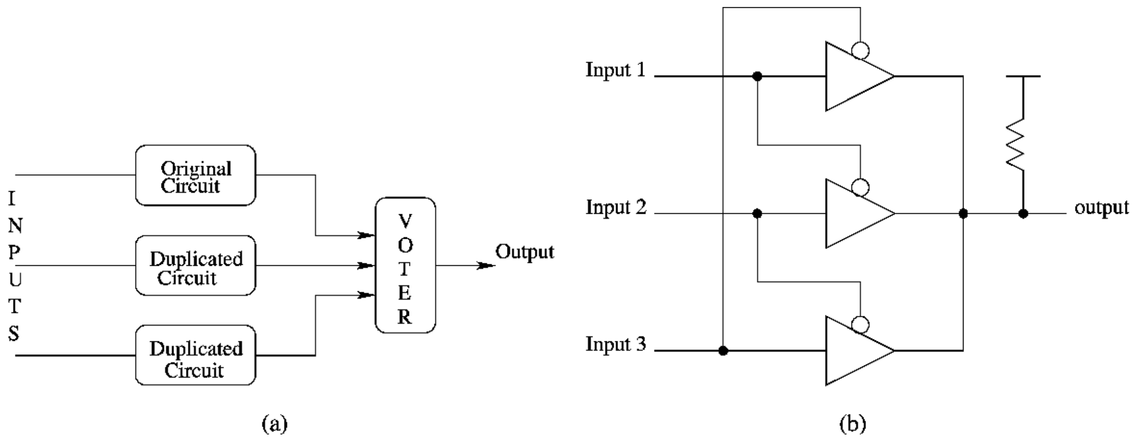
Source**:** Hailes et al. (2015)

To determine the effects of radiation on FPGAs, they will be classified in three categories, based on the technology used to store the configuration data: antifuse, flash and SRAM-based FPGAs. Antifuse FPGAs are nonvolatile and the configuration data cannot be changed once the fuses have been programmed. This type of FPGA is usually the most reliable, since the configuration cells are made from passive, programmed fuses, and they are generally immune to SEEs. Flash FPGAs are also nonvolatile but they may be reprogrammed only for a certain number of times, which may not be suitable for reconfigurable systems requiring frequent reconfiguration. Flash FPGAs have a major concern with SETs through the combinational logic data path and routing resources (STERPONE & DU, 2014).

SRAM-based FPGAs, on the other hand, are volatile, so they lose their configuration when power is removed. Although SRAM cells require more power than antifuse or flash cells, they can be reprogrammed an unlimited number of times. SRAM-based FPGAs have a primary reliability concern in SEUs within the configuration memory, since these cells are made using standard static memory techniques and comprise the majority of the memory cells on the device. The incidence of radiation in the configuration memory may lead to changes in the logic and routing of the operating circuit, deviating from the function they were supposed to fulfill (WIRTHLIN, 2015).

TMR is the most classical and robust technique with the purpose of mitigating SEUs in FPGAs, where a module is replicated three times and the output is extracted from a majority voter, as shown in Figure 2.4 (SAMUDRALA et al., 2004). In this work, Double Modular Redundancy (DMR) is proposed to protect the circuit. It provides error detection by comparing two copies of the circuit. DMR has at least 100% area and power overheads compared to the unhardened design. For TMR, on the other hand, these overheads are at least 200%, making its use very expensive. For applications with stringent cost and power limitations, the application of TMR may not be desirable, making DMR and other less costly error detection mechanisms more attractive choices (NAZAR et al., 2013). The re-writing of the configuration memory is called *scrubbing* and it is the approach proposed to correct the errors detected by DMR in this work.

Figure 2.4 – TMR (a) and voter circuit (b)

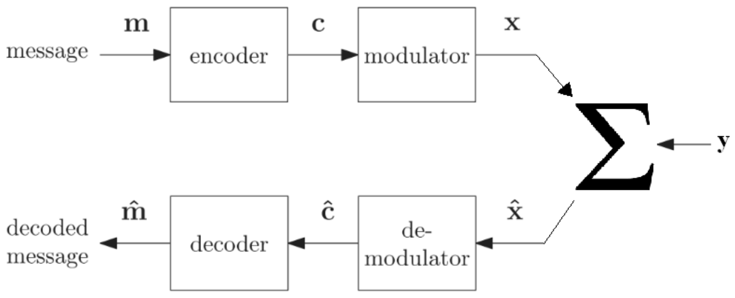


Source: Samudrala et al. (2004)

2.3 Forward Error Correction

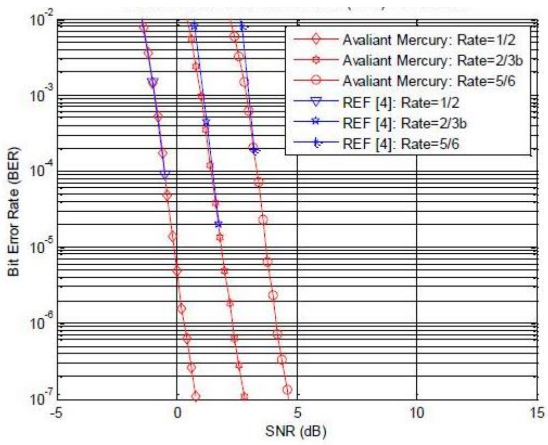
Figure 2.5 shows the scheme of a communication system. The message *m* to be transmitted is composed of binary symbols. The encoder accepts bits of the message and adds redundancy according to a predefined rule, producing data encoded in a higher bit-rate. In order to generate an *(n, k)* block code, the encoder receives blocks of *k* bits. For each block, it adds *n-k* redundant bits and produces a coded block of *n* bits, where $n > k$. The decoder in the receiver side explores this redundancy to decide which bits of the message were indeed transmitted (HAYKIN, 2004). The relation $r = k/n$ is called code rate, where $0 < r < 1$. Figure 2.6 shows BER performance for IEEE 802.16 with different code rates.

Figure 2.5 – Communication system scheme



Source: adapted from Hailes et al. (2016)

Figure 2.6 – BER performance for IEEE 802.16 LDPC using BPSK modulation



Source: Avaliant (2018)

Assuming a Binary Phase-Shift Keying (BPSK) modulation, the modulated symbol vector $\mathbf{x} = \{x_j\}^N_{j=1}$ may be represented through the energy per symbol, $x_j = +\sqrt{E_s}$ when $c_j = 0$ and $x_j = -\sqrt{E_s}$ when $c_j = 1$. Moreover, assuming the Additive White Gaussian Noise (AWGN) channel, $\hat{x}_j = x_j + \mathcal{N}(0, N_0)$, where $\mathcal{N}$ is the normal distribution and N_0 is the noise power spectral density. The relation between E_s and E_b is given by:

$$E_b = \frac{E_s}{r} \tag{2.1}$$

In order to convert received symbols into demodulated bits, Logarithmic-Likelihood Ratio (LLR) is often used. The sign (positive or negative) expresses the most likely value for the corresponding bit (0 or 1) and the magnitude represents the certainty on the value of the bit. The value of the LLR of a demodulated bit is calculated as (2.2), where c_i is the transmitted bit and $\hat{x}$ is the received symbol.

$$\mathrm{LLR}(\hat{c}_i) = \log \frac{P(c_i=0 \mid \hat{x}_i)}{P(c_i=1 \mid \hat{x}_i)} \tag{2.2}$$

Considering a BPSK modulation over an AWGN channel, LLR values can be calculated as follows:

$$\mathrm{LLR}(\hat{c}_i) = 4 * r * \frac{E_b}{N_0} * \hat{x}_i \tag{2.3}$$

In terms of E_S:

$$\mathrm{LLR}(\hat{c}_i) = 4 * \frac{E_s}{N_0} * \hat{x}_i \tag{2.4}$$

2.4 LDPC Codes

2.4.1 Encoding

LDPC codes can be described as a *k*-dimensional subspace C of the vector space of binary *n*-tuples over the binary field $\mathbf{F}_2$. We first describe a basis B = {g_0, g_1, …, g_{k-1}} which spans C. The process of encoding a message is given by 2.3. Each codeword $\mathbf{c} \in$ C can be written as $\mathbf{c} = u_0g_0+u_1g_1+...+u_{k-1}g_{k-1}$, or simply

$$\mathbf{c} = \mathbf{u}G \tag{2.5}$$

where $\mathbf{u} = [u_0, u_1, \ldots, u_{k-1}]$ is the message to be transmitted and G is the *k x n* generator matrix whose rows are the vectors {g_i}.

The *(n – k)*-dimensional null space $C^{\perp}$ of G comprises all the vectors **x** for which $\mathbf{x}G^T = 0$ and is spanned by the basis $B^{\perp} = \{h_0, h_1, \ldots h_{n-k-1}\}$. For each $\mathbf{c} \in$ C, $\mathbf{ch}^T_i = 0$, or simply

$$\mathbf{cH}^T = 0 \tag{2.6}$$

where H is the *(n-k) x n* parity-check matrix whose rows are the vector {h_i} and is the generator matrix for the null space $C^{\perp}$ (RYAN, 2003).

We can obtain G from H by some simple steps. It is necessary first to transform H with Gauss-Jordan elimination in order to get H in the form:

$$H = [A, I_{n-k}] \tag{2.7}$$

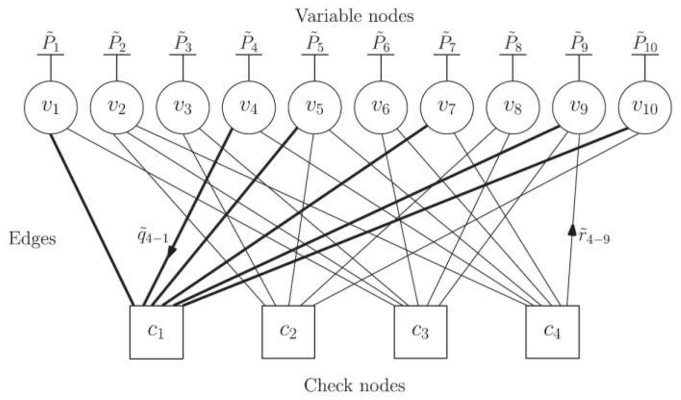
where I is the identity matrix of dimension *n-k* and A is a matrix of size *(n - k) x k*. From that, G can be easily found:

$$G = [I_k, A^T] \tag{2.8}$$

2.4.2 Decoding

LDPC decoding is usually done with belief propagation (BP) algorithm. In this approach, LLR values are passed in both directions along the edges between connected nodes of the factor graph describing the code. An important characteristic of the BP algorithm is that any message sent to a particular node does not depend on the message received from that node. In Figure 2.7, for example, CN c_4 is connected to VNs v_2, v_4, v_5, v_6, v_7 and v_9. The message $\tilde{r}_{4-9}$, however, will be calculated based on the values received from VNs v_2, v_4, v_5, v_6 and v_7.

Figure 2.7 – Factor graph representing messages exchanges in BP algorithm



Source: Hailes et al. (2016)

LDPC decoder's schedule, i.e., the order in which the nodes are activated, has a significant effect upon the error correction capability provided by the decoder. The three most common schedules schemes are Flooding, Layered Belief Propagation (LBP) and Informed Dynamic Scheduling (IDS).

Flooding is the simplest decoding schedule, where each iteration comprises the activation of all CNs simultaneously followed by the activation of all VNs. This approach offers a high degree of parallel processing; however, it demands high area to be implemented.

In LBP schedule, the nodes are processed sequentially within each iteration. Once a CN has been activated, all its connected VNs are activated before moving to the next CN. This schedule results in a lower throughput, higher latency and high complexity per iteration. However, LBP tends to converge to the correct result using fewer iterations, resulting in lower complexity overall when compared to flooding. In addition, a certain level of parallelism can be exploited when using QC codes with this policy. LBP will be more exploited later, since is the schedule used in the decoder used as reference in this work.

IDS verifies the messages passed between the nodes and activates the node that offers the greatest improvement in belief. This schedule requires additional calculations, increasing the complexity per iterations, but the complexity overall is decreased since it demands fewer iterations to achieve the correct output.

2.5 Related Works

Several works in literature propose techniques to mitigate SEUs in LDPC decoders. Li et al. (2016), for instance, proposes the design of a fault-tolerant LDPC decoder that corrects soft-errors caused by SEUs inside the control logic with a Hamming decoder. For RAM cells, a layered pipelined architecture is proposed as well as a scheme that detects soft errors by parity check, then the errors will be corrected by the inherent decoder's iterative process. This approach could save 42% of cell area compared with TMR method and the reduction of 12% of memory bits compared with similar works.

The work presented in May et al. (2008) proposes a technique that assumes that not all data bits of a message or channel value have the same importance and corruption in higher significant bits has a larger impact on the overall communications performance than corruption in lower bits. If an LLR value which is calculated by a functional node is corrupted, the value is reset to 0. In other words, the corresponding node/edge is temporarily removed for the current iteration. Thereby, no information is associated with the respective bit and the error tends to be minimized.

In order to protect FPGAs from the effects of SEUs, several publications have proposed alternatives to the traditional TMR method by protecting specifics subsets of the circuit, the ones considered most critical. Foster et al. (2010) presents several methodologies for selecting these subsets, such as metrics that consider the number of logic cells that use the cell’s output signals as inputs, the number of logic resources necessary to add TMR to the logic cell and the number of logic cells in longest propagation path through the logic cell.

Another approach is presented in Samudrala et al. (2004), where a Selective Triple Modular Redundancy (STMR) technique is described. The logic cells are classified by the “sensitivity” to SEUs, which is measured by the signal probability of its inputs. It is assumed that the primary inputs of the circuit are specified by the user in terms of signal probabilities and then it is propagated to compute the signal probability of each internal node.

In a different way, Pratt et al. (2006) prioritizes the protection of structures causing “persistent” errors within the design. Configuration bits are categorized in “persistent” and “non-persistent”. A non-persistent configuration bit will cause a design fault when upset and may be repaired through configuration scrubbing, which will lead the design back to normal operation. Persistent bits, on the other hand, will also cause a design fault when upset, but after repairing persistent bits through configuration scrubbing, the FPGA circuit does not return to normal operation.

Unlike the other works focused on LDPC decoders, this work specifically targets LDPC in FPGAs and the associated failure model, which is different from ASICs. In addition, unlike the other works in partial redundancy for FPGAs, this work is specific to an application, that is, the impact of the failures in metrics relevant to the application in question will be taken into account.

3 LDPC DECODER ARCHITECTURE

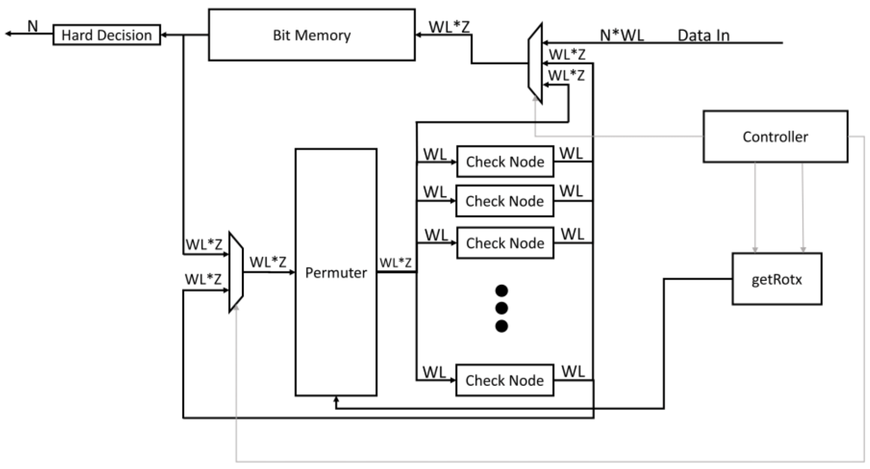
LBP is often used in LDPC decoders since it can achieve effectively high decoding throughput with low computation complexity (CUI et al., 2012). In this policy, the parity check matrix is viewed as horizontal layers, each layer represents a component code and the message updating is performed layer by layer. The model presented in Hocevar (2004) consists in a memory used to store the bit values of the message to be decoded, *Parity Check Update Blocks* (PCUBs) and *Bit Update Blocks* (BUBs), which implement the message exchange calculations, as well as a router and a reverse router to arrange data as needed by the algorithm and architecture.

Hess (2016) implemented an HDL description and a bit-accurate software simulator of a modified version of the architecture presented in Hocevar (2004), as shown in Figure 3.1. *Bit Memory* is used to store the LLR values of the message bits, the *Controller* implements a finite state machine and is responsible for controlling the data flow among the other components of the circuit, and the *Check Nodes* perform the message exchange calculations. It was used a single router, called *Permuter*, since the output of the CNs will be routed only in the processing of the last row of the matrix. The *Permuter* rotates the values from the Bit Memory by the value obtained from *GetRotX*, in order to deliver the correct inputs to CNs.

The algorithm used to decode the messages is a modified version of Min-Sum algorithm (KARKOOTI et al., 2008). It calculates two messages: one that considers the smallest value among all VNs connected, and another that considers the second smallest value. The first message is sent to all VNs connected and the second message is sent to the VN connected that owns the smallest value. A correction factor ($\beta > 0$) is applied to the calculation of the messages aiming to reduce the information loss due to the simplified function that is used. Besides the advantages brought by the original version of the algorithm, as the reduced complexity to calculate the messages and the small area necessary, the modified version offers a greater energy efficiency. The Modified Min-Sum algorithm, along with the Layered Decoding policy, is described in Algorithm 1.

The internal structure of the CN is represented in Figure 3.2. Initially, the subtractor (*Sub*) receives LLR values from bit r_j and from the values stored in the internal memories. *MagMem* stores the magnitude of the messages with lower values, *SignMem* contains the signals and *IdxMem* contains the indexes of these messages.

Figure 3.1 – Modified Layered Decoding architecture



Source: Hess (2016)

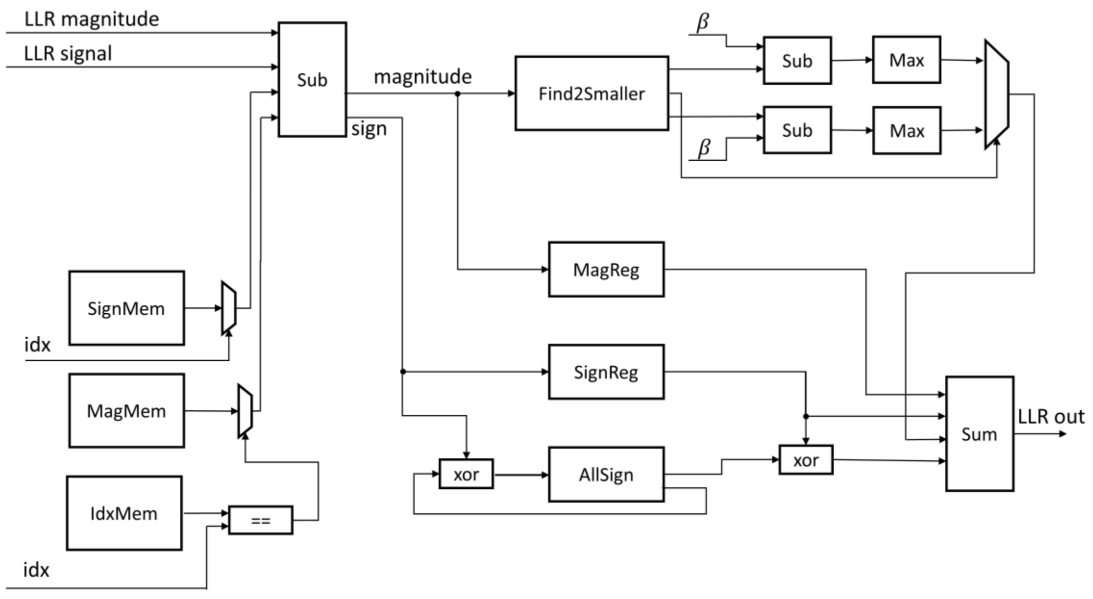
Algorithm 1 – LD-MMS

```
Input: r, max_iterations
Output: $\hat{s}$
begin
    $iterations = 0$
    while $iterations < max\_iterations$ do
        for $j = 1 : M$ do
            for $i \epsilon B_j$ do
                // Messages from VNs
                $M_{i,j} = r_i - E_{j,i}$
                // Messages from CNs
                $E_{j,i} =$
                $\prod_{i' \epsilon B_{j,i' \neq i}} sign(M_{j,i'}) * max(min_{i' \epsilon B_{j,i' \neq i}}(\|M_{j,i'}\|) - \beta, 0)$
                // Updated VNs values
                $ri = M_{i,j} + E_{j,i}$
            end
        end
        $iterations = iterations + 1$
    end
    for $i = 1 : N$ do
        $L_i = \sum_{j \epsilon A_i} E_{j,i} + r_i$
        $\hat{s}_i = \begin{cases} 1, L_i \leq 0 \\ 0, L_i > 0 \end{cases}$
    end
end
```

Source: Hess (2016)

The magnitude of the signal generated by *Sub* is sent to a register (*MagReg*) and to *Find2Smaller*, which calculates the two smallest values among all. Both values are subtracted by the correction factor *β* and are compared with 0 in *Max*. The decision by which value to use is made by a signal from *Find2Smaller*. *AllSign* stores the sum of all signs and, by applying XOR with the sign from previous iteration (stored in *SignReg*), we have the correct sign. Finally, *Sum* calculates the output of the CN based on the values from previous iteration and the value given by *Find2Smaller*.

Figure 3.2 – Architecture of a *Check Node*



Source: Hess (2016)

4 FAULT INJECTION METHOD

Fault injection methods can be seen as techniques for testing systems with respect to the effects of faults on their behavior. They are applicable when it is not possible to get statistical data from field operation or when preliminary studies of the behavior of the system in the presence of faults should be considered in development phase. Moreover, it can identify implementation errors in fault-tolerance mechanisms and provide feedback on those mechanisms' efficiency (CLARK & PRADHAN, 1995; ARLAT et al., 1990).

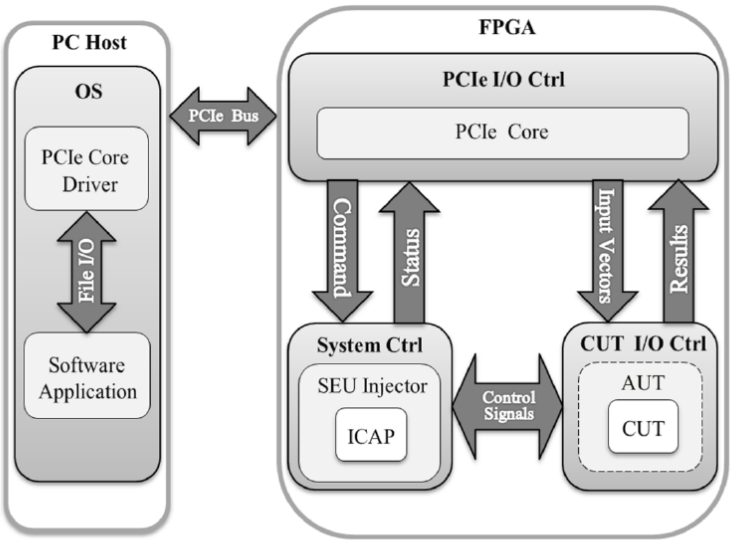
This technique may be categorized in five main groups: hardware, software, simulation, emulation and hybrid-based fault injection. Hardware-based approach is accomplished when applied at physical level, by disturbing the IC itself with environments parameters (such as heavy ion radiation, electromagnetic interferences or power supply disturbances). Software-based fault injection consists of a software implementation reproducing the errors that would have been produced upon occurring faults in the hardware. In a different way, simulation-based approach consists of injecting faults in high-level models (usually HDL models) and emulation-based fault injection, which is applied in this work, takes advantage of FPGAs for effective circuit emulation and speeding-up fault simulation. Hybrid-based approaches mix software-implemented fault injection and hardware monitoring (ZIADE et al., 2004).

Leipnitz (2016) proposes a fault injection platform, focused on communication systems, for a Virtex-5 device to emulate SEUs, whose structure is shown in Figure 4.1. The approach presented consists in an FPGA device connected to a host computer, used to define the fault injection campaign, control the injection experiments, display the results and communicate with the board though the PCIe interface. The hardware emulated by the FPGA contains a module called *PCIe I/O Ctrl* to perform communication with the host computer. *System Ctrl* is responsible for performing fault injection/removal in the configuration memory and *CUT I/O Ctrl* performs the execution of the circuit under test.

It was used a modified version of the platform presented by Leipnitz (2016) to be synthesized in Kintex-7 devices, since the original version was designed for Virtex-5 devices. The overall structure of the platform remains the same, but several adaptations had to be done due to differences in the communication bus, configuration memory addressing, interface with internal ports and circuit placement.

The PCIe interface used to perform communication between the FPGA and computer host is implemented with Xillybus system, developed by Eli Billauer (2014). It consists of an

Figure 4.1 – Fault injection platform structure proposed by Leipnitz (2016)



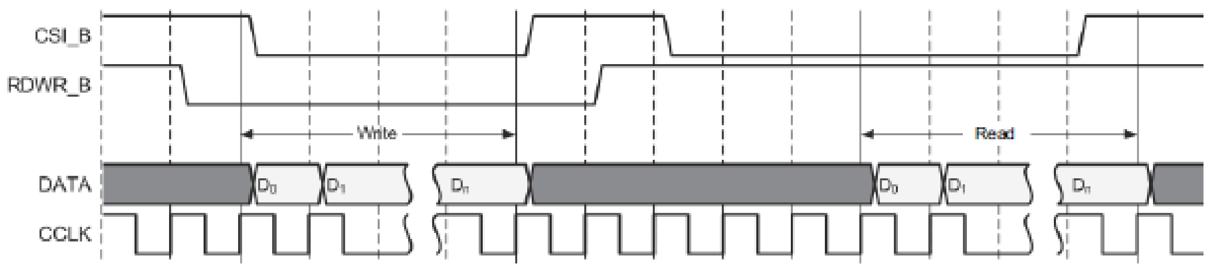
Source: Leipnitz (2016)

IP core and a host driver and provides low latency, full-duplex communication and data rates between 200MB/s and 800MB/s. The Virtex-5 device used in the original version has a PCIe 1.0 x1 and can reach at most 250MB/s, which limits the communication rate. Kintex-7 device has a PCI-express 2.0 x4 that achieves 2GB/s, which makes the upper bound rate defined by the IP core. Despite changing the IP core module and the driver in the host computer, several parameters and internal buffers had to be altered due to the different number of lanes in the bus as well as the placement of PCIe ports.

To emulate the effects of SEUs in the FPGA's configuration memory, bit-flips in the memory content are performed. The access is made through the Internal Configuration Access Port (ICAP), a mechanism provided by Xilinx devices which allows accessing the configuration memory within the device and guarantees faster fault injection/removal times. The original version of the platform treats a signal called *busy* from ICAP to get readback data. Kintex-7 devices counts with ICAPE2, a newer version of the interface, in which this signal was discontinued and readback data is synchronous with other signals, as illustrated in Figure 4.2.

The configuration memory is divided into frames that can be accessed with an addressing scheme, as shown in Figure 4.3. The FPGA structure is divided in rows, which may be in the top half or in the bottom half of the lattice. Each row is numbered from 0, starting from the center. The rows are divided into columns, which may be either a CLB, DSP, block RAM or IOB block. The columns are numbered from 0, starting from the left. Each column contains a certain number of frames, depending on the block type, as shown in Table 4.1.

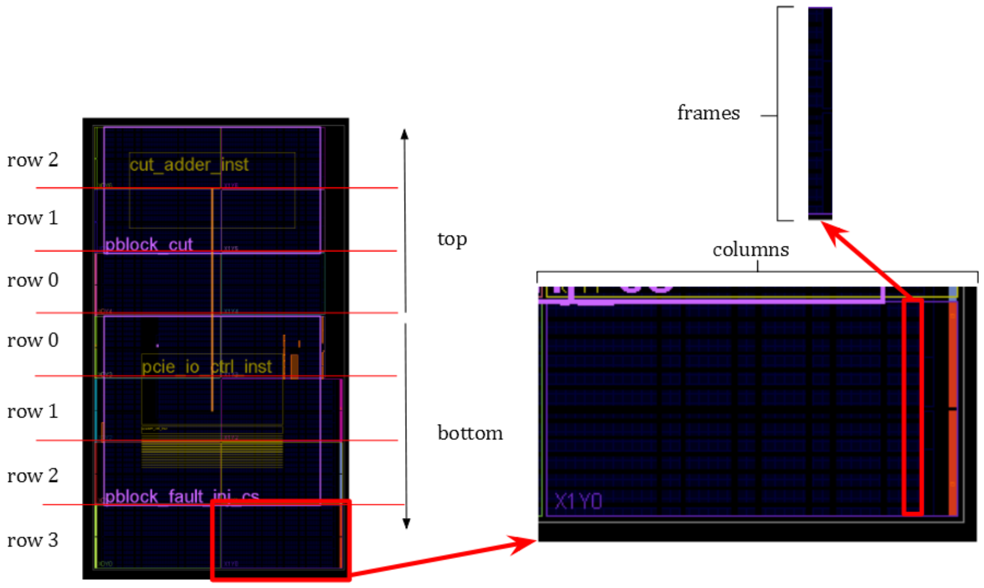
Figure 4.2 – Readback data is available deterministically three clock cycles after CSI_B is set to 0 in Kintex-7 devices



Source – Xilinx UG470 (2016)

System Ctrl had to be modified to match the addressing scheme of Kintex-7 devices, which contains 3232 bits per frame, against 1312 bits in Virtex-5 devices. The word size remains the same (32 bits), but the words per frame has increased from 41 to 101 from one FPGA to the other.

Figure 4.3 – Frame addressing scheme of Kintex-7 devices



Source: the author

Table 4.1 – Number of frames per block type

Block type	**Number of frames**
CLB	36
DSP	28
Block RAM	30
IOB	54
Clock	4

Source: Xilinx UG470 (2016)

5 COARSE-GRAINED REDUNDANCY

This section presents a coarse-grained analysis in the CNs in order to identify the ones that are most critical to the overall system. By identifying the CNs that cause greater impact on the decoder's BER performance, it is possible to protect them with a selective redundancy technique.

The simulations made in this work take advantage of QC code construction to parallelize the processing of multiple CNs and reduce the hardware area occupied. The codeword length (*n*) is 648, the code rate (*r*) is 0.5 and the parity-check matrix is formed by square submatrices of length 27 (*Z*). Since each row is connected to a single column in the same submatrix, it is possible to process the rows of each submatrix in parallel. The HDL model implements 27 physical CNs, each one corresponding to a row in a submatrix (code-level CNs), which are activated simultaneously. Table 5.1 shows the mapping between physical and code-level CNs, (an iteration corresponds to the processing of a submatrix).

Table 5.1 – Physical CNs x code-level CNs mapping

Physical CN index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	Iteration index
Code CN index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	0
	27	28	29	30	31	32	33	34	35	36	37	38	39	40	1
	54	55	56	57	58	59	60	61	62	63	64	65	66	67	2
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
	297	298	299	300	301	302	303	304	305	306	307	308	309	310	11

Physical CN index	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	Iteration index
Code CN index	14	15	16	17	18	19	20	21	22	23	24	25	26	0
	41	42	43	44	45	46	47	48	49	50	51	52	53	1
	68	69	70	71	72	73	74	75	76	77	78	79	80	2
	...	...	...	...	...	...	...	...	...	...	...	...	...	...
	311	312	313	314	315	316	317	318	319	320	321	322	323	11

Source: the author

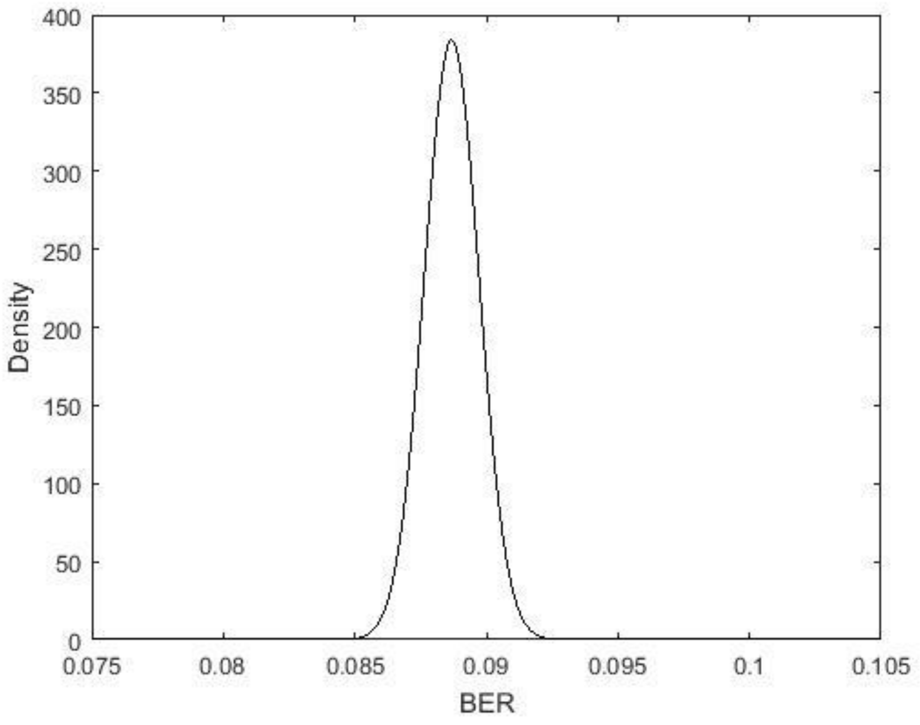
Each physical CN was placed in a faulty-state in the decoder’s software model. To simulate the faults, the CN’s outputs had the magnitudes attenuated or increased by a random factor and the signal was changed depending also of a random factor. Note that a fault induced in the first physical CN, for instance, corresponds to 12 code-level CNs (of indexes 0, 27, 54, etc.) operating inappropriately. The BER measured for a fault-free execution was 0.00011, Table 5.2 shows the BER performance for each CN in a faulty-state and Figure 5.1 shows the corresponding normal distribution (the average is equal to 0.08867 and the standard deviation is 0.00103).

Table 5.2 - BER per physical CN over failure

Physical CN index	BER	Physical CN index	BER	Physical CN index	BER
0	0.088034	9	0.087201	18	0.088772
1	0.088035	10	0.089398	19	0.091276
2	0.088644	11	0.089682	20	0.08881
3	0.086128	12	0.088972	21	0.089552
4	0.089682	13	0.089432	22	0.087181
5	0.088052	14	0.089139	23	0.08811
6	0.088424	15	0.088645	24	0.088412
7	0.088528	16	0.08779	25	0.090668
8	0.08927	17	0.087991	26	0.088508

Source: the author

Figure 5.1 – Normal distribution for BER per physical CN over failure

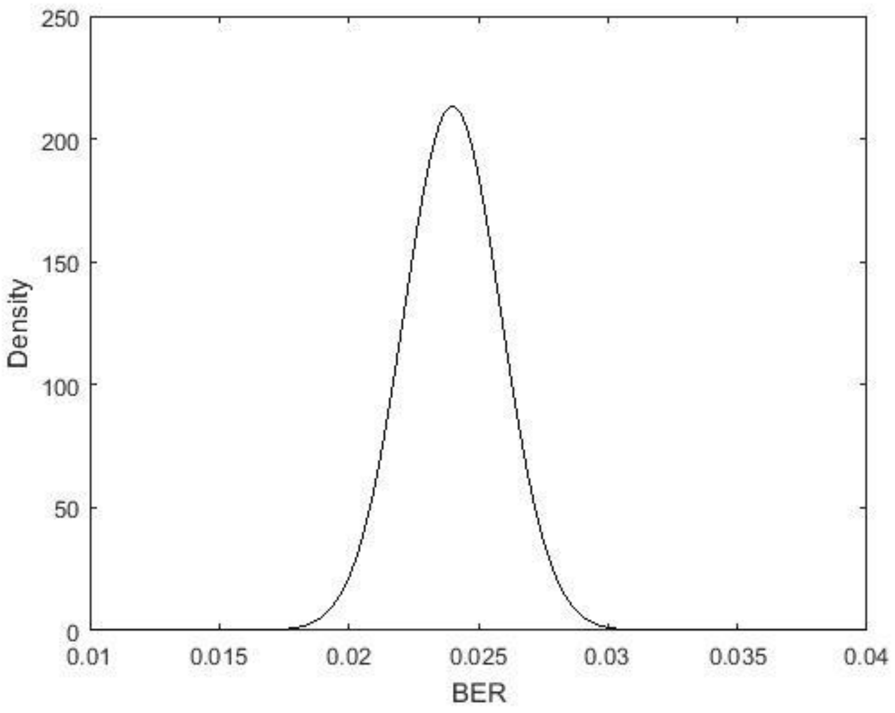


Source: the author

Given these results, we can conclude that all the physical CNs are similarly critical for proper system behavior, since the BER performance among them is within the same order-of-magnitude and is highly degraded compared to a fault-free execution.

The code-level CNs were then evaluated in the overall system's performance. The same approach to simulate the faults was adopted, but this time individually for each of the 324 code-level CNs. The average obtained for BER is equal to 0.0240 and the standard deviation is 0.00187. Note that the code-level CNs are individually less relevant to the system performance than the physical CNs, which was already expected since each physical CN corresponds to twelve faulty code-level CNs. Yet, due to the low standard deviation of BER performance among them and high degradation compared to a fault-free execution, they are similarly critical for proper system behavior, as illustrated in Figure 5.2.

Figure 5.2 – Normal distribution for BER per code-level CN over failure



Source: the author

6 FINE-GRAINED REDUNDANCY

The next step is to evaluate the internal structure of the CN in order to compare the performance loss, in terms of BER increase, caused by each module in a faulty-state and the area overhead obtained by applying selective redundancy. Here, we assume that DMR will be applied to detect the errors, *scrubbing* will be applied to correct the errors and the signals will be reprocessed.

Fault injections were performed in the components of a single CN for LLR values with E_b/N_0 = 2.5 dB and the results were obtained for twenty input blocks of the LDPC decoder. Each input block of the decoder produces 440 inputs and 525 outputs in the CN, totaling 8800 inputs and 10500 outputs in the CN. The total amount of 51712 bits of the configuration memory were affected and 22536 (43,58%) produced some error in the CN’s output. The fault injections were performed in all configuration bits associated with logic and routing. Memory elements (BRAMs) were not evaluated since these components have a different model of failures and may be protected with different fault tolerance techniques, such as error-correcting codes. Table 6.1 shows the distribution of sensitive bits, LUTs and FFs on the modules evaluated.

Table 6.1 – Number of sensitive bits, LUTs and FFs per module of the CN

Module	Sensitive bits	LUTs	FFs
SUM	5422	50	17
SUB	4929	50	17
SUB_BETA_1	3267	50	17
Find2Smaller	3076	22	23
SUB_BETA_0	2793	50	17
AllSign	2416	3	0
MAX_0	529	7	0
MAX_1	104	7	0
Total	22536	239	91

Source: the author

The CN’s outputs may be either correct or categorized within one or more classes of errors. The errors with respect to the value produced are:

- Change of sign;
- Increase in magnitude;
- Decrease in magnitude;

If the behavior of the circuit is affected, the errors may be classified in:

- Control errors (regarding the signals that control the behavior of the CN);
- Timeout (if the CN does not produce any output).

Table 6.2 shows the average of errors caused by sensitive bits per module of the CN. For example, each sensitive bit of *Find2Smaller* produced, on average, 16.8% outputs with wrong signal.

Table 6.2 – Average of errors caused by sensitive bits per module of the CN

Module	Change of sign	Increase in magnitude	Decrease in magnitude	Control errors	Timeout	Correct outputs
Find2Smaller	16.8%	43.6%	19.3%	2.9%	2.3%	24.3%
MAX_0	29.7%	26.6%	32.2%	5.2%	1.1%	31.0%
MAX_1	39.9%	28.0%	28.2%	10.4%	3.2%	29.5%
SUB	31.9%	37.2%	4.6%	31.4%	2.2%	22.3%
SUB_BETA_0	28.3%	32.6%	24.1%	8.3%	2.0%	30.8%
SUB_BETA_1	17.2%	31.6%	31.6%	8.4%	2.3%	24.1%
SUM	53.2%	1.1%	44.6%	23.0%	0.3%	14.8%
AllSign	27.0%	5.9%	12.3%	5.2%	3.9%	51.4%

Source: the author

The behavior obtained in the fault injections, shown in Table 6.2, was replicated in the software model in order to evaluate the impact of the faults in the decoder's BER performance. The results are presented in Table 6.3.

Table 6.3 – BER performance for each module under a faulty-state

Module	BER
SUB	0.238585
SUM	0.238522
Find2Smaller	0.231251
SUB_BETA_0	0.101422
MAX_0	0.046245
SUB_BETA_1	0.001991
MAX_1	0.000411
AllSign	0.000245

Source: the author

We can define the *Weighted Performance Degradation (WPD)* of a module M as the product of the relative quantity of sensitive bits in M by the increase in BER performance produced by M:

$$WPD(M) = \frac{sensitive\ bits\ in\ M}{total\ sensitive\ bits\ in\ the\ CN} * \frac{BER\ for\ M\ under\ fault}{BER\ for\ fault\ free} \quad (6.1)$$

The *Area Overhead (AO)* of the module M is simply the hardware area occupied by M since we are applying DMR, i.e., the sum of LUTs and FFs:

$$AO(M) = LUTs(M) + FFs(M) \quad (6.2)$$

The ratio of the *Weighted Performance Degradation* to the *Area Overhead* is defined as the *Gain* and is a quantity we want to maximize:

$$Gain(M) = \frac{WPD(M)}{AO(M)} \quad (6.3)$$

Table 6.4 shows the metrics (6.1), (6.2) and (6.3) for each module of the CN.

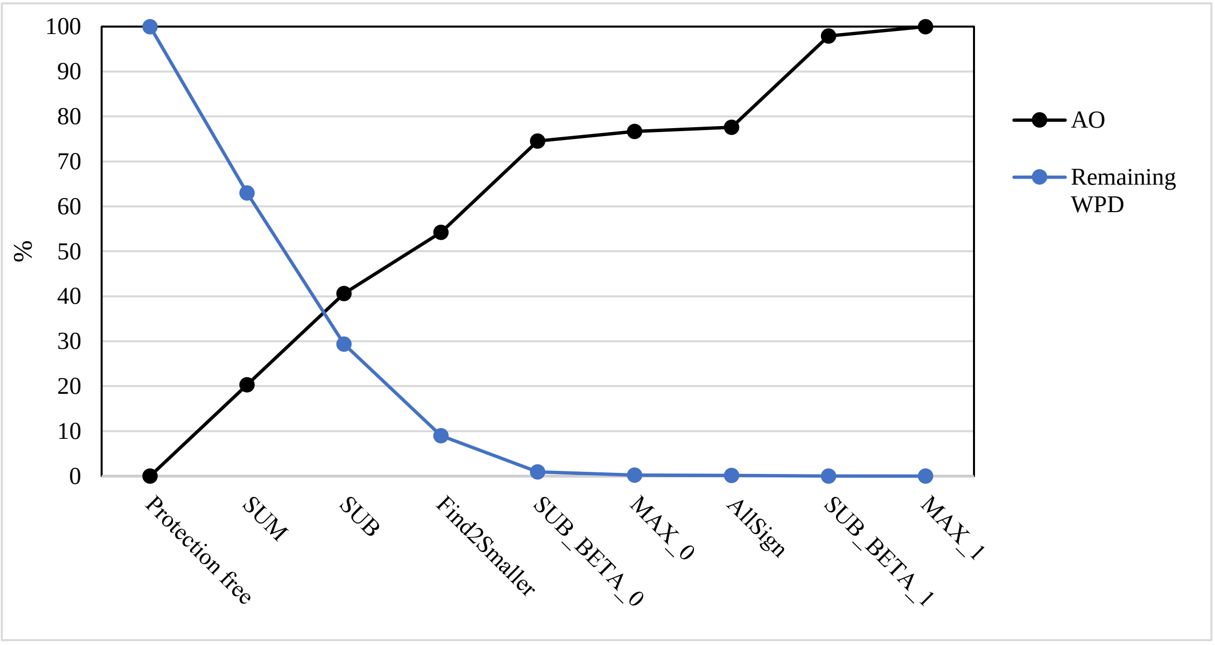
Table 6.4 – WPD, AO and Gain for each module of the CN

Module	WPD	AO	Gain
SUM	438.066	67	6.538
SUB	398.340	67	5.945
Find2Smaller	240.947	45	5.354
SUB_BETA_0	95.952	67	1.432
MAX_0	8.287	7	1.184
AllSign	0.201	3	0.067
SUB_BETA_1	2.203	67	0.033
MAX_1	0.014	7	0.002

Source: the author

Figure 6.1 illustrates the scenarios of applying cumulative DMR in the modules of the CN. The x-axis is ordered by the gain (shown in Table 6.4) and each module represented in the x-axis incurs in the protection of itself and all the others to the left. Observe that by applying redundancy in *SUM*, *SUB* and *Find2Smaller*, the area would be increased about 55% and the remaining WPD would be about 10%, which means that 90% of the possible WPD would have been achieved. The protection of *SUB_BETA_1* and *MAX_1* has a high cost in area, the WPD obtained is low and if the area occupation is a major concern to the hardware designer, it would probably not be worthy.

Figure 6.1 – Cost benefit scenarios of applying DMR cumulatively in the modules of the CN



Source: the author

7 CONCLUSIONS

In this work, we have presented a study about the effects of SEUs in an FPGA-based LDPC decoder and proposed a selective technique to improve reliability in this specific application.

It was shown the destructive and non-destructive effects and models of failures of radiation in semiconductor devices. The structure of FPGAs was presented and these devices were categorized by the technology used to store the configuration data. SRAM-based FPGAs were best exploited since they are especially susceptible to SEUs and were targets of this work. Regarding the application evaluated, an overview of FEC codes and communication systems was presented as well as the processes to encode and decode a message with LDPC codes. The decoding encompasses not only the algorithm used (belief propagation), but also the advantages and disadvantages of different schedules that may be applied.

The architecture and algorithm used as reference was shown and the structure of a CN was exploited, since it is the most critical element of LDPC decoders (they execute all arithmetic operations, occupy most of the total area, and have the greatest workload of the circuit). Fault injections were performed in the HDL modules and a bit-accurate software model was used to obtain BER performance for the analyses taken.

It was used a modified version of the platform described in Leipnitz (2016) to Kintex-7 devices and several adaptations had to be done due to differences in the communication protocol to perform partial reconfiguration, communication bus with the host computer and frame addressing of the configuration memory.

The first approach to propose a selective redundancy to the circuit was made in CN-level, by identifying the CNs that cause greater impact on the decoder's BER performance. Initially, it was analyzed the 27 physical CNs, by placing each one in a faulty-state in the decoder's software model. Then, a similar approach was applied, but this time for the 324 code-level CNs. The results have shown that both physical and code-level CNs are equally critical for proper system behavior.

The internal structure of the CN was then analyzed and the results have shown that *SUB, SUM* and *Find2Smaller* are the modules that cause greater impact in the decoder’s BER performance. Within the metrics created to evaluate the application of DMR, *SUM* is the module that provides the best gain when protected, *SUB_BETA_1* and *MAX_1* are the ones that provide the smaller contribution in the decoder’s BER performance when we consider their area occupation. Nevertheless, there are several parameters that must be taken into account by a hardware designer to develop the best solution for the application's needs. If the concern is with the type of error that the modules are more susceptible to produce instead with area occupation, for example, the protection of these modules may be considered to best fit in the project's requirements.

REFERENCES

GALLAGER, R. G. Low-density parity-check codes. Information Theory, **IRE Transactions on communications**, IEEE, v. 8, n. 1, p. 21–28, 1962.

MACKAY, D. J.; NEAL, R. M. **Near Shannon limit performance of low-density parity check codes.** In: . [S.l.]: [Stevenage, etc., Institution of Electrical Engineers], 1996. v. 32, n. 18, p. 1645–1646.

TANNER, R. M. **A recursive approach to low complexity codes.** In: [S.l.]: IEEE, 1981. v. 27, n. 5, p. 533–547.

BUELL, Duncan; EL-GHAZAWI, Tarek; GAJ, Kris; KINDRATENKO, Volodymyr. **High-performance reconfigurable computing.** Computer, IEEE Computer Society, vol. 40, no. 3, pp.23–27, March 2007. [Online] Available: http://www.computer.org/csdl/mags/co/2007/03/r3023.pdf.

HAILES, P. et al. **A survey of FPGA-Based LDPC Decoders**. IEEE Communications Surveys & Tutorials, IEEE, v. 18, n. 2, p. 1098–1122, 2015.

WIRTHLIN, Michael. **High-reliability FPGA-based systems: space, high-energy physics, and beyond.** IEEE Proceedings, vol. 103, no. 3, pp. 379–389, March 2015.

CARRASCO, Rolando; JOHNSTON, Martin. **Non-Binary Error Control Coding for Wireless Communication and Data Storage.** Pages 201-235. 2009. ISBN: 978-0-470-51819-9

RYAN, William. **An Introduction to LDPC Codes**. CRC Handbook for Coding and Signal Processing for Recording Systems, B. Vasic, ed., CRC Press.

JÚNIOR, Geferson L. H. **Implementação e caracterização de falhas em um decodificador LDPC**. December 2016.

PRATT, Brian; CAFFREY, Michael; GRAHAM, Paul; MORGAN, Keith; WIRTHLIN, Michael. **Improving FPGA Design Robustness with Partial TMR**. *IEEE International Reliability Physics Symposium Proceedings*, San Jose, CA, 2006, pp. 226-232. DOI: 10.1109/RELPHY.2006.251221

FOSTER, David L.; HANNA, Darrin M. 2010. **Maximizing area-constrained partial fault tolerance in reconfigurable logic**. *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*. ACM, New York, NY, USA, 259-262. DOI: http://dx.doi.org/10.1145/1723112.1723155

SAMUDRALA, Praveen K.; RAMOS, Jeremy; KATKOORI, Srinivas. **Selective triple Modular redundancy (STMR) based single event upset (SEU) tolerant synthesis for FPGAs**. *IEEE Transactions on Nuclear Science*, vol. 51, no. 5, pp. 2957-2969, Oct. 2004. DOI: 10.1109/TNS.2004.834955

HOCEVAR, D. E. **A reduced complexity decoder architecture via layered decoding of LDPC codes**. In: IEEE. Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on. [S.l.], 2004. p. 107–112.

LEIPNITZ, M. T.; HESS, G. L.; NAZAR, G. L. **A fault injection platform for fpga-based communication systems.** In: IEEE. 2016 IEEE 7th Latin American Symposium on Circuits & Systems (LASCAS). [S.l.], 2016. p. 59–62.

MAY, M.; ALLES, M.; WEHN, N. **A case study in reliability-aware design: a resilient LDPC code decoder**. In: ACM. Proceedings of the conference on Design, automation and test in Europe. [S.l.], 2008. p. 456–461.

BAUMANN, Robert C**. Radiation-induced soft errors in advanced semiconductor technologies**. Device and Materials Reliability, IEEE Transactions on, vol. 5, no. 3, pp. 305-316, Sept. 2005.

MUNTEANU, D.; AUTRAN, J. L**. Modeling and simulation of single-event effects in digital devices and ICs**. Nuclear Science, IEEE Transactions on, vol. 55, no. 4, pp. 1854— 1878, Aug 2008.

SHANNON, C. E. **A mathematical theory of communication**. Bell System Technical Journal. The, vol. 27, no. 3, pp. 379-423, July 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

KASTENSMIDT, F.G.; NEUBERGER, G.; HENTSCHKE, R.F. **Designing fault-tolerant techniques for SRAM-based FPGAs**. IEEE Design & Test of Computers, vol. 21, no. 6, pp.552-562, Nov 2004. DOI: 10.1109/MDT.2004.85

BARNABY, H.J. **Total-Ionizing-Dose Effects in Modern CMOS Technologies**. IEEE Transactions on Nuclear Science, vol. 53, no. 6, pp. 3103-3121, Dec 2006. DOI: 10.1109/TNS.2006.885952

HAYKIN, Simon. **Communication Systems**. 4th Edition. John Wiley & Sons Inc., 2004. 840 p.

CUI, Z.; WANG, Z.; ZHANG, X. **Reduced-complexity column-layered decoding and implementation for LDPC codes**. IET Communications, vol. 5, no. 15, pp. 2177-2186, Oct 2011. DOI: 10.1049/iet-com.2010.1002

KARKOOTI, M.; RADOSAVLJEVIC, P.; CAVALLARO, J. R. **Configurable LDPC Decoder Architecture for Regular and Irregular Codes**. Springer Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology, vol. 53, no. 1-2, Nov 2008. DOI: 10.1007/s11265-008-0221-7

CLARK, J.A.; PRADHAN, D.K**. Fault injection: a method for validating computer-system dependability**. IEEE Computer, vol. 28, no. 6, pp. 47-56, Jun 1995. DOI: 10.1109/2.386985

ARLAT, J.; AGUERA, M.; AMAT, L.; CROUZET, Y.; FABRE, J.C.; LAPRIE, J.C.; MARTINS, E.; POWELL, D. **Fault injection for dependability validation: a methodology and some applications**. IEEE Transactions on Software Engineering, vol. 16, no. 2, pp. 166-182. DOI: 10.1109/32.44380

ZIADE, H.; AYOUBI, R.; VELAZCO, R. **A Survey on Fault Injection Techniques**. The International Arab Journal of Information Technology, vol. 1, no. 2, Jul 2004

BILLAUER, Eli. **An FPGA IP core for easy DMA over PCIe with Windows and Linux**. 2014 [Online]. Available: http://xillybus.com.

XILINX, Inc. **7 Series FPGAs Configuration User Guide**. UG470, Aug 2018 [Online]. Available:
https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf

LI, B.; PEI, Y.; GE, N.; **Area-Efficient Fault-Tolerant Design for Low-Density Parity-Check Decoders**. 2016 IEEE 84th Vehicular Technology Conference, Sep 2016. DOI: 10.1109/VTCFall.2016.7880909

AVALIANT. **Case Study: Performance Results of Avaliant Mercury**. Sep 2018 [Online]. Available:
https://static1.squarespace.com/static/553e7ab4e4b07293cf6dd681/t/59120dfaff7c507ca01ff4e3/1494355459355/LDPC_Case_Study_v6.pdf

NAZAR, G.L.; SANTOS, L.P.; CARRO, L. **Accelerated FPGA Repair through shifted scrubbing**. 23rd International Conference on Field programmable Logic and Applications. Porto, Portugal. Sep 2013. DOI: 10.1109/FPL.2013.6645533

STERPONE, L.; DU, B. **Analysis and mitigation of single event effects on flash-based FPGAS.** 19th IEEE European Test Symposium (ETS). Paderborn, Germany. May, 2014. DOI: 10.1109/ETS.2014.6847804

APPENDIX A – GRADUTATION PROJECT I

Reliable FPGA-based LDPC Decoder

Eduardo Nunes de Souza

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

ensouza@inf.ufrgs.br

Abstract. *LDPC codes are extremely advantageous, both from the theoretical point of view – which makes them attractive to the academic community – and from the applicability perspective, which justifies their wide use in data communication applications. In particular, FPGAs are very timely to the implementation of these codes: the high level of parallelism given by these devices combined with the efficiency of LDPC codes ensures an elevated performance of these applications. In critical systems, the data reliability is a major requirement and, in such cases, as communication satellites, these devices are exposed to a high incidence of ionizing radiation, which may lead to failures. In this sense, this work presents a study about LDPC codes and the effects of soft errors in FPGAs. To workaround this issue, it is proposed the implementation of fault-tolerance techniques in an FPGA-based LDPC decoder.*

1. Introduction

Low-Density Parity-Check (LDPC) codes represent a powerful class of Forward Error Correction (FEC) codes. They were conceived by [Gallager 1962] in his doctoral dissertation, but at the time, they were impractical to implement. Thirty-four years later, [Mackay and Neal 1996] verified that the performance of LDPC codes are equivalent as Turbo codes and could approach Shannon's bound. Since their rediscovery, LDPC codes have been extensively used in several standards like WiFi, WiMAX, DVB-S2, CCSDS and ITU G.hn [Hailes et al. 2015]. These codes indeed offer an excellent performance and when hardware resources capable of implementing them showed up, they became a success in many communication systems.

In the meantime between the creation and the beginning of effective usage, LDPC codes were not heavily investigated. One notable exception is the work of Tanner, in which he generalized LDPC codes and proposed a graphical representation for them [Tanner 1981]. LDPC codes are defined by a sparse matrix (containing mostly zero elements), called parity-check matrix. Each row represents a parity check equation and each column of the matrix represents a codeword bit. The so-called Tanner graphs introduce the concepts of Check Nodes (CN), each of them is attached with a row of the parity-check matrix, and the Virtual Nodes (VN), related with the columns of the parity-check matrix.

Besides the implementation of LDPC codes uses low-complexity calculations, a high level of parallelism can be exploited, which makes them very suitable to be implemented in FPGAs (Field Programmable Gate Arrays). These devices are extremely versatile and can offer high degree of parallel processing. In addition, they offer rapid prototyping and are specially useful for measuring Bit Error Rate (BER) performance, due to the reduced time that the simulations take (when compared to a general-purpose processor, for instance) [Hailes et al. 2015]. Their usage have been increasing more and more, and they are not restricted to coding applications, FPGAs are present in other fields such as industrial, automobile and medical applications.

When these applications run over the presence of ionizing radiation, FPGAs (and any other semiconductor device) are susceptible to many effects that vary in magnitude from data disruptions to permanent damage ranging from parametric shifts to complete device failure [Baumann 2005]. The immediate effects caused by radiation are called single-event effects (SEEs). The particular type of SEE that we are interested are the single-event upsets (SEUs). SEUs are soft errors, because the device itself is not permanently damaged by radiation, but the radiation event causes enough charge disturbance to reverse or flip the data of a memory cell, register, latch or flip-flop. SRAM-based FPGAs are specially susceptible to SEUs within the configuration memory [Wirthlin 2015].

In order to ensure data reliability on an FPGA-based LDPC decoder, this work aims to mitigate SEUs within the design, taking advantage of the architecture of the circuit. To the best of our knowledge, this is the first work proposing fault tolerance mechanisms focusing on FPGA-based LDPC decoders. ASIC implementations have been proposed, however, as in [Li et al. 2016], which presents an area-efficient design by using Hamming decoders inside the control logic and [May et al. 2008] which presents techniques to improve the performance of the decoder in the presence of SEUs. ASIC and FPGAs have a different model of failures, though.

The decoder used as reference is presented by [Júnior 2016]. Fault injections and the characterization of faults inside the check nodes (CN) were performed, since it is the main module of the decoder. It implements a Layered Decoding architecture and the Modified Min-Sum algorithm.

2. Background

The following topics detail the basic concepts of this work. Initially, in section 2.1, it is discussed the effects of radiation in FPGAs, followed by some basic concepts of LDPC codes (section 2.2). Finally, section 2.3 presents previous works related with this paper.

2.1. Radiation Effects on FPGAs

FPGAs are semiconductor devices consisting of logic blocks, RAM blocks and I/O blocks. The most fundamental logic block of an FPGA is formed by a Lookup Table (LUT) and a Flip-Flop (FF). The I/O blocks surround the outer edge of the microchip, providing I/O access to the pins on the exterior of the FPGA package. Programmable routing is implemented so that it is possible to connect logic blocks and IOBs to logic blocks arbitrarily [Buell et al. 2007], as shown in Figure 1.

Xilinx devices quantify the logic resources in terms of “slices”, each of them contains several LUTs and FFs - the nature and quantity of the hardware resources available in each slice depends on the model and generation of the FPGA. The terminology given by Xilinx also introduces the concept of Configurable Logic Blocks (CLBs), which consists of multiple slices [Buell et al. 2007].

To determine the effects of radiation on a FPGA, it is necessary to classify these devices in three categories, based on the technology used to store the configuration data: antifuse, flash and SRAM-based FPGAs. Antifuse FPGAs are nonvolatile and the

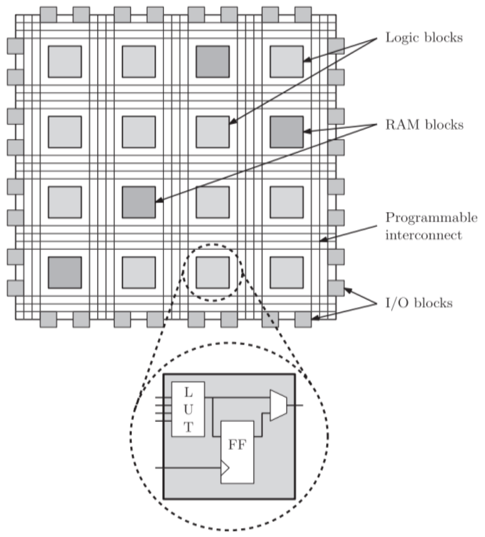
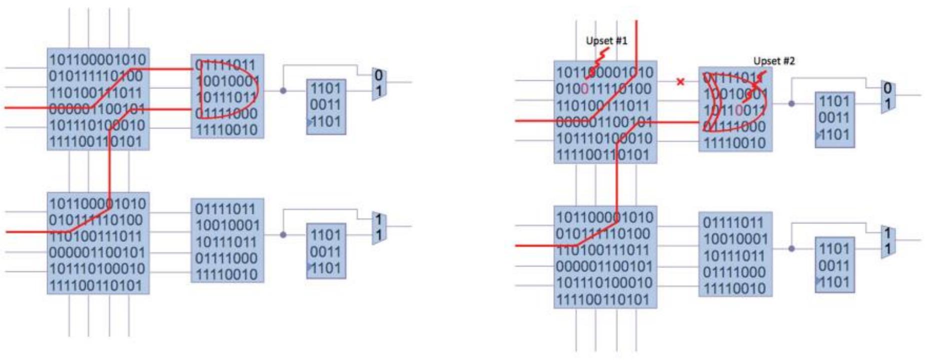


Figure 1: FPGA structure [Hailes et al. 2015]

configuration data cannot be changed once the fuses have been programmed. This type of FPGA is usually the most reliable, since the configuration cells are made from passive, programmed fuses, and they are generally immune to single-event effects. Flash FPGAs are also nonvolatile but they may be reprogrammed only for a certain number of times, which may not be suitable for reconfigurable systems requiring frequent reconfiguration. Flash FPGAs are as well immune to SEUs and both types of FPGAs have a primary radiation concern in SEUs within the user flip-flops and block memories.

SRAM-based FPGAs, on the other hand, are volatile, so they lose their configuration when power is removed. Although SRAM cells require more power than antifuse or flash cells, they can be reprogrammed an unlimited number of times. SRAM FPGAs have a primary reliability concern in SEUs within the configuration memory, since these cells are made using standard static memory techniques and comprise the majority of the memory cells on the device. [Wirthlin 2015].

The incidence of radiation in the configuration memory causes SEUs, which may lead to changes in the logic and routing of the operating circuit, deviating from the function they were supposed to fulfill. Figure 2 illustrates this behavior.



(a) (b)

Figure 2: (a) configuration bits used to a certain logic and routing (b) logic and routing are modified due to upsets caused by radiation [Wirthlin 2015]

2.2. LDPC Codes

An LDPC code is defined by its parity check matrix (H), a matrix containing mostly zero elements and few nonzero elements. Three important parameters are its code word length (n), its dimension (k) and the number of parity bits (m = n – k).

We can categorize these codes by regular and irregular. A code is regular if the number of nonzero elements do not vary amongst each row or each column of the matrix H. Irregular codes, on the other hand, do not respect this property. In general, irregular codes have a better performance than the regular ones [Carrasco 2009]. Figure 3 illustrates examples of both types of codes.

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \qquad \mathbf{H}_2 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

(a) (b)

Figure 3: parity-check matrices (a) from an irregular code (b) from a regular code

LDPC codes can be described as a *k*-dimensional subspace C of the vector space of binary *n*-tuples over the binary field $\mathbf{F}_2$. We first describe a basis B = $\{g_0, g_1, \ldots, g_{k-1}\}$ which spans C. Each **c** ∈ C can be written as **c** = $u_0g_0+u_1g_1+\ldots+u_{k-1}g_{k-1}$, or simply

$$\mathbf{c} = \mathbf{u}G \tag{2.1}$$

where u = $[u_0, u_1, \ldots, u_{k-1}]$ and G is the *k x n* generator matrix whose rows are the vectors $\{g_i\}$. The *(n – k)*-dimensional null space $C^{\perp}$ of G comprises all the vectors **x** for which $\mathbf{x}G^T = 0$ and is spanned by the basis $B^{\perp} = \{h_0, h_1, \ldots h_{n-k-1}\}$. For each **c** ∈ C, $\mathbf{ch}^T_i = 0$, or simply

$$\mathbf{cH}^T = 0 \tag{2.2}$$

where H is the *(n-k) x n* parity-check matrix whose rows are the vector $\{h_i\}$ and is the generator matrix for the null space $C^{\perp}$ [Ryan 2003].

The process of encoding a message is given by 2.1. We can obtain G from H by some simple steps. It is necessary first to transform H with Gauss-Jordan elimination in order to get H in the form:

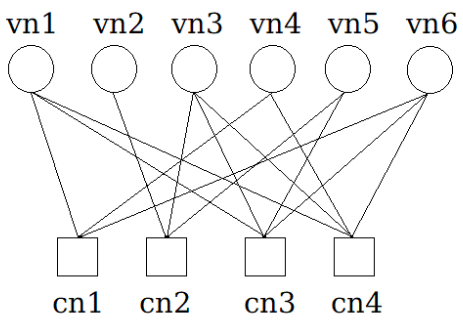
$$H = [A, I_{n-k}] \tag{2.3}$$

where I is the identity matrix of dimension n-k and A is a matrix of size *(n - k) x k*. From that, G can be easily found:

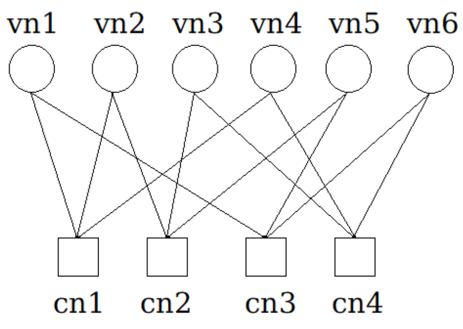
$$G = [I_k, A^T] \tag{2.4}$$

2.2.1. Tanner Graph Representation

Each row of the parity-check matrix (H) represents a parity-check equation and is represented by a Check Node (CN) in the Tanner graph. In the same way, each column represents a coded bit and is represented by a Virtual Node (VN). Edges on Tanner graphs may only connect two nodes of different types and an edge between a check node *j* and variable node *i* exists whenever element h_{ji} in H is equal to 1. Figure 4 shows the Tanner graphs corresponding to the matrices of Figure 3.



(a)



(b)

Figure 4: Tanner graphs corresponding to matrix (a) H1 (b) H2

2.3. Related Works

The following topics present works related with this paper. The first two propose techniques to mitigate soft errors in LDPC decoders implemented in ASICs. The third presents the implementation of an FPGA-based LDPC decoder and a study about the effect of faults in this scenario.

2.3.1 Area-efficient LDPC decoder

[Li et al. 2016] proposes the design of a fault-tolerant LDPC decoder that corrects soft-errors caused by SEUs inside the control logic with a Hamming decoder. For RAM cells, a layered pipelined architecture is presented as well as a scheme that detects soft errors by parity check, then the errors will be corrected by the inherent decoder's iterative process. This approach could save 42% of cell area compared with TMR method and the reduction of 42% and 12% of memory bits compared with similar works.

2.3.2 Resilient LDPC decoder

[May et al. 2008] presents a technique that assumes that not all data bits of a message or channel value have the same importance and corruption in higher significant bits has a larger impact on the overall communications performance than corruption in lower bits. If an LLR value which is calculated by a functional node is corrupted, the value is reset to 0. In other words, the corresponding node/edge is temporarily removed for the current iteration. Thereby, no information is associated with the respective bit and the error tends to be minimized.

2.3.3 FPGA-based LDPC decoder and characterization of faults

[Júnior 2016] implemented a parameterizable LDPC decoder using FPGA, as well as a bit accurate simulator written in C. In order to evaluate the effects of faults on an FPGA-based LDPC decoder, it is also presented the results of fault injections campaigns performed inside the check nodes, the most important module of the circuit. Modified Min-Sum algorithm is implemented, as well as a Layered Decoding architecture, as shown in Figure 5.

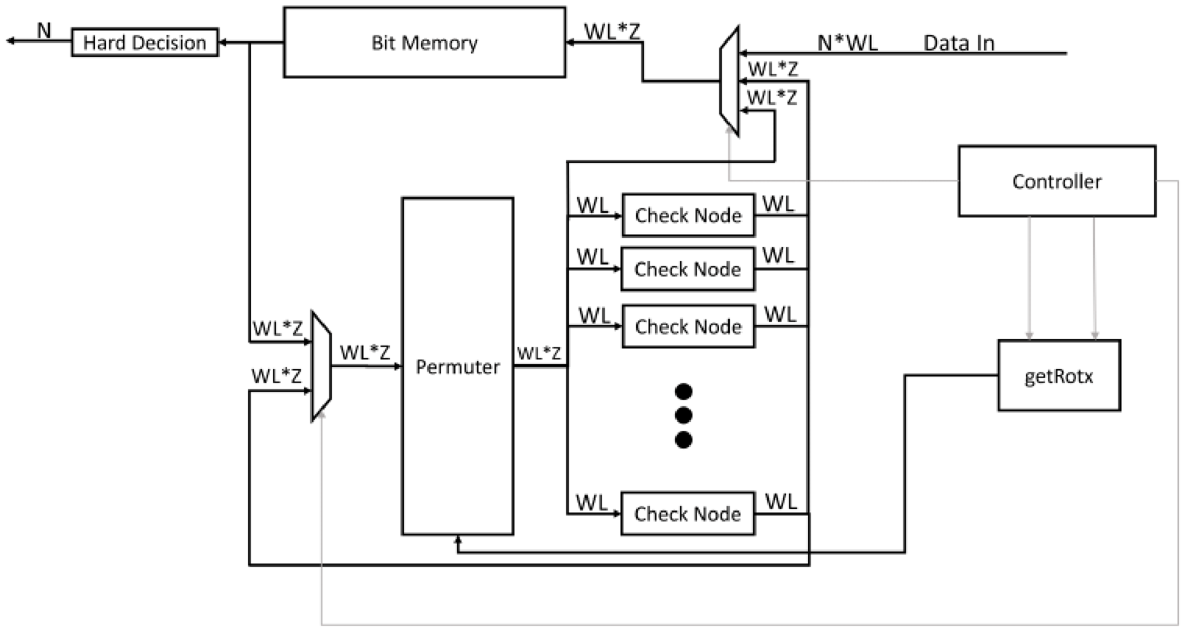


Figure 5: architecture implemented by [Júnior 2016]

Figure 6 illustrates the importance of the mitigation of soft errors in FPGA-based LDPC decoders. The signal is highly degraded in the presence of faults.

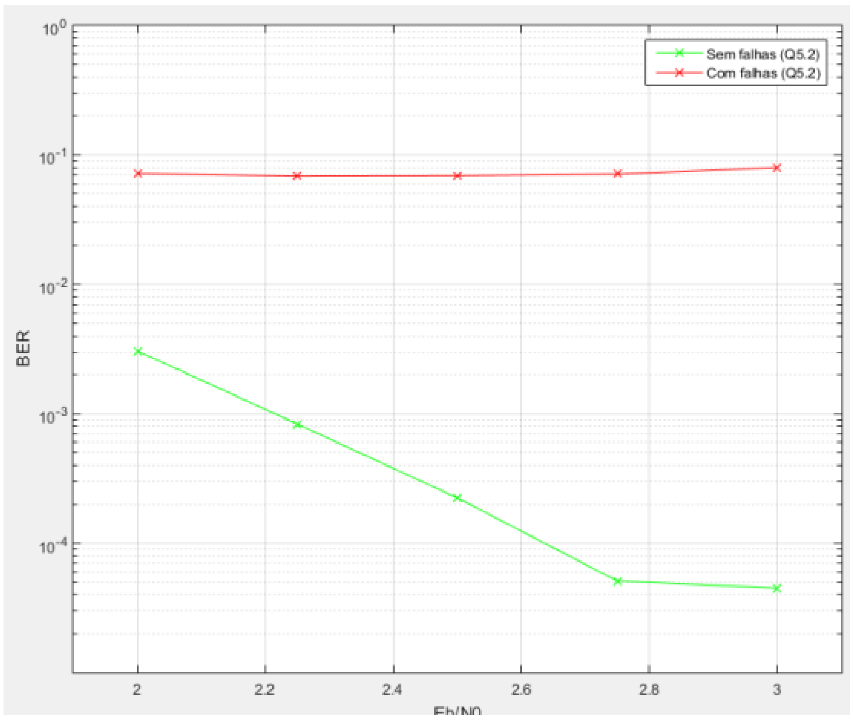


Figure 6: comparison between BER with and without faults [Júnior 2016]

3. Ongoing work

3.1. Fault Injection Platform Adaptation

The fault injection campaigns performed in [Júnior 2016] took place in a fault injection platform specific for data communication systems [Leipnitz et al. 2016]. A Xilinx Virtex-5 XUPV5-LX110T FPGA board was used (for which the platform was originally developed). Due to a bigger amount of logic resources, this work will use a Kintex-7 XC7K325T-1FFG676 FPGA board, also from Xilinx. Since the FPGAs have a different architecture, an adaptation of the platform from one board to another has already been made.

The platform is composed of a module responsible for the communication with the PC Host (PCIe I/O Control), another one to perform readback and partial reconfiguration in the configuration memory of the FPGA (System Control) and the circuit that will be submitted to faults (CUT I/O Control). The first two modules were the ones that suffered the most significant changes due to the difference in the PCIe bus width and the scheme to access the bits in the configuration memory, that differs from one board to another. Figure 7 shows the architecture of the platform.

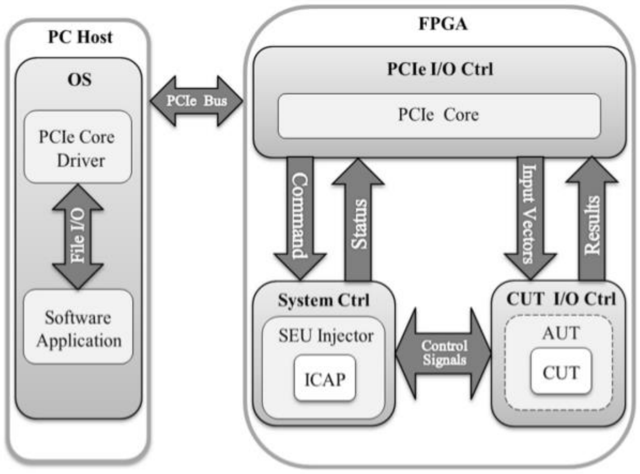


Figure 7: fault injection platform architecture [Leipnitz 2016]

3.2. Selective Technique to mitigate SEUs

The next step of this work encompasses a detailed study in the architecture implemented in [Júnior 2016] in order to find the best approach to mitigate SEUs. TMR (Triple Modular Redundancy) is the most classical and robust technique with this purpose, where a module is replicated three times and the output is extracted from a majority voter. This technique, however, demands excessive area overhead and, in SRAM FPGAs, the voter circuit has to be implemented using SRAM cells which themselves are highly susceptible to upsets [Samudrala et al. 2004].

Several publications have proposed alternatives to the traditional TMR method by protecting specifics subsets of the circuit, the ones considered most critical. [Foster et al. 2010] presents several methodologies for selecting these subsets, such as metrics that consider the number of logic cells that use the cell's output signals as inputs, the number of logic resources necessary to add TMR to the logic cell and the number of logic cells in longest propagation path through the logic cell.

Another approach is presented by [Samudrala et al. 2004], where a Selective Triple Modular Redundancy (STMR) technique is described. The logic cells are classified by the "sensitivity" to SEUs, which is measured by the signal probability of its inputs. It is assumed that the primary inputs of the circuit are specified by the user in terms of signal probabilities and then it is propagated to compute the signal probability of each internal node.

In a different way, [Pratt et al. 2006] prioritizes the protection of structures causing "persistent" errors within the design. Configuration bits are categorized in "persistent" and "non-persistent". A non-persistent configuration bit will cause a design fault when upset and may be repaired through configuration scrubbing, which will lead the design back to normal operation. Persistent bits, on the other hand, will also cause a design fault when upset, but after repairing persistent bits through configuration scrubbing, the FPGA circuit does not return to normal operation.

An analysis of the "severity" of CNs will be performed in this work. In other words, faults will be injected sequentially in each CN in order to get the BERs caused by individual CNs under a faulty state, in order to determine the most critical CNs in the circuit and propose a selective mechanism to protect the circuit from SEUs.

4. Schedule

The development of the second stage of this work also comprises the implementation of the technique in the C-based simulator and in the VHDL modules. The steps are listed below and Table 1 shows the estimated amount of time necessary for each task.

1. Analysis of the "severity" of CNs and definition of the selective technique to mitigate SEUs
2. Implementation of the technique in the LDPC decoder simulator
3. Validation and performing of fault injection campaigns in the simulator in order to evaluate the technique implemented
4. Implementation of the technique in the VHDLs modules
5. Validation and performing of fault injection campaigns in the FPGA board
6. Writing and presentation of Graduation Work II

Task	Feb	Mar	Apr	May	Jun	Jul
1	X					
2	X	X				
3		X	X			
4			X	X		
5				X	X	
6					X	X

Table 1: activities schedule

5. Final Considerations

This work presented a study about LDPC codes and the effects of radiation on FPGAs. It discussed about the architecture of an FPGA and explained why it is susceptible to soft-errors, particularly SEUs. It has also given a formal definition to LDPC codes and the graphical representation for them, the Tanner graphs.

As the concern with fault-tolerance mechanisms in FPGA-based LDPC decoders is not heavily exploited in literature, we presented related publications targeting ASIC implementations. It was also shown the LDPC decoder implementation used as reference in this work, the fault injection platform and the modifications required to exchange the FPGA board used.

Finally, it was explained the approach that will be taken to define the best technique to mitigate SEUs in the circuit and the tasks necessary to conclude Graduation Work II.

6. References

GALLAGER, R. G. Low-density parity-check codes. Information Theory, **IRE Transactions on communications**, IEEE, v. 8, n. 1, p. 21–28, 1962.

MACKAY, D. J.; NEAL, R. M. **Near Shannon limit performance of low density parity check codes.** In: . [S.l.]: [Stevenage, etc., Institution of Electrical Engineers], 1996. v. 32, n. 18, p. 1645–1646.

TANNER, R. M. **A recursive approach to low complexity codes.** In: . [S.l.]: IEEE, 1981. v. 27, n. 5, p. 533–547.

BUELL, Duncan; EL-GHAZAWI, Tarek; GAJ, Kris; KINDRATENKO, Volodymyr. **High-performance reconfigurable computing.** Computer, IEEE Computer Society, vol. 40, no. 3, pp.23–27, March 2007. [Online] Available: http://www.computer.org/csdl/mags/co/2007/03/r3023.pdf.

HAILES, P. et al. **A survey of FPGA-Based LDPC Decoders**. IEEE Communications Surveys & Tutorials, IEEE, v. 18, n. 2, p. 1098–1122, 2015.

WIRTHLIN, Michael. **High-reliability FPGA-based systems: space, high-energy physics, and beyond.** IEEE Proceedings, vol. 103, no. 3, pp. 379–389, March 2015.

CARRASCO, Rolando; JOHNSTON, Martin. **Non-Binary Error Control Coding for Wireless Communication and Data Storage.** Pages 201-235. 2009. ISBN: 978-0-470-51819-9

RYAN, William. **An Introduction to LDPC Codes**. CRC Handbook for Coding and Signal Processing for Recording Systems, B. Vasic, ed., CRC Press.

JÚNIOR, Geferson L. H. **Implementação e caracterização de falhas em um decodificador**

LDPC. December 2016.

PRATT, Brian; CAFFREY, Michael; GRAHAM, Paul; MORGAN, Keith; WIRTHLIN, Michael. **Improving FPGA Design Robustness with Partial TMR**. IEEE International Reliability Physics Symposium Proceedings, San Jose, CA, 2006, pp. 226-232. DOI: 10.1109/RELPHY.2006.251221

FOSTER, David L.; HANNA, Darrin M. 2010. **Maximizing area-constrained partial fault tolerance in reconfigurable logic**. Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays. ACM, New York, NY, USA, 259-262. DOI: http://dx.doi.org/10.1145/1723112.1723155

SAMUDRALA, Praveen K.; RAMOS, Jeremy; KATKOORI, Srinivas. **Selective triple Modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs**. IEEE Transactions on Nuclear Science, vol. 51, no. 5, pp. 2957-2969, Oct. 2004. DOI: 10.1109/TNS.2004.834955