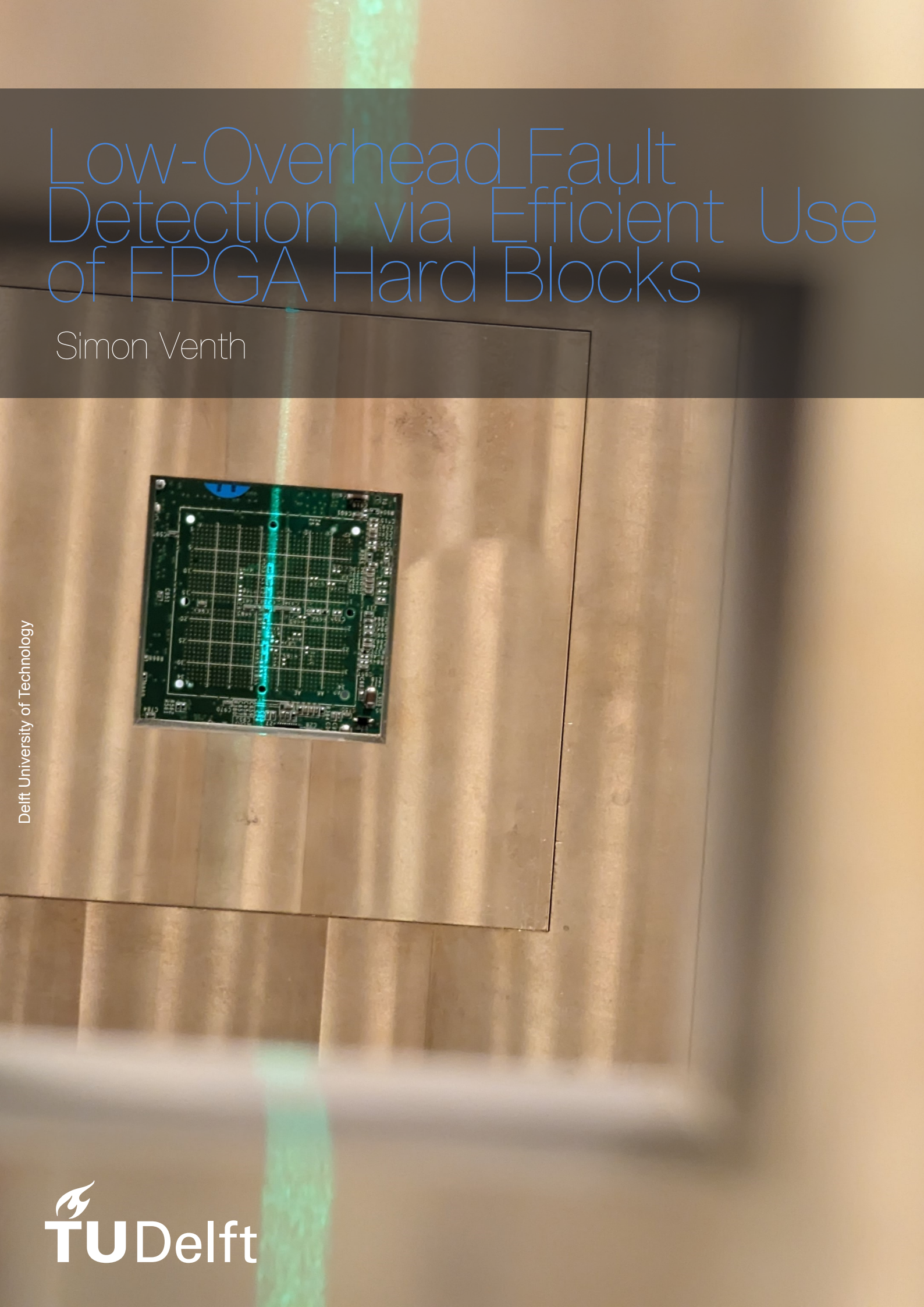
Low-Overhead Fault Detection via Efficient Use of FPGA Hard Blocks

Simon Venth



Delft University of Technology

TU Delft

Low-Overhead Fault Detection via Efficient Use of FPGA Hard Blocks

by

Simon Venth

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday 23rd of November, 2023 at 12:00.

Student number: 4614577
Project duration: December, 2022 – November, 2023
Committee:

S. Speretta	Chair, TU Delft
A. Menicucci,	Thesis Supervisor, TU Delft
A. J. van Genderen,	Examiner, TU Delft

Cover: Path of the proton beam through brass collimators, targeting the FPGA.

TU Delft

Preface

I would like to express my gratitude to everybody that supported me in this work. I embarked on this project knowing very little about integrated circuits, fueled only by curiosity, and I ended up discovering a discipline I never knew I would love. The world as we know it would not exist without semiconductors and digital circuits, and yet we take their existence and ever-improving performance over the past decades for granted.

First and foremost I want to thank my family; Lilli, Ines, Bernward, Georg, Helga & Margot. Thank you for always believing in me and enabling me to obtain this degree.

Thank you Alessandra, for your support and guidance on- and off-topic, and for introducing me to the wonderful world of embedded systems. Thank you to the people at Argotec and at TU Delft that were open to discussion and gave crucial hints during challenging moments.

Of course, thank you to all my friends for being in my life. In Aachen, in Delft, somewhere in the Netherlands or wherever you are, I am grateful for your support and I hope I see you soon. Thank you also to the community at Makerspace Delft, that continues to inspire me and will always be an anchor in Delft.

Simon Venth
Leiden, November 2023

Summary

Modern space systems operate payloads of ever increasing complexity while the size of the average spacecraft is decreasing. Communications payloads specifically can generate significant amounts of data that require a sophisticated data handling subsystem for processing. A key enabling technology is the Field Programmable Gate Array, or FPGA. An FPGA is an integrated circuit that can be configured in software to implement logic in hardware. As such, it can offer the throughput advantage of custom circuitry while circumventing some of the drawback such as high lead times and high unit cost for small production runs. As for any electronic component in the space environment, radiation is a threat to dependable operation. Interactions between charged particles and the semiconductor material deteriorate the device and can cause momentary effects called single event effects. These effects can falsify data or in the case of SRAM based FPGAs, also modify the functionality of the integrated circuit. Entirely avoiding the introduction of such faults is infeasible, as such the system must be able to operate in the presence of faults, it must be fault tolerant to be dependable.

The EuFRATE project aims to demonstrate a novel data handling system architecture based on a cluster of FPGAs. A guiding principle is to create a regular structure on several levels. Each FPGA node hosts soft processors, communication interfaces and a regular structure of hardware accelerators, called programmable functional unit or PFU. This array of accelerators performs digital signal decoding on a low density parity check (LDPC) code. A 1024-bit block length code with code rate ½ is processed using the Min-Sum algorithm. The design exhibits varying levels of fault tolerance, while the higher level entities are well protected the PFUs only generate a basic health signal.

The implementation of the LDPC decoder was analyzed and compared to the current state-of-the-art. It was found that the classic Min-Sum algorithm is unstable at higher signal to noise per bit ratios. It was also found that an 8 bit quantization is implemented while modified Min-Sum algorithms have been shown to perform well at lower quantizations. As such, the algorithm was adjusted to a Normalized Min-Sum algorithm using a normalization factor of 1/2. The quantization was successfully reduced from 8 to 5 bit without any measurable loss of decoding performance. This reduction in data width enabled a reallocation of compute resources and the application of information redundancy using a duplication with compare approach. The new health signal originating from the comparison monitors the duplicated data streams directly and thus allows for fast failure detection and localization.

Both the baseline and the proposed design were tested using the proton beam at HollandPTC. A test setup was developed that allowed for workload emulation and data extraction from the device. From this it could be determined when a critical fault caused the decoding task to fail and the health signals used for fault detection could be analyzed. Around 130 irradiation runs were conducted in total, most of which at a proton energy of 120 MeV and fluence of around 5×10^8 protons.cm^{-2}. It was found that the original health signal did not indicate failures reliably. Using the duplication with compare methodology a majority of data corruptions could be detected accurately. Only 12 % of cases were not identified with this approach either. Overall, fault detection and localization has been improved, allowing for quicker and more targeted recovery and thus improved availability of the entire system.

Due to the reduction in data width, the overhead of the fault detection mechanism was limited and far lower than the 100 % typical for a duplication methodology. Flip-flop utilization increased by about 4.7 % relative to the baseline while logic look-up utilization even decreased by 2.4 %. Digital signal processor utilization remained constant. Only the overhead in block RAM was significant at 22.7 % due to limitations of the BRAM primitives. Since the memory is underutilized in this configuration, it is assumed that further optimization could reduce the block RAM overhead considerably.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
ADC	Analog-to-Digital Converter
ALU	Arithmethic Logic Unit
ARTES	Advanced Research in Telecommunications Systems
ASIC	Application Specific Integrated Circuit
AXI	Advanced eXtensible Interface
BP	Belief Propagation
BRM	Base Functional Model
BRAM	Block Random Access Memory
CB	Connection Block
CMOS	Complementary Metal-Oxide Semiconductor
CNN	Convolutional Neural Network
CRAM	Configuration Random Access Memory
DMA	Direct Memory Access
DSP	Digital Signal Processor
DUT	Device Under Test
DWC	Duplication With Compare
ESA	European Space Agency
EuFRATE	European FPGA Radiation-hardened Architecture for Telecommunications
FF	Flip-Flop
FMC	FPGA Mezzanine Card
FSM	Finite State Machine
FPGA	Field Programmable Gate Array
GCR	Galactic Cosmic Rays
GEO	Geosynchronous Orbit
HDL	Hardware Description Language
IC	Integrated Circuit
IP	Intellectual Property
LDPC	Low Density Parity Check
LET	Linear Energy Transfer
LLR	Log Likelihood Ratio
LUT	Lookup Table
MBU	Multi Bit Upset
MCU	Multi Cell Upset
MUX	Multiplexer
NA	Network Adapter
NRE	Non-Recurring Engineering
PFU	Programmable Functional Unit
PIP	Programmable Interconnect Points
RAM	Random Access Memory
SEE	Single Event Effect
SEFI	Single Event Functional Interrupt
SEL	Single Event Latchup
SET	Single Event Transient
SEU	Single Event Upset
SNR	Signal-to-Noise Ratio

Abbreviation	Definition
SRAM	Static Random Access Memory
TID	Total Ionizing Dose
TMR	Triple Modular Redundant
TOWER	Triple Overwatch Kernel
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
VHDL	Very High-Speed Integrated Circuit HDL

Symbols

Symbol	Definition
$\vec{c}$	Codeword vector
d_c	Check node degree
d_v	Variable node degree
E_b	Energy per bit
H	H-matrix
N_0	Noise per bit
α	Normalization factor
β	Offset value

1 Introduction

Modern space systems heavily rely on digital electronics to operate. They are essential to receive, process, and transmit information, both within the spacecraft and with other space- or ground-based infrastructure. Depending on the size of the spacecraft and mission profile, hundreds or thousands of electronic components are carried on board and interconnected to form the data handling subsystem of the craft [35]. Typically, the majority of the components are Integrated Circuits (IC), based on semiconductor materials and colloquially known as chips. The integrated circuits on board can vary significantly in complexity, ranging from basic analog-to-digital converters and operational amplifiers to processors and specialized signal processors.
The continued doubling of performance of digital electronics roughly every two years according to Moore's law has made modern devices incredibly powerful and efficient [63]. Two key observations can be made for the spaceflight domain specifically. First, the increased capability of on-board digital systems allows for more computations to be performed in-situ as opposed to on ground. This opens new opportunities such as large degrees of autonomy for spacecraft, a more selective transmission of payload data to ground or smart routing for satellite-to-satellite communication. Secondly, space-grade electronics are on average seven years behind their terrestrial equivalent in terms of performance as also shown in Figure 1.1 [23]. While it may not sound like much, Moore's law implies the performance of space-grade components is thus at least eight times worse. Considering that the price of space-grade hardware is also significantly higher, and with the emergence of New Space, it is logical that the space industry increasingly employs terrestrial-grade components, also classified as commercial-off-the-shelf (COTS) technology [24].

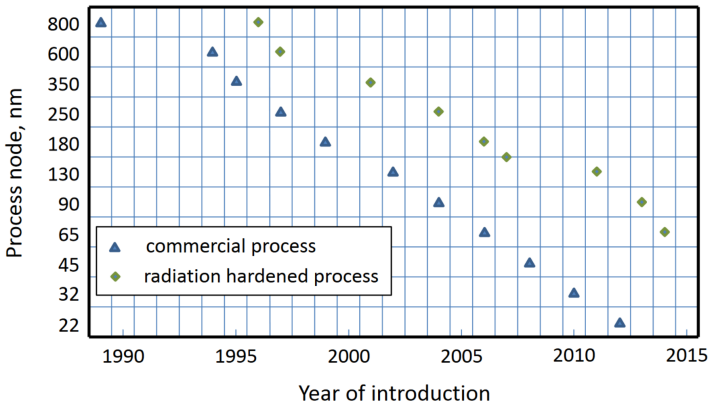


Figure 1.1: Radiation-hardened process nodes are on average seven years behind terrestrial [23].

1.1. FPGAs in Spaceflight

Field Programmable Gate Arrays (FPGA) are a type of integrated circuit that can be configured, in software, for any task at hand. This differentiates them from Application Specific Integrated Circuits (ASIC), where the hardware is designed and manufactured for a specific function. Both technologies come with advantages and disadvantages. While ASICs typically represent the most efficient implementation of a given functionality, they do not exist for all use cases. FPGAs promise to adapt to any functionality, at the price of reduced efficiency. Both types of IC are commonly found aboard spacecraft, however FPGAs have become increasingly popular over recent years for a number of reasons [35].
Using FPGAs allows for a shorter development cycle and can also significantly decrease the Non-recurring Engineering (NRE) cost otherwise spent on hardware development. This is especially attractive for space systems, where the total number of devices required is low compared to terrestrial applications, leading to a much larger unit price. The technology is also suitable for high-throughput data handling, which is increasingly relevant for e.g. high-resolution imaging payloads or communication payloads. A general trend can be observed where the size of satellite decreases while they handle more complex data processing tasks on-board [74]. Another key aspect of FPGAs based on static RAM (SRAM) technology is that they allow for reconfiguration, meaning the software mapped to hardware on the device can be adjusted or updated over-the-air. This is a unique property of FPGAs that creates new opportunities for the designer to improve the circuit design after launch or to respond to changes in the mission environment. The increasing usage of FPGAs compared to competing technologies is also plotted in Figure 1.2.

Reliable operation of an FPGA in space is not a given, considering there are external threats stemming from the intense radiation environment. One answer is naturally the use of space-grade components, which underwent extensive testing and qualification processes to ensure certain behavior in the presence of radiation. For reasons previously mentioned, a space-grade component may not be desirable however. COTS devices developed for the terrestrial market are not protected by design against radiation effects. The interaction of the FPGA's semiconductor material and energetic particles, trapped in Earth's magnetic field or originating from the sun or outside of the solar system, can produced Single Event Effects (SEE) momentarily and also permanently degrade or damage the device [71]. In the event of the former, an ionized particle strikes the chip, subsequently decelerates and releases its kinetic energy into the device. This energy is problematic as it will cause voltage transients in the integrated circuit that when latched into a memory element, will falsify the logic state of the circuit and cause a so-called soft error or upset. Additionally, FPGAs based on SRAM technology are vulnerable to upsets that affect circuit functionality, as their volatile configuration memory can be affected. This is even more detrimental than a problem with the circuit state, as it will cause the circuit to malfunction continuously until the fault is removed [71]. Fortunately, this fault removal is relatively simple for SRAM devices. By exploiting partial reconfiguration, the fault can be removed while the chip continues to operate the unaffected modules [10].

The European FPGA Radiation-hardened Architecture for Telecommunications (EuFRATE) project, in the context of which this research was performed, aims to demonstrate a novel data handling architecture based on COTS devices. A cluster of multiple FPGAs is proposed, where each FPGA is interchangeable and able to operate autonomously. Each FPGA node represents a heterogeneous system comprised of soft processors, networking infrastructure and memory controllers. The actual data processing task is performed by a highly regular accelerator array, that is supplied with data and controlled by the previously named modules. In order to reach sufficient dependability, highly critical components in the design are protected through redundancy. The accelerator array is not one of them, and fault tolerance is only facilitated by periodic scanning for faults and subsequent removal via reconfiguration. A principal design driver in the EuFRATE project is to limit the resource overhead associated with making the system fault-tolerant, by deciding the degree of protection per component.

1.2. Motivation

With the increased adoption of COTS devices in spaceflight applications, new challenges arise for creating systems sufficiently dependable under the influence of radiation. When components are not radiation-hardened by design, it is up to the system architect to develop and implement effective miti-

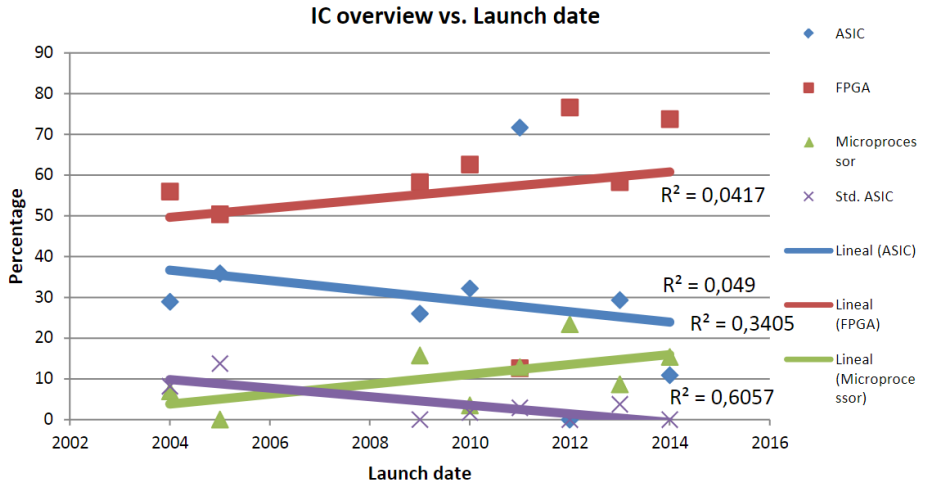


Figure 1.2: Percentage of complex ICs used on European missions over time [35].

gation strategies for radiation effects. As the introduction of faults in SRAM FPGAs cannot be avoided realistically, the configured circuit must be able to cope with these faults while still meeting performance and availability requirements.

Numerous methods to achieve the above have been developed and applied in the past. A standard approach is to apply spatial redundancy by duplicating or triplicating the circuit and comparing the resulting data streams. This allows for a detection or a detection & masking of a fault respectively. While these methods have been shown to effectively improve the fault tolerance of any circuit, the associated increase in resources required is immense [52, 42]. Also termed overhead, duplicating or triplicating the circuit also creates overhead exceeding 100 % or 200 % in compute resources. Applying this to an entire system architecture is often not feasible in inherently resource-constrained space systems. A family of low-overhead methods exists that aim to find a compromise between overhead and the achieved level of fault tolerance. Algorithm-based fault tolerance is one such method, that exploits the conservation of certain properties in matrix multiplication [40]. Another is reduced-precision redundancy, where a complex calculation is checked against one or more estimates of the result that were computed less precisely and thus with less overhead [55]. Naturally, these methods cannot be generalized as easily and are often specific to certain circuits or processes.

In summary, an increased adoption of terrestrial-grade electronics in space applications can be observed. These devices are not radiation-hardened by design and thus prone to disruptions threatening dependability in the intense space radiation environment. Ensuring sufficient dependability is required however, and thus the system designer must provide adequate means to achieve fault tolerance. Due to the high resource overhead associated with general methods, such as duplication with compare or triple modular redundancy, they are not attractive and should be applied sparingly. Instead, low-overhead methods tailored to a specific design can represent a better solution. In this project, solutions applicable to the accelerator array in the EuFRATE architecture were investigated. Since an effective fault removal mechanism is already present in the architecture, the focus was put on reliable and fast fault detection.

1.3. Outline

This thesis report is organized in four main parts. First and foremost, in the following chapter, background information will be provided on FPGA technology, its working principles, as well as the context of this project. The decoding application executed in EuFRATE's accelerator array will be presented in detail, and variations of the underlying algorithm will be discussed. Furthermore, the radiation environment will be characterized, the interaction between energetic particles and semiconductor material will be considered, and their impact on the device discussed.

In chapter 3, the implementation of the application on the FPGA will be presented. This will proceed in a structured way, where first the baseline and its operation are given, followed by necessary modifications in anticipation of the fault detection mechanism. Lastly, said mechanism will be proposed, including the exposed health signal and a watchdog module for test purposes.

Chapter 4 is concerned with the experimental campaign conducted for this research. The radiation facility will be introduced, the developed test setup and the test procedure will be elaborated. A number of auxiliary systems had to be developed to extract data from the unit under test. These systems will be presented as part of the test setup chapter as well.

Chapter 5 gives the results of this project regarding the application's performance, its dependability attributes and the overhead of the proposed fault tolerance mechanisms. These results will be contextualized and linked to other works when possible. Finally, conclusions are drawn in chapter 6 and a number of recommendations are made in chapter 7.

2

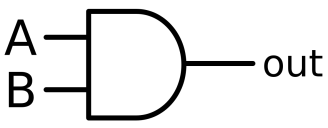
Background

In this chapter, relevant background information will be presented to the reader to ensure a thorough understanding of the most critical concepts for this project. First the FPGA technology will be discussed briefly, after which the EuFRATE project will be presented. The EUFRATE project provided the real-life context of this research and the baseline which the author tried to improve upon. Another key concept to be presented are Low Density Parity Check (LDPC) codes, which are the workload foreseen in EuFRATE. Lastly, the radiation environment in space will be discussed. Principal sources of radiation as well as an estimation of the environment anticipated for EuFRATE are given. Radiation effects will also be discussed towards the end of this chapter. Finally, some guiding research questions are formulated and presented.

2.1. FPGA Technology

In essence, an FPGA is made up of a high number of identical configurable resources and wiring arranged in a certain layout. While the resources available, their size and their placement have evolved over the years, certain building blocks are universal to FPGA technology. This section will present an overview of said blocks and their purpose, enabling a thorough understanding of this type of integrated circuit.

The smallest computational unit in an FPGA is a Lookup Table (LUT), which takes a fixed number of binary inputs and gives a corresponding binary output value. In this way it is analogous to a truth table, which can be evaluated for the complete set of possible input values. Figure 2.1 shows a basic AND gate on the left and the corresponding truth table on the right. In fact, any Boolean algebra operation can be captured in a truth table and thus any logic gate can be captured in a LUT [48]. Note that the limiting variable is the number of inputs, for a specified number of inputs any logic gate structure resulting in one output can be represented, as the LUT covers the complete set of input combinations.
In order to implement sequential or state-holding logic, a form of register must be present in the component. Thus a LUT is usually supplemented with a D flip-flop (FF) and an output multiplexer (MUX), such that the current output of the LUT or a retained value stored in the flip-flop can be output on demand. This set of a LUT, a FF and a MUX forms a *logic block* and is an essential low-level component of FPGAs. The logic blocks found in modern devices bear close resemblance with this structure [25].



(a) AND gate

Input A	Input B	Output
0	0	0
0	1	0
1	0	0
1	1	1

(b) Truth table

Figure 2.1: 2-input system, AND gate and truth table behave identically.

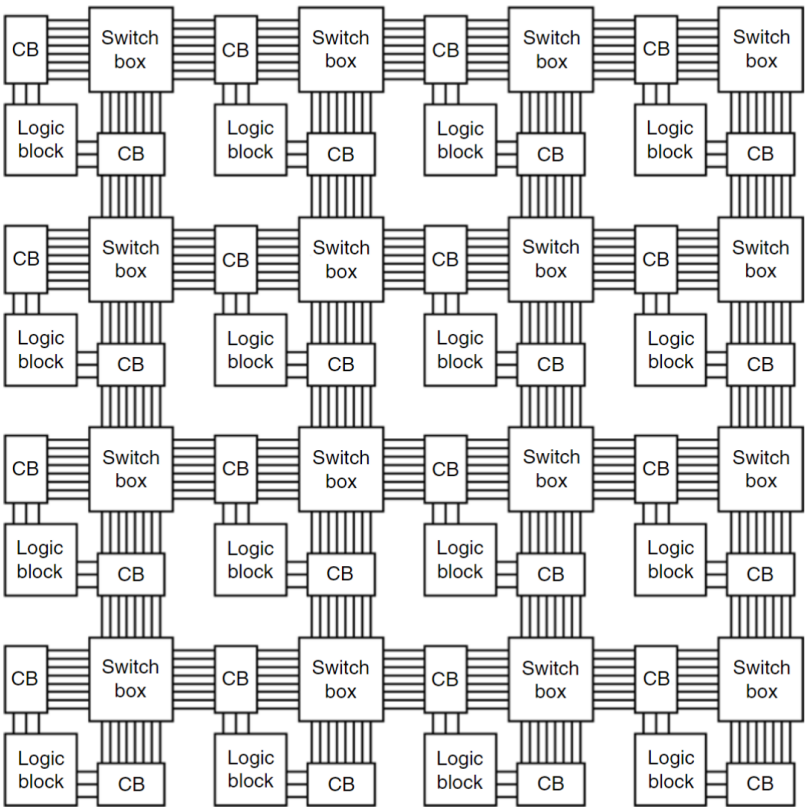


Figure 2.2: An segmented island-style FPGA architecture [25].

Naturally, a single logic block is severely limited in its capabilities. In order to implement more sophisticated computations, several tens or hundreds of blocks are required and thus signals need to be routed between logic blocks. These routing resources are also called *interconnect* and consist of wires and programmable switches that form the required connection [33]. As the number of wires required rapidly scales with the number of logic blocks, the routing resources take up a large area on the FPGA silicon. This leads to the term *island-style* architecture, where the logic blocks would be islands in a sea of interconnect.
With the large number of logic blocks present in modern devices, it is infeasible to interconnect all blocks. Furthermore it is inefficient to have logic blocks perform routing tasks and there is an interest in keeping connections localized for the sake of latency. Architectures found in modern devices are more segmented or hierarchical to respond to these challenges [25]. Figure 2.2 illustrates a segmented architecture where the logic blocks interface with *connection blocks* (CB) that are placed on horizontal and vertical tracks. At the track junctions, *switch boxes* can connect and disconnect the tracks using so-called *programmable interconnect points* (PIP). This approach is sometimes called *mesh-based* [33] and can thus exhibit both short and long wires by segmenting the tracks, without consuming logic block resources for routing tasks. This architecture can be further augmented by e.g. adding direct connections between adjacent logic blocks to relieve the interconnect or adding wires between non-adjacent switch blocks to reduce delay [25]. This is also illustrated in Figure 2.3. The design tools must consider those circuit-level optimizations when placing a design on the device in order to exploit e.g. *fast carry chains* that significantly speed up addition operations [25].

It is evident that the routing between logic blocks is a complex issue that has sprouted many different developments in the past. The routing network of an FPGA can take up to 80-90% of silicon area with only 10-20% dedicated to logic elements [17]. Similarly, up to 80% of the configuration memory are reserved for routing resources. Understanding faults in routing remains challenging as vendors only disclose ”limited information about how routing resources are mapped into the bitstreams” [22].

Although basic logic elements are extremely flexible and can in principle implement any computa-

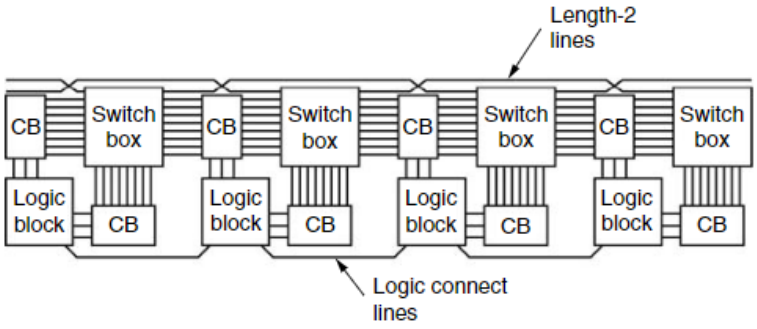


Figure 2.3: An augmented interconnect fabric [25].

tion, this does not imply that every computation can be *efficiently* implemented. Taking the example of multiplication, the designer can choose between an array multiplier or shift-accumulate method, however the former introduces significant delay and the latter is very area-consumptive when implemented with basic logic blocks [25]. Since multiplication is required in a large share of computations, it makes sense to include dedicated multiplier blocks that interface with the general FPGA fabric since they improve performance significantly. As their design is cast in silicon and cannot be modified significantly, they are also called *hard blocks* [46] or *dedicated logic blocks* and have been introduced in FPGA designs in the early 2000s [67]. Modern FPGAs contain a variety of these hard blocks to accelerate the potential range of applications. Wirthlin [71] mentions internal block memory, programmable processors, high-speed multigigabit transceivers, digital signal processing blocks, analog-to-digital converters, PCIexpress interfaces, clock-management resources etc., making the complete FPGA package much more heterogeneous than evident from its basic logic and routing building blocks. Nevertheless, ”for applications that used them, they reduced the overhead of programmability in area, performance, power and design effort” [67].

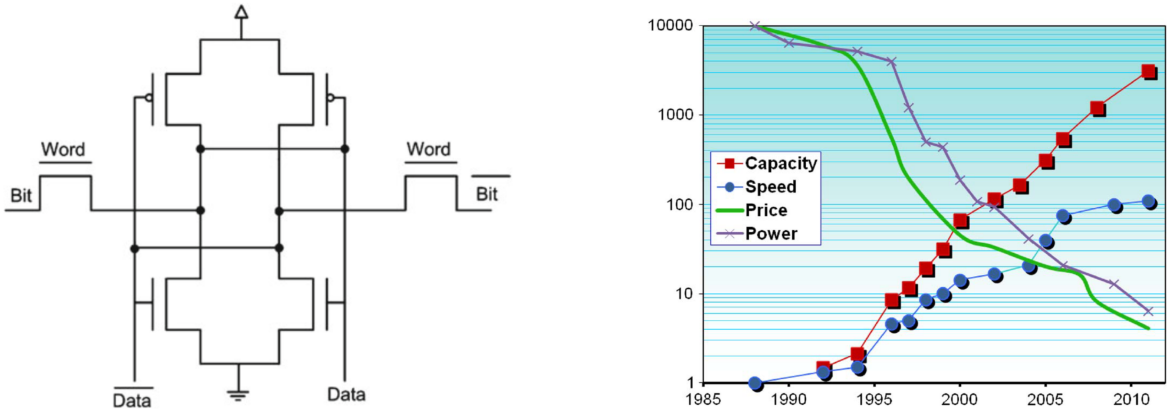
Different technologies exist to realize the above functionality in semiconductors, namely antifuse, flash and SRAM. As of this writing, the most prominent technology for configuration memory is SRAM. It is attractive because it offers fast reconfiguration of the device for an infinite number of times. Disadvantages are that the a SRAM cell is relatively large in silicon, volatile and dissipates significant static power [25]. One reason for this is the relative complexity of a static memory cell. As can be seen in Figure 2.4a, retaining one bit of information requires a total of six transistors. A strong cause for the predominance of this technology in FPGAs is that it is based on standard CMOS process technology, which has been applied to manufacture the majority of semiconductor devices over the last decades [13]. As illustrated in Figure 2.4b, SRAM FPGAs have undergone an exponential rise in capacity (logic element count) and significant improvements in speed over the past decades, while simultaneously decreasing the associated cost and power [67].

In this project, an AMD Kintex UltraScale series FPGA is used, model XCKU040. The device offers high performance for an affordable cost and a high ratio of hard blocks to regular fabric [3]. The same device is used for other testbeds in the context of EuFRATE, while the target node for a flight version would be the model XCKU060 from the same series [21]. Apart from having more of the same compute resources available on chip, there is no significant difference between the two models.

2.2. EuFRATE Project

This thesis research project was undertaken in the periphery of the European FPGA Radiation-hardened Architecture for TElecommunications (EuFRATE) project. The project was launched in 2022 with italian aerospace engineering company Argotec S.r.l Italy being the prime contractor. The development is supported by the the European Space Agency’s (ESA) Advanced Research in Telecommunications Systems (ARTES) program [1]. The goal can be described as demonstrating a novel architecture for

[1] EuFRATE was selected in the invitation to tender AO/1-10240/20/UK/ND: *Hardening techniques for commercial-off-the-shelf FPGA for digital telecommunications payload*[31]



(a) A static memory cell using six transistors [33].

(b) Evolution of Xilinx SRAM FPGA attributes [67].

Figure 2.4: Static Random Access Memory (SRAM) technology.

on-board data handling using multiple COTS FPGAs in a cluster configuration. While the prospect of reduced cost and increased performance through the usage of COTS components is attractive, careful investigation is required since telecommunications payloads typically require the highest levels of reliability [15]. The project responds well to current trends in space embedded systems, where data handling subsystems rely more heavily on programmable components, more COTS devices are being (partially) qualified for space applications, and designers want more flexibility and scalability in their fault-tolerance measures [35, 64].

The overall design philosophy of EuFRATE is to create a system comprised of several components that are functionally equivalent. While those units are not in an information-redundant configuration, they can fulfill each others functionality in principle. An overhead in available components would then allow for the activation of a spare while a defective component is restored through reconfiguration. This design philosophy is followed on several levels. To keep the following presentation of the baseline design structured, it will be given moving from large scale to small scale.

As mentioned previously, EuFRATE will leverage multiple FPGAs in a cluster configuration. The largest design unit is called a tile, which comprises several nodes that work in conjunction. In the preliminary design one tile consisting of four nodes is foreseen, however this can be adjusted depending on performance requirements and resource consumption of a node. Tiles and nodes are relevant with respect to fault tolerance too, tiles monitor the status of other tiles and nodes monitor other nodes. Figure 2.5 illustrates how the nodes within a tile can be connected. In this case, each node represents a different FPGA, which will be assumed to be the case for the rest of this report. To transmit data between the devices they are connected using the high performance Aurora protocol [4] on a serial link connecting two nodes at a time. Note that each node has three interfaces connecting to other nodes in the tile and one available to connect to other tiles. Within a tile, we thus have a full mesh topology. The router component either forwards packages to the next interface or the network adapter, which exposes the data to its host node [21].

Each node hosts a number of systems related to networking, computing and monitoring. While the node architecture itself is heterogeneous, the nodes are interchangeable and can act as both master and slave depending on the situation. The spacecraft may also act as master for the cluster. Below the primary components of a node are listed:

- Processing System
- Beacon Controller
- Accelerator Array
- Router & Network Adapter
- Memory Controller

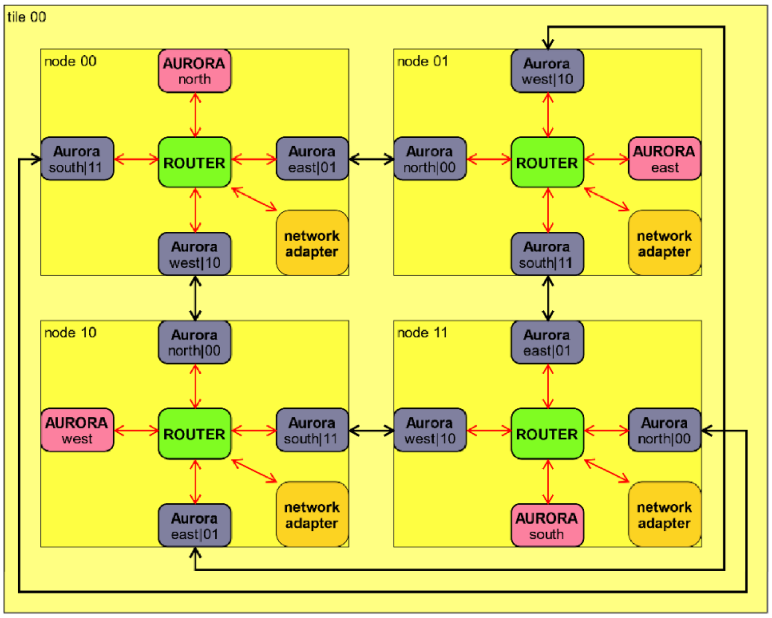


Figure 2.5: Communication infrastructure between different nodes in a tile [21].

- Aurora Interfaces

The processing system represents a double domain approach, with a hardened microprocessor running the main reliability and cluster management services, and a basic microprocessor to support application-specific operations and data flow management. The former is called TOWER (Triple Over-Watch kERnel) and consists of three MicroBlaze soft processors in triple modular redundant (TMR) configuration. Due to its criticality for the operation of the whole system, it is one of the few components that are fully TMR protected and thus highly unlikely to fail due to soft errors. This component also handles the regular control link with other nodes/tiles or the spacecraft.
The other microprocessor is called GaRDeN (General and Real-time Domain) and does not benefit of the same fault-tolerance measures. It is a single MicroBlaze core in real-time preset that deals with less critical tasks related to the application. Logically, the TOWER and GaRDeN units form the BaByloN (Blazes and Blaze On-Node) processing system together.

The beacon controller is again comprised of two distinct units, the local and the global controllers. The local controller monitors the health signals of the node, while the global controller monitors the signals of other nodes. Note that both controllers are also TMR protected due to their importance in detecting failures and initiating recovery. The nature of the health signals will be elaborated at a later stage.

The accelerator array is the component of the node that performs the actual data processing, in this case LDPC decoding. As the name suggests, it is an array of identical units which are also called Programmable Functional Unit (PFU). The design and functioning of a PFU will be presented in detail in section 3.2. According to the EuFRATE philosophy, this accelerator array is the only part that is highly application specific. While the PFUs are used for LDPC decoding in this context, a different accelerator array design could be created for a different task in a telecommunications context or even other applications such as image processing.

Proceeding with the router and network adapter (NA), these building blocks are naturally required to handle the data exchange with other nodes, tiles and the rest of the spacecraft data handling subsystem. It comprises the typical receive and transmit infrastructure and has direct access to the node's memory controller for on-chip memory. Creating dedicated modules for networking relieves the soft processors of these operations that are well defined but critical for latency and reliability of data exchange. Router

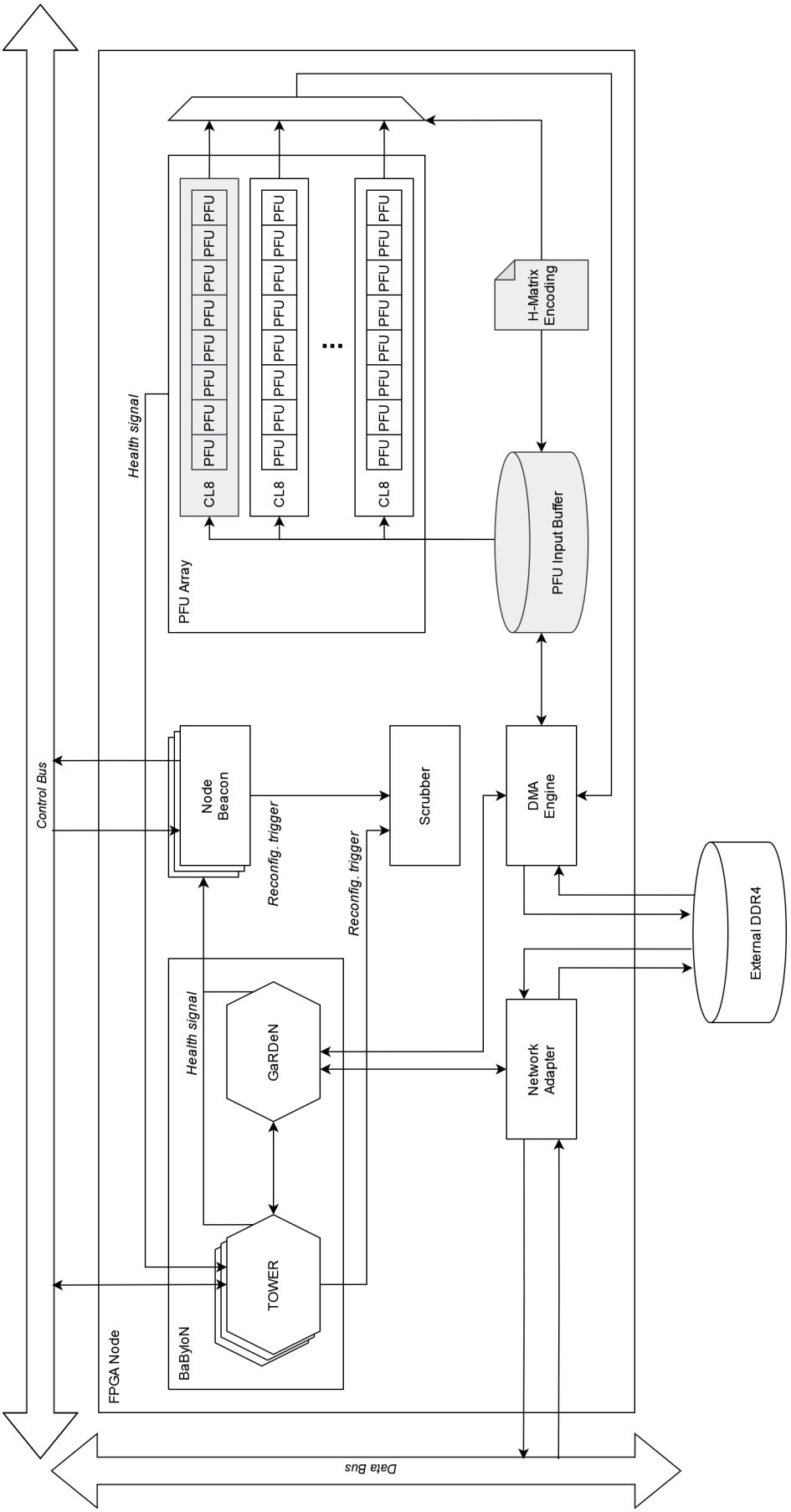


Figure 2.6: EuFRATE envisioned node architecture. Grey colored systems were available for this project.

and NA are directly controlled by the Babylon processing system and not further protected against radiation effects [21].

The memory controller enables the node to access external DDR4 memory and offloads this tasks from the processing system. While it would be possible to configure part of the FPGA fabric as memory, this would be an inefficient use of resources and an external chip is much more suited for the task.

Finally, the Aurora serial interfaces have been described earlier when showcasing the tile architecture. They are essential for allowing high throughput data transmission between different nodes and tiles. As opposed to the router and network adapter which operate in the application layer, this module embodies the firmware of the interface, ensuring correct timing and other properties of the interface.

Due to this research being conducted concurrently with the development of the above subsystems, the author only had access to the accelerator array implementation. This included however a basic state machine that allows for LDPC decoding on reduced block length in a base functional model configuration. Other components of the EuFRATE architecture were not tested in the scope of this project.

2.3. LDPC Codes

Low Density Parity Check (LDPC) codes play a pivotal role in modern telecommunications, especially in error-prone communication channels like wireless and optical networks as used in space systems. Their impressive error correction performance and adaptability to various channel conditions make them ideal for applications such as 5G, near-earth communication, and deep-space communication. LDCP codes are rivaled by Turbo codes, which also apply an iterative decoding scheme but cannot parralellize iterations and generally perform worse at high code rates [61]. In recent years, LDPC codes have become the prominent choice over Turbo codes and are a fundamental part of communication standards such as DVB-S2 for satellite broadcasting and 5G NR for mobile applications [2, 1].

2.3.1. Working Principles

LDPC codes are linear block codes characterized by a sparse parity-check matrix, giving rise to their ”low-density” descriptor. A sparse matrix is attractive with respect to memory requirements, reducing complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ [61]. The parity-check matrix is denoted H and forms the basis of the code. An example is shown in Equation 2.1, for a (10, 5) regular code. As for other parity-check algorithms, the rows represent the checks and will produce the null vector when a valid codeword $\vec{c}$ is multiplied according to Equation 2.2 [60]. The construction of the] parity-check matrix H can be done with different methods that will not be presented in detail here. The original method proposed by Gallager divides the H matrix into d_v submatrices H_d, which are column-permutations of the first submatrix H_1. This is a straightforward way of guaranteeing the variable and check node degrees to be regular, but more efficient schemes have been proposed since then [60]. Regular codes have a constant number of ’1’ entries per row and column, also called variable and check node degrees d_v and d_c respectively. Regular codes simplify matrix construction and decoding [60].

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{2.1}$$

$$\vec{c}H^T = \vec{0} \tag{2.2}$$

The H matrix represents the rules to be satisfied during decoding, however it gives no insight about the decoding process itself. In order to eliminate errors in the received codeword, statistical methods are necessary to calculate the probabilities that a certain bit of the codeword is correct or incorrect. Fundamentally, this is based on the knowledge of how many parity checks, that the message bit contributes to, are correct and whether the parity checks themselves are computed using likely correct or

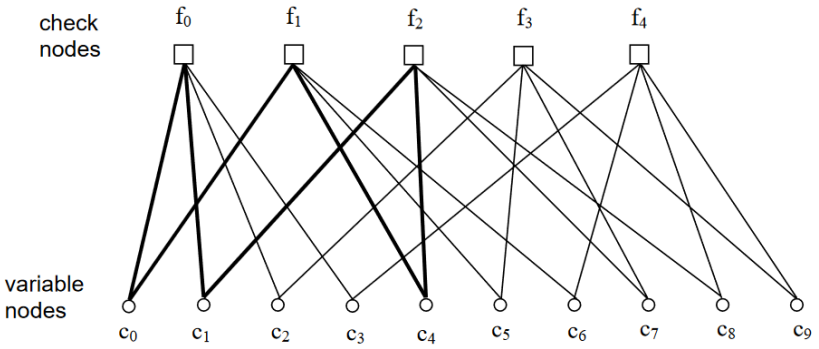


Figure 2.7: Tanner graph for a regular code with $d_c = 4$ and $d_v = 2$ [60].

likely incorrect bits. This iterative information exchange process is called Belief Propagation (BP) and will be presented in the following.

At its core, LDPC decoding using BP involves iteratively refining the estimated codeword by exploiting the relationships between variable nodes (representing transmitted bits) and check nodes (representing parity constraints). The algorithm updates these nodes' beliefs based on information gathered from other nodes. The '1's in the parity-check matrix indicated which nodes are exchanging information. The process gradually mitigates errors in the received codeword in a cyclic process. In graph theory, it can be represented as a message passing algorithm on the Tanner graph, named after its inventor and shown in Figure 2.7 for this example [66]. The Tanner graph is a bipartite graph, meaning that no two check nodes are connected and no two variable nodes are connected, connections only exist across the two sets. Marked bold in the figure is a cycle in the graph, variable nodes c_0, c_1 and c_4 are connected to the same set of check nodes. Cycles can degrade decoding performance as they form so called trapping sets [60]. The existence of cycles also highlights the need for an iterative decoding scheme, as there is no means to solve cycles analytically due to mutual dependence.

Before moving on to different node processing methods, the concept of log-likelihood-ratio (LLR) should be introduced briefly. As mentioned previously, decoding schemes heavily rely on probabilities. As an example, take the likelihood that a transmitted bit X is '1' given the observation Y: $P(X = 1|Y)$, which will be some value between zero and one. As the code is binary, there will be a complementary likelihood $P(X = 0|Y)$ and the two likelihoods are required to sum to one. This has to be ensured explicitly through a normalization step. Furthermore, the probabilities are more prone to numerical instability issues and require a larger number of bits in quantization to capture all dynamics of the decoder [61]. To combat the above, the log-likelihood-ratio given in Equation 2.3 can be applied instead of probabilities. By dividing the probabilities, any normalization factor becomes redundant. Furthermore the logarithmic representation projects the values to a more stable range and any multiplication of probabilities becomes addition. Note also that the initial conversion of a received bit is straightforward and done by simply multiplying the received value with a constant, which in turn depends on channel noise properties [61].

$$LLR(X) = \ln\left(\frac{P(X=0)}{P(X=1)}\right) \tag{2.3}$$

From the above equation, three regions of the LLR can be distinguished. When the probability of '0' is higher than '1', the fraction will be larger than 1 and the LLR will be positive. If the probabilities are equal the LLR is zero. If the probability for '1' is larger the LLR will be negative. Thus the sign of the ratio indicates the decision for a binary value, while the magnitude indicates how reliable said decision is. This is considered soft-decision decoding, as opposed to hard-decision decoding where the messages could only take two values. In practice, soft-decision decoding of LDPC codes outperforms hard-decision decoding by a large margin and soft-decision decoding in the logarithmic domain represents the standard approach [43].

2.3.2. Decoding Algorithms

The following algorithmic descriptions have been adapted from Chen and Fossorier [27]. Each iteration of the decoding algorithm consists of a set of messages from check nodes to variable nodes and a set of messages from variable nodes to check nodes. Let i denote the iteration and L and Z the messages originating from check (index m) and variable (index n) nodes respectively. Furthermore $\mathcal{N}(m)$ and $\mathcal{M}(n)$ denote the set of neighboring nodes of nodes m and n. For regular LDPC codes, the number of neighboring check and variable nodes is constant and called the degree d_c and d_v. In each iteration, the outgoing messages are calculated as a function of the incoming messages. The check-to-variable message is given in equation 2.4, which is evaluated for all m and all $n \in \mathcal{N}(m)$.

$$L_{mn}^{(i)} = 2\tanh^{-1}\left(\prod_{n' \in \mathcal{N}(m)\setminus n} \tanh\left(\frac{Z_{mn'}^{(i-1)}}{2}\right)\right) \tag{2.4}$$

The variable-to-check message is given in equation 2.5, evaluated for all n and all $m \in \mathcal{M}(n)$.

$$Z_{mn}^{(i)} = L_n^{(0)} + \sum_{m' \in \mathcal{M}(n)\setminus m} L_{m'n}^{(i)} \tag{2.5}$$

This formulation is also called the Sum-Product algorithm, owing to the fact that check nodes must compute a product of the incoming messages and variable nodes a sum to determine the outgoing messages. It was first proposed by Kschischang in 2001, and can be applied for more than just LDPC decoding [45].
Note that the starting condition is that the check variables are simply the received value y_n as shown in Equation 2.6. After the node processing, the best estimate of the codeword can be determined by summing in each variable node the initially received value and the incoming messages from check nodes according to Equation 2.7. This is also called the a-posteriori probability [61]. After this a hard decision on the binary value of each message bit can be made from the LLR as described earlier.

$$L_{mn}^{(0)} = y_n \tag{2.6}$$

$$L_n = L_{mn}^{(0)} + \sum_{m' \in \mathcal{M}(n)} L_{mn}^{(i)} \tag{2.7}$$

The codeword estimate is checked with the H matrix for validity. Depending on the decoder design, the decoding may stop after a valid codeword is found or after a fixed number of iterations is performed. There are no guarantees on convergence, however the algorithm is channel-agnostic and applicable as long as initial LLRs can be determined [61].

Since the check node processing using tanh and $\tanh^{-1}$ is rather computationally expensive, a widely adopted modification is the Min-Sum algorithm, that calculates check nodes according to Equation 2.8 [66, 77]. In this approximation, the check node message is said to be the product of the signs of the incoming messages multiplied with the minimum absolute value of the incoming messages. The algorithm's product of hyperbolic tangents is thus replaced by a minimum operation, which is also apparent in its naming. The derivation showing why this is a valid modification of the Sum-Product algorithm will not be presented here but is available in literature [27].

$$L_{mn}^{(i)} \approx \prod_{n' \in \mathcal{N}(m)\setminus n} \operatorname{sgn}\left(Z_{mn'}^{(i-1)}\right) \cdot \min_{n' \in \mathcal{N}(m)\setminus n} \left|Z_{mn'}^{(i-1)}\right| \tag{2.8}$$

While significantly easier to compute, the Min-Sum algorithm also comes with a small performance loss. This is due to the fact that the check-to-variable messages L are larger than those in the Sum-Product algorithm as proven in [27]. To remedy this, two modifications can be applied: normalization and offset. Normalization is simply performed by dividing by an additional factor $\alpha > 1$ as shown in Equation 2.9. Obtaining the best possible normalization factor can be done analytically by a method called density evolution or by simulation, it is directly dependent on the channel's signal to noise ratio and the optimal value also varies by decoding iteration. To achieve the best possible decoding performance, a set of values should thus be determined. However it has been shown that the performance

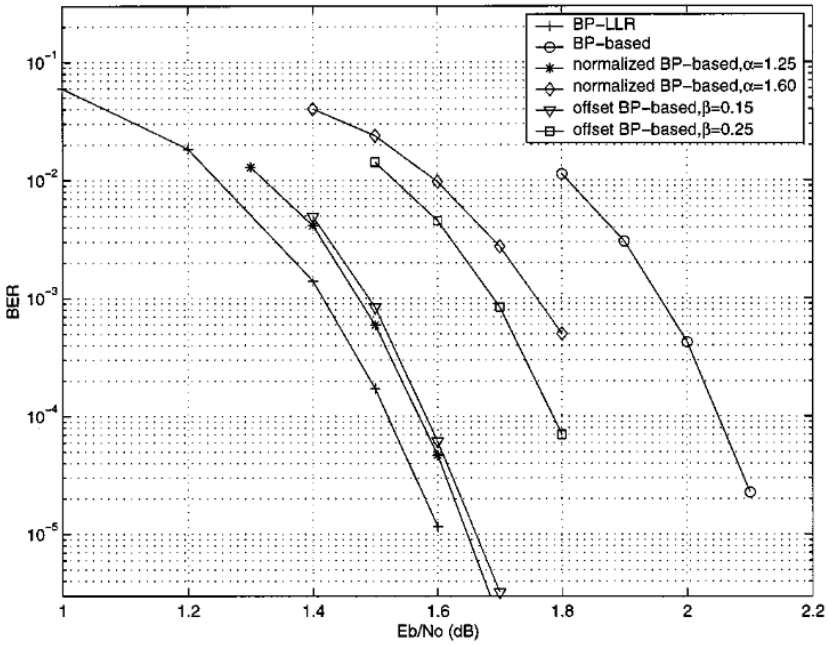


Figure 2.8: Exemplary decoding performance of the Min-Product (BP-LLR), Min-Sum (BP-based) and modified Min-Sum (others) algorithms [27].

loss of a fixed normalization factor is relatively small [27].
Another approach to limit the overestimation of the Min-Sum algorithm is to use an offset value $\beta > 0$. This value is then subtracted from the check-to-variable messages. In order to make this correction stable, the absolute value has to be considered before subtracting as shown in Equation 2.10. Note also that the sign is preserved by giving the subtraction a lower bound of zero before multiplying back with the original sign of the message. As is the case with the normalization, the ideal offset value varies by iteration but can be kept constant for simplicity and without significant losses in performance [77].

$$L_{mn}^{(i)} \leftarrow \frac{1}{\alpha} \cdot L_{mn}^{(i)} \tag{2.9}$$

$$L_{mn}^{(i)} \leftarrow \mathrm{sgn}\left(L_{mn}^{(i)}\right) \cdot \max\left(\left|L_{mn}^{(i)}\right| - \beta, 0\right). \tag{2.10}$$

Both methods of improving the algorithm can compensate for the performance loss between the Sum-Product and Min-Sum algorithms. This is also illustrated in Figure 2.8, where the left most curve represents the Min-Product algorithm and the right most curve the Min-Sum algorithm. The curves plotted for the modifications clearly fall between the two extreme cases and thus offer a compromise between performance and complexity. Note also that the values of α and β need to be carefully chosen, as even small variations can impact the performance significantly [75].

2.3.3. Hardware Mapping

Since the decoder will ultimately be run on an FPGA, the implications of the algorithm and its modification for the hardware implementation also need to be considered. Specialized circuitry of significant footprint would be required to implement a non-standard function such as tanh [18]. In comparison, both a product of signs and a minimum operation are relatively simple operations. In two's complement logic, the first bit denotes the sign, and thus a XOR operation between the first bits of all involved numbers is sufficient to find the sign product. Similarly the absolute value can be found straightforward for positive numbers, while requiring some bitwise operations for negative values. The minimum number in a set will have the largest number of leading zeros, not considering the sign bit. Thus both operations can be performed with relatively small combinatorial circuits.

Generally speaking, binary addition or subtraction is cheaper to implement in hardware than multiplication or division and the length of the output is bounded much closer the the length of the input. As such, the offset modification is preferred from a hardware perspective. The one exception to this is a multiplication or division of multiples of two. To divide a binary number by two, all bits are shifted one position to the right and the least significant bit ”drops off”. Similarly, fractions like 3/4 can be computed by dividing by two and by four separately, and the adding the two results back together. Regarding cost of implementation, it can outperform even the subtraction proposed in the Offset Min-Sum algorithm.

Two important concepts that are also exclusively relevant in a numerical context are clipping and quantization. Both relate to the methodology of mapping the mathematical formulation of the algorithm to compute resources, and are directly impacting decoding performance.

Any number in digital systems is expressed on a discrete interval rather than a continuous one. While mathematically there are infinitely many numbers between e.g. zero and one, in a digital system it very much depends on how we express the number. Taking the very simple case of a 1-bit quantization, only 0 and 1 could be expressed. Increasing the quantization to 2-bit already gives four possible values, namely 0, 1/3, 2/3 and 1. Since this increases with the power of two, a 32-bit quantization could already express the range in increments of less than 2.33×10^{-10}. It is important to realize that any computation will be limited in accuracy based on the underlying quantization. This is why a LDPC decoder using floating point notation (typically 32-bit) is often used for comparison with other, lower bitwidth schemes. From a hardware perspective, the quantization drives the required size of all components interacting with the data such as memory elements, ALUs and communication buses. Thus, as long as decoding performance requirements are met, it is is desirable to have the quantization as small as possible to decrease the footprint of the circuitry. In recent years, the performance of various algorithms, when quantized to low or very low bitwidth, has been analyzed more closely [76, 47]. New FPGA architectures that are more flexible with respect to low quantization computations have also been proposed [20].

Clipping refers to limiting the range of values. As discussed previously, an initial log-likelihood-ratio is required to start the decoding process. The spacecraft receiver does receive an analog signal however, into which the message is encoded typically using binary phase shift keying [2]. In the broader picture, the system thus also includes some type of analog-to-digial (ADC) conversion and initial LLR computation. A conversion takes place between the continuous analog domain and the discrete digital domain. As outlined in the previous paragraph, the quantization dictates how many discrete values can be expressed numerically. However this is not equivalent to the resolution, the resolution is a result of the quantization and the range to be expressed in conjunction. Circling back to the ADC and initial LLR computation, values high in magnitude correspond to high certainty, while values around zero indicate low certainty. From the perspective of the decoder, it may be more attractive to express the small uncertain values in better resolution than to capture the certain far extremes. This is where clipping becomes relevant. Research has shown that clipping can significantly improve the threshold of Min-Sum algorithms at low quantization, which will enter the waterfall region at a lower E_b/N_0 [77] when clipped appropriately. Clipping can also lead to a lower error floor, however in combination with a normalization or offset correction this problem can be alleviated. Zhao et al [77] were able to achieve near optimal performance with their 4-bit quantized, clipped Offset-Min-Sum algorithm. Note that their proposed offset correction is conditional and thus slightly more complex than the regular offset modification.

In this project, clipping was performed implicitly as part of the floating point to fixed point conversion. Fixed point formats are typically expressed in the Qm.n notation, which indicates how many integer bits (m) and how many fractional bits (n) are used to express the number [69]. The sign bit is not included in this specification but assumed to be there, otherwise the format is usually expressed as UQm.n for unsigned numbers. From the Q format, the range and resolution of the fixed point number can be derived in a straightforward manner and is given below.

Range:	$[-(2^{m-1}), 2^{m-1} - 2^{-n}]$
Resolution:	2^{-n}

It is evident that clipping for a value x occurs if $|x| > 2^{m-1}$ because the number falls outside the range of values that can be expressed in fixed point notation. The values outside the range are typically saturated to the minimum or maximum value. Note that once the numbers are clipped and in fixed point notation, the decoder is agnostic to their original numerical value. Whether 01111_2 corresponds to 15_{10} (Q4.0) or 0.875_{10} (Q1.3) will not influence the decoding process as it is conducted entirely in binary representation.

2.4. Space Radiation

This section will introduce the radiation environment that an FPGA typically faces when operating aboard a spacecraft. To do so, different types of radiation will first be described qualitatively, after which the operational environment anticipated for EuFRATE will be characterized. Lastly the ways in which radiation can introduce faults in the FPGA will be elaborated.

Wirthlin [71] defines radiation as the ”the transmission of energy through atomic and subatomic particles with very high kinetic energy”. When these particles interact with semiconductor materials they can deposit energy or cause displacement damage. In addition to that, the ionization tracks left by transiting particles can cause transient effects in the device. This will be discussed further in section 2.4.4, the discussion will be limited to flux per particle for now as this represents a driving parameter for radiation effects [65]. Since the severity of a particle strike is highly dependent on its energy, the other parameter of interest is the energy of the particle, typically measured in eV.

2.4.1. Radiation Sources

Generally speaking, three major sources of radiation affecting electronics can be distinguished. While other sources exist, such as the *Bremsstrahlung*, caused by the deceleration of charged particles in materials [64], those are considered secondary and will not be discussed further here. The three principle sources of radiation to be considered in an Earth orbit are:

- Trapped radiation
- Galactic cosmic rays
- Solar energetic particles

Trapped radiation describes the flux of energetic electrons, protons and low energy heavy ions that are trapped in Earth’s magnetic field, forming radiation belts. The particles move along as well as spiral around the magnetic field lines and bounce between conjugate pairs of maximum magnetic field strength along their trajectories [65]. This motion is illustrated in Figure 2.9 where the drift due to the particle’s charge is also shown. The belts extend from around 100 km to 65000 km and consist of electrons up to several MeV and protons up to several hundred MeV [32]. Note also that the fluxes vary over time with solar activity, as most trapped particles are of solar origin.

Galactic Cosmic Rays (GCR) are a collective term for particles stemming from outside the solar system. These consist of approximately 85% protons, 14% alpha particles and 1% heavier nuclei [65]. GCRs can have very high energies of up to 10 GeV and the flux can be considered isotropic and continuous. Although the flux is relatively low compared to other sources of radiation, the presence of energetic heavy ions is problematic for spacecraft electronics [32]. Specific locations in low Earth orbits are shielded against this form of radiation by the magnetic field, however the effect does not exist at higher orbital altitudes [65].

Solar energetic particles are observed in bursts associated with solar activity that can last from a few hours to a few days [32]. Their origin on the sun can be two-fold: activity on the solar disks such as flares and shock waves (caused by coronal mass ejections) travelling across the heliosphere [32]. The radiation released during the former tends to be electron-rich, while the latter is typically proton-rich [19]. Figure 2.10 illustrates the daily fluence of high-energy protons over a 30 year time frame, as measured by the IMP-8 and GOES spacecraft [19]. Note the influence of the 11-year solar cycle; there are significantly more solar energetic particles during periods of maximum solar activity. Solar energetic particles can display a wide range of energies, from tens of keV to several GeV [32].

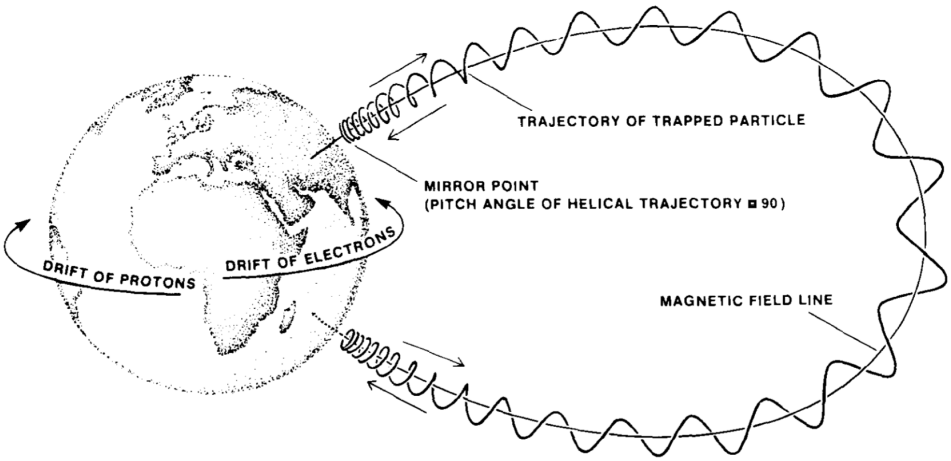


Figure 2.9: The motion of trapped particles in Earth's magnetic field [65].

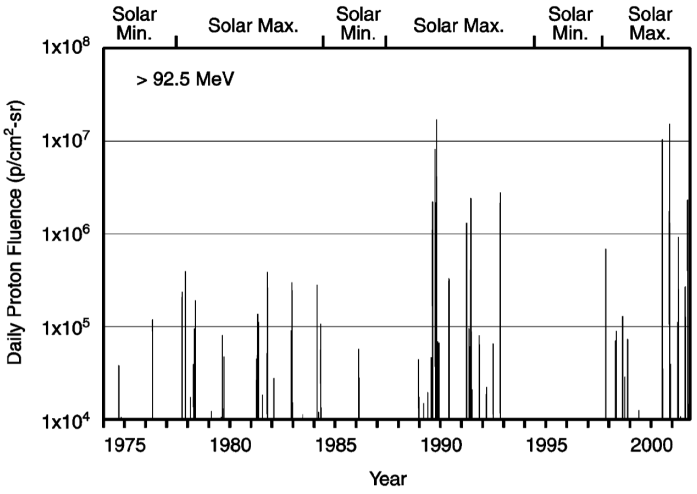


Figure 2.10: Daily fluences of protons with energy > 92.5 MeV due to solar particle events [19].

2.4.2. Interaction of Radiation with Semiconductors

The combined fluxes stemming from the above mentioned radiation sources lead to countless particles interactions for the spacecraft and its component every day. While this may cause degradation to any material, semiconductor materials are especially affected and subject to much more complex failure modes. In this section, the physical interaction between the energetic particle and the semiconductor material will be discussed. Furthermore, the influence of particle energy and type will be discussed among other parameters of interest.
At the atomic level, a charge is introduced to the semiconductor material via two primary mechanisms. These mechanisms are direct ionization by the energetic particle and ionization by secondary particles created by nuclear reactions. Direct ionization is the most important mechanism for heavy ions, while secondary particles are more relevant to electron and proton interactions [62].

When an energetic particle travels through the device, it loses energy while simultaneously creating electron-hole pairs in it. This causes direct ionization and a deceleration of the particle. The range parameter describes how deep the particle will penetrate before coming to rest and needs to be carefully considered for laboratory heavy ion testing to capture all possible failure modes [62]. The energy deposited in a material per unit length is its stopping power. Normalized by the density of the material, it is also know as the Linear Energy Transfer (LET) parameter given in Equation 2.12, where ρ is the density and dE/dx is the rate of energy loss in the material. LET is usually expressed in the unit MeV cm²/mg. Thus, determining the total deposited energy requires integration over the range of penetration or the thickness of the silicon if the particle is very highly energetic.

$$LET = \frac{1}{\rho}\frac{dE}{dx} \tag{2.12}$$

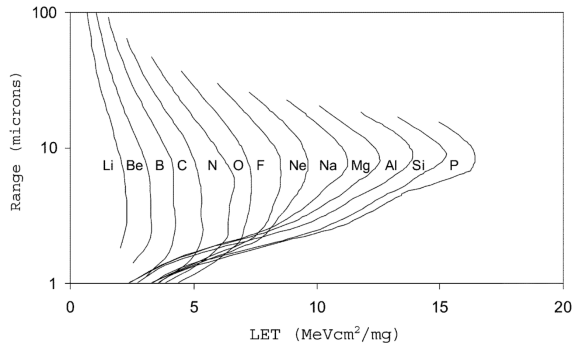
Note that the deposited energy is not equivalent to the charge, which can be determined according to Equation 2.13. In this equation Q is the charge and E_p is the electron-hole pair ionization energy, a material property of semiconductors.

$$Q = \frac{1.6 \times 10^{-2} \cdot LET \cdot \rho}{E_p} \tag{2.13}$$

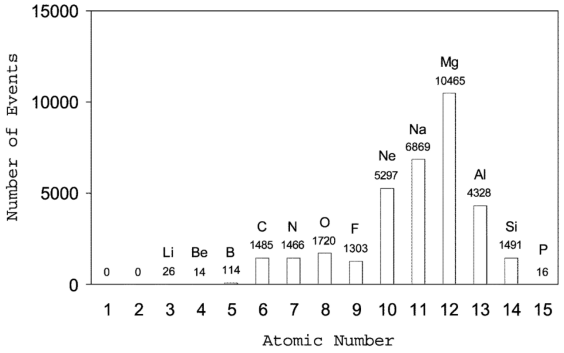
In summary, the LET characteristics of a particle describe its energy loss behavior in a material. The type of nuclei, energy and range of penetration are all important variables in this process. Generally speaking, analysis of a component is done with respect to an LET spectrum representative of the space environment [65]. A threshold LET value can also be determined, below which the device is not affected by the charge introduced by direct ionization alone. Before discussing the mechanism by which the semiconductor is disrupted by ionization, ionization by secondary particles will be elaborated briefly.

Secondary particles are the result of a collision between the energetic particle and a silicon (Si) nucleus of the semiconductor. This can cause one or several nuclear reactions of three primary types: an elastic collision than produces Si recoils; emission of alpha or gamma particles plus the recoil of a lighter daughter nucleus or spallation reactions where the Si nucleus breaks into lighter ions. These so called secondary particles will travel through the material and cause ionization just like external particles that ionize directly [30]. The mechanism of ionization via secondary particles is predominant for lighter particles such as protons and neutrons, implying that they may cause upsets even at LET values lower than the threshold value [62]. This is because the secondary particles are typically much heavier and will exhibit much higher LET values too. Figure 2.11a shows the LET plotted versus range for different nuclei. For the rare chance that a phosphorus nuclei is produced in a nuclear reaction, the LET can be as high as $16\,\mathrm{MeVcm^2/mg}$ [37]. In Figure 2.11b, the likelihood of certain nuclei being produced by $200\,\mathrm{MeV}$ protons is shown and a clear peak around atomic number 12 can be observed. Lower energy protons are more likely to produce magnesium or aluminium nuclei, while this peak practically disappears for very high energy protons of $500\,\mathrm{MeV}$ energy [37].

Having discussed the physical phenomena leading to undesired charges inside the integrated circuit, the vulnerability of SRAM memory technology to this charge will now be analysed. Remembering that a SRAM memory cell consists of several transistors as shown earlier in Figure 2.4a for example, each



(a) Range versus LET of nuclei produced by 500 MeV protons in silicon [37].



(b) Contribution to LET spectrum as a function of atomic number for 200 MeV protons [37].

Figure 2.11: Simulation results for proton induced secondary particle interactions in silicon.

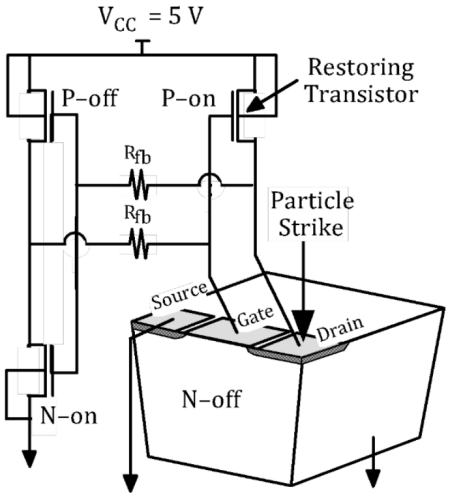


Figure 2.12: Illustration of a particle strike on the n-channel off drain [30].

transistor will exhibit some sensitive locations. A typical vulnerability is the reverse-biased drain junction of a transistor in the off state. Charge collected at the junction can induce an unwanted current. This current will in turn create a current in the on channel transistor to balance the memory cell, which will show as a voltage drop at its drain. This voltage transient can be mistaken as a write pulse and locked into the cell, causing a erroneous circuit state [30]. Given that there are several transistors necessary to form an SRAM memory cell, multiple sensitive locations exist for each cell.

Whether an introduced voltage transient will lead to an upset of the circuit state via direct ionization is highly dependent on the circuit and semiconductor design as well as the particle's LET. As can be seen in Figure 2.13, even particle below the threshold LET can introduce significant voltage transients in the memory cell. The circuits characteristics will determine whether the cell recovers sufficiently fast by absorbing the introduced charge, or whether the inverter opposite of the struck transistor is faster and locks the faulty memory state. Determining the threshold LET is thus complex and requires careful analysis of the circuit's three dimensional structure and response to voltage transients [29].

In summary, faults are introduced to the integrated circuit by an undesired charge in a sensitive location of a transistor. The charge is deposited by an energetic particle via direct ionization or ionization through secondary particles. The direct ionization process is driven by the linear energy transfer characteristics of the particle, which describe the energy introduced to the semiconductor material as function of penetration range. A threshold LET value can be defined, below which the circuit is able to

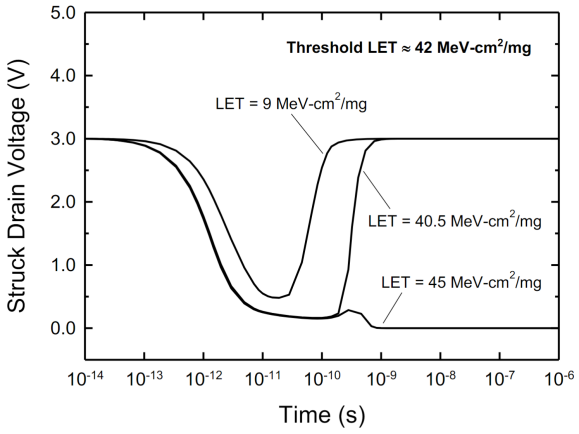


Figure 2.13: Transient voltages in SRAM transistors in response to particle strikes below, just below and above the threshold LET [30].

recover before the unwanted charge is latched into logic. Direct ionization is the primary mechanism for interactions with heavy ions, which typically have high LET values. For lighter particles such as protons or neutrons, ionization via secondary particles is the prominent mechanism. Collisions between the energetic particle and nuclei of the semiconductor material can cause nuclear reactions that create particles heavier than the incient particle. These secondary particles will in turn cause direct ionization as described previously. As such, even particles below the threshold LET can very well cause upsets.

2.4.3. EuFRATE Environment

For any spacecraft, a well defined environment is key to a successful design. In the case of EuFRATE, the proposed architecture serves as a demonstrator for future telecommunications satellite technology. Given that such high performance payloads are typically flown on large satellites, the system is assumed to operate in geosynchronous orbit (GEO), to which it is launched directly. The proposed mission duration is 15 years. Regarding shielding, 10 mm of aluminium equivalent is assumed. This is sufficient to shield from some energetic electrons but the majority of higher energy particles will not be stopped [65]. The shielding parameter is a typical input to SEE rate models such as CREME96 used in this context [68]. The mission profile is also summarized in Table 2.1.

Table 2.1: Expected mission profile of EuFRATE [49].

Mission Duration	15 years
Spacecraft Shielding	10 mm (Al Equivalent)
Spacecraft Location	Geosynchronous Orbit
Launch	Direct in GEO

The EURFRATE consortium performed a preliminary estimation of upset rates using the CREME96 model [49], the results of which are given in Table 2.2 and Figure 2.14. Both solar minimum and solar maximum conditions are considered. It can be seen that the at the worst there would be a SEE rate device per day of about one per day. Note how the rate is lower for for solar minimum conditions. This is due to the fact that the Earth magnetic field is stronger at said condition, and thus provides better shielding against GCR [68]. Most parameters are device properties, with the exception of the essential bits rate. This rate relates to the actual design configured on the device, and considers that not all SEEs will affect bits essential for the functioning of the device. The value given in the table is the result of a preliminary study performed by Argotec and considers approximately 25 % of the bits essential. A more detailed discussion and classification of SEEs will follow in the next section.

2.4.4. Single Event Effects

Figure 2.15 shows a breakdown of radiation effects in FPGAs. Total ionization effects are accumulative phenomena that degrade the semiconductor material and may lead to partial or complete failure of the

Table 2.2: Single event effect rates for different conditions [49].

SEE Rate	Solar Condition	
	Minimum	Maximum
SEE Rate device per day	1.3×10^{0}	8.4×10^{-1}
SEE Rate device per second	1.5×10^{-5}	9.7×10^{-6}
SEE Rate bit per day	1.1×10^{-8}	6.6×10^{-9}
SEE Rate bit per second	1.2×10^{-13}	7.6×10^{-14}
SEE Rate of Essential Bits per Day	3.4×10^{-1}	2.1×10^{-1}

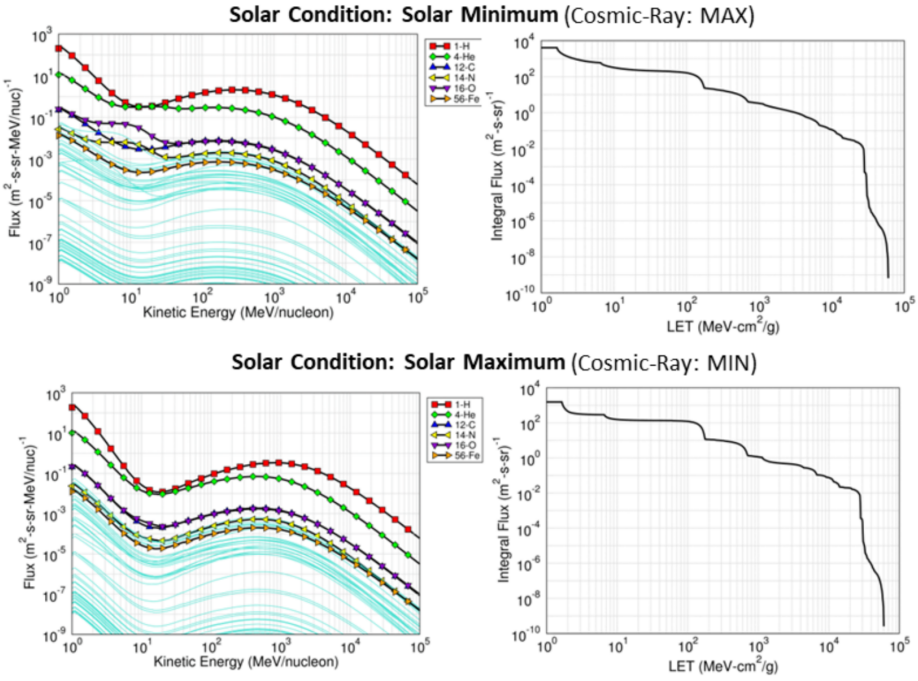


Figure 2.14: Flux against particle kineteic energy and LET for different solar conditions [49].

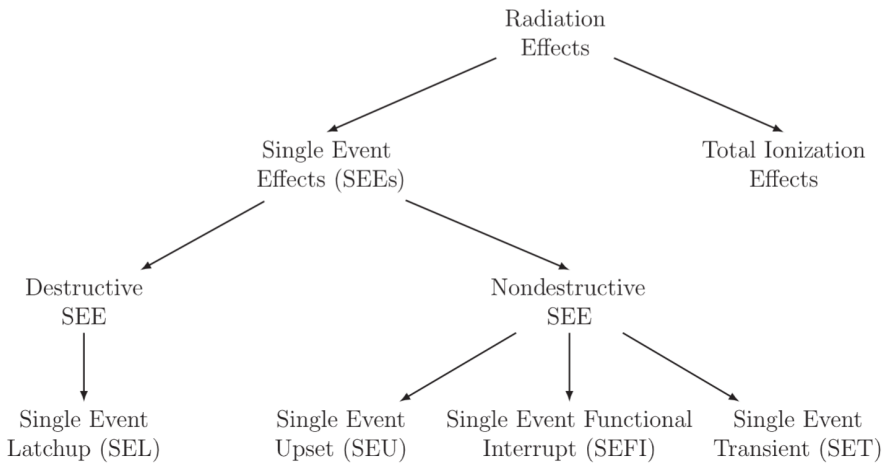


Figure 2.15: Common radiation effects in FPGAs [64].

device. They are considered out of scope here, also because the FPGA under study has been shown to meet typical mission requirements in that respect. The focus is put on Single Event Effects (SSE), which can further be subdivided into destructive and nondestructive events. All events will be presented, although the vast majority is nondestructive in nature [71].

A potentially destructive SEE in SRAM FPGAs is the Single Event Latchup (SEL), where ”a single charged particle induces a parasitic p-n-p-n structure” [71]. This results in large currents flowing through the parasitic transistors that can destroy the device unless power is removed quickly. Space-grade FPGAs exist that implement additional hardware measures to avoid SELs entirely [73]. Not shown in Figure 2.15 are the Single Event Gate Rupture (SEGR) and the Single Event Burnout (SEB), both destructive events that have become relatively rare on modern devices [44]. Another term used to describe an event that causes permanent damage to the device is hard error. For the device used in this project, no SEL has been observed and the chip is thus considered immune to this effect [16].

Moving on to nondestructive effects, also called *soft errors*, the Single Event Upset (SEU) is typically the most common type of event. A particle causes a transfer of charge between two nodes and can thus change a critical node in a memory cell. In that case, the logic value of the memory element can be flipped, a logic ”1” becomes a ”0” or vice versa [71]. This is not damaging to the device but causes a soft error, in other words a wrong signal somewhere in the integrated circuit. According to Siegle [64], it is ”the most common effect for SRAM-based FPGAs, as it may affect the configuration memory as well as memory cells that are used as part of the user logic (flip-flops, embedded RAM).”

Closely related to the SEU is the Multiple Bit Upset (MBU). Due to the decrease in semiconductor feature size it is possible for one particle to interact with more than one transistor. The term MBU is used if multiple corrupted bits belong to the same memory word [44]. This is because several bit flips in the same memory word cannot be corrected by basic information redundancy methods. If a single particles causes upsets in different memory words it is not considered an MBU, as common SEU mitigation techniques remain applicable. In this case the phenomenon would be a Multi Cell Upset (MCU) not resulting in an MBU [44]. It should be noted that for protons specifically, the cross section for MBUs is significantly smaller by 2-3 orders of magnitude. A heavy ion particle has a larger chance of causing an MBU due to higher LET [57].

A Single Event Transient (SET) is transient voltage pulse in the FPGA fabric caused by a particle interaction. This pulse can propagate through logic and potentially be latched into a memory element, which would cause a single event upset. For this to happen, the transient needs to occur during the setup and hold time of the register, called the window of vulnerability [58]. With the increasing clock

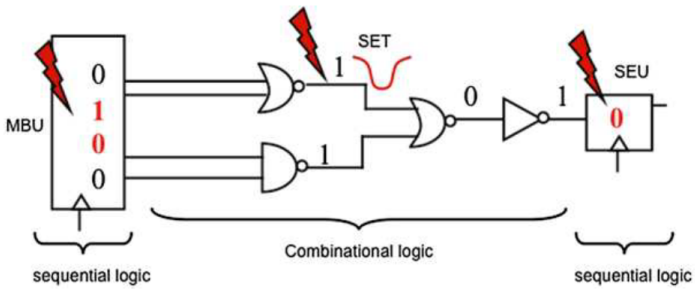


Figure 2.16: SEU and MBU occur in sequential logic, an SET occurs in combinatorial logic [44].

speeds of devices, latched SETs have become more common, however their exact contribution to the overall SEU rate is hard to observe [58]. Figure 2.16 also illustrates the difference between upsets and transients, where upsets occur in sequential logic such as a LUT and transients affect combinational logic. A transient voltage pulse is not physically harmful to the device but can cause unwanted behavior.

A Single Event Functional Interrupt (SEFI) is the most severe nondestructive event that can be expected. In this case the radiation interferes with a critical function of the device, namely internal state registers or global signals such as clock or reset. This can cause the FPGA it to lock up, reboot or fail in other ways [71]. It is considered the most severe as it renders the entire FPGA unavailable until recovered, however this type of event is relatively rare as the area allocated to critical control logic is small. [71]

In summary, instantaneous effects caused by a particle interaction can have vastly different outcomes depending on the affected component and nature of the interaction. A clear distinction exists between hard errors, that permanently damage (parts of) the device, and soft errors, that only affect the circuit state or functioning. On a modern SRAM device, soft errors happen very frequently and as such much of the research and development activities are concerned with the mitigation of soft errors. In the following two chapters, the vulnerabilities of FPGAs specifically and prominent mitigation techniques will be presented in more detail.

2.5. Research questions

Given the context of the project and the state-of-the-art of dependable FPGA-based systems for space applications, the potential for additional research was recognized. The following main research question was formulated and is to be answered by this research:

Mitigating soft errors in FPGAs significantly increases resource consumption. How can this overhead be reduced while maintaining fault tolerance attributes?

To answer the above, two sub-questions shall be investigated:

Can unused or underused resources in an unmitigated system be exploited to achieve fault tolerance with minimal functional impact on the system?

Which means to achieve fault tolerance are applicable for a system exhibiting high degrees of regularity and parallelization?

3

Implementation

The following chapter will describe the implementation of the LDPC decoder in hardware. While some initial considerations for mapping the mathematical formulation of an algorithm to digital technology were already discussed in section 2.3.3, this chapter will cover the specifics regarding this application and FPGA. First the base functional model will be presented and its mode of operation will be explained. Then the focus will be put on the PFU, the component that was studied in detail and modified the most for the proposed design. Lastly, the implementation of the duplication with compare strategy will be presented, covering considerations with respect to granularity of comparison and design of health signals.

3.1. Base Functional Model

As outlined in section 2.2, due to the concurrent development of all subsystems of EuFRATE by the consortium, the author only had access to a base functional model of the accelerator array available. Thus the main soft processors in Babylon, as well as routing and network modules were neither available to facilitate testing nor were they tested themselves. The implementation required for this research was therefore limited to the accelerator array and some auxiliary systems necessary to put the array under load and have it decode will being irradiated. The base functional model hosts a cluster of eight programmable functional units. The term cluster will exclusively be used to describe this accelerator array from here on, and will not refer to the group of several FPGAs envisioned at the top level.

The base functional model of the cluster performs decoding on a relatively small LDPC code. While the final version of EuFRATE is envisioned to work with long codewords of block length 16384 with code rate 1/2 [21], this smaller base model works with a block length of 1024 while maintaining the same code rate. Note that the block length is directly related to the size of the H matrix, and decoding on a larger matrix is inherently more computationally demanding and complex to implement. Nonetheless, the LDPC code used is compliant with the CCSDS standard and is considered a representaive benchmark for the target code [21].

Regarding the base functional model's architecture, the system is relatively simple and consists of three components. The first component is the communication interface that enables data or instruction exchange with other modules on the FPGA based on the Advanced eXtensible Interface (AXI) communication bus protocol. The AXI protocol has been adopted by AMD as their primary bus for FPGA IP, making it the de facto standard for communication buses on their adaptive platforms [8]. Some variations of the standard exist, namely AXI4-Lite and AXI4-Stream. The former is used for the configuration ports, where only a few control registers need to be set. The latter is used for the data input and output ports, and is optimized for high-speed streaming of data. Note also that the AXI4-Lite bus is memory-mapped, meaning every transaction involves a target address and data to be moved, while a AXI4-Stream bus does not utilize memory addresses for the sake of efficiency and speed [8].
Apart from communication, data storage is also required to operate the decoding cluster. This component, also denoted scratchpad in this context, not only serves as a buffer for the in- and output buses.

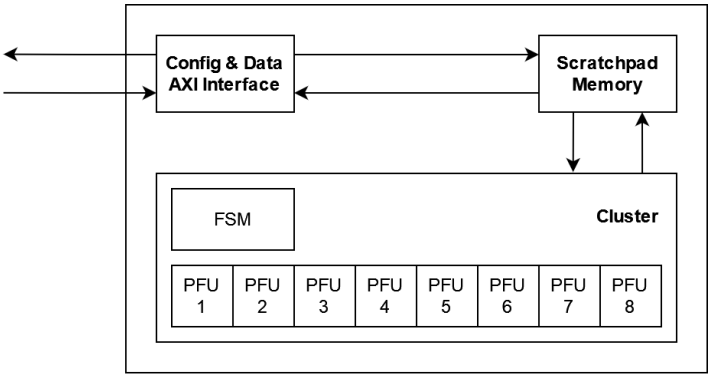


Figure 3.1: Principal components of the base functional model.

During decoding, the component is also very active feeding data to and harvesting data from the cluster based on instructions provided. It is used to store the LLR best estimate of the codeword and variable-to-check and check-to-variable messages that may get used for several computations. The scratchpad uses several block random access memories (BRAM) hard blocks available on the device for efficient hardware implementation. Since the memory controller is custom and works with offsets, phase shifts and the like, it presented itself as very nontransparent. Modifications to it where avoided as much as possible since no detailed documentation was available as a starting point.

The most important component is naturally the decoding cluster, which consists of eight PFUs. The PFUs are cascaded and share data-in and data-out buses. An outer finite state machine (FSM) manages the operation of the cluster. Six main states are defined:

1. Idle
2. Start-up
3. Compute absolute & minimum values
4. Compute check-to-variable messages
5. Update variable nodes
6. End

According to these states, and progress information collected during decoding, the FSM compiles detailed instructions for each of the PFUs in the cluster. These instructions are communicated in the form of a 50-bit control sequence that is passed in a cascading manner between PFUs too. The design of the PFU will be presented in detail in the next section.

Figure 3.1 shows an overview of the aforementioned principal components. It should be noted that the unit is mostly autonomous. The configuration interface is minimal and primarily allows for the number of decoding iterations to be changed, while the data interface is generic and will accept any data arriving via the stream interface. It is formerly the finite state machine embedded in the cluster that orchestrates the decoding by generating memory addresses for the scratchpad memory and instructions for the PFU array and packing this information into the control sequence.

3.2. Programmable Functional Unit

While changes had to be made to the majority of the system, the component affected the most is the Programmable Functional Unit (PFU). Before elaborating on the fault-tolerant version in detail, some design drivers and the PFU architecture implemented in the base functional model will be presented.

3.2.1. Architecture

The array of PFUs represents the main computing resource for digital signal processing in the EuFRATE system. Modern FPGAs, such as the Kintex UltraScale device used as target node for this project, typically contain hard blocks, which are circuits optimized for a given function such as data

storage or arithmetic operations. This is in contrast to the regular logic fabric designed primarily for flexibility. This is presented in greater detail in section 2.1. One type of hard block available is the so called Digital Signal Processing (DSP) slice, optimized for computations involving multiplication and/or accumulation [7]. For most hard blocks, AMD offers instantiation templates, that allow the designer to quickly commission a design and expose its key interfaces to the rest of the architecture [12]. These templates are also denoted primitives, and the primitive for the DSP block used here is the DSP48E2 primitive [7]. The basic functionality of the DSP48E2 primitive in shown in Figure 3.2, where several logic stages and in- and outputs can be distinguished. The 48 in the name refers to the maximum data width that can be exploited, namely 48-bit wide main in- and output signals. Furthermore several control inputs are available that allow for different operations to be performed and the changing of operations at run time. It is also possible to cascade several units for processing wider inputs or performing operations sequentially with less control infrastructure required.

EuFRATE implements a PFU architecture that is centered around the DSP48E2 slice. As outlined in section 2.3, typical mathematical operations for the Min-Sum algorithm are addition and minimum determination, as well as sign and absolute value operations. Mapping these operations onto DSP slices is sensible, as they represent hard blocks they are present on chip anyways, utilized or not, and using them can free up regular configurable logic for other applications. Figure 3.3 illustrates how the DSP slices are laid out in hardware with respect to BRAM and regular FPGA fabric. The 1-to-1 correspondence of each DSP slice and an 18kb BRAM is exploited by supplying each slice with its own tightly coupled memory. This reduces latency and complexity in routing and is an approach recommended by the device vendor AMD [7]. Thus, limiting the PFU architecture to comprise only one DSP48E2 slice, one 18K BRAM slice and some configurable logic around it is attractive and will enable a highly regular and space efficient design. It is also important to keep in mind that the application, in this case LDPC decoding, mostly influences PFU design, while other subsystems such has the Babylon processors are more universal. Constraining the footprint of the PFU is thus also desirable as it simplifies interchanging the 'LDPC PFU' for a 'CNN PFU' for example and thus gives the overall EuFRATE architecture more flexibility [21].

In greater detail, a PFU functions as follows. Apart from the usual clock and reset inputs, DATA_IN and CTRL_SEQ inputs are present. The control sequence CTRL_SEQ orchestrates the entire operation of the PFU and is a 50-bit long vector containing mostly settings bits and memory addresses. Part of that control sequence also determines whether the data on DATA_IN is saved to memory or not. As described in section 3.1, the base functional architecture cascades eight PFUs into a cluster, which shares a data bus. Thus the DATA_OUT of a PFU is directly connected to the DATA_IN of the next PFU and data is passed through unless the control sequence imposes storage of the data.
Another crucial piece of information contained in the control sequence is the configuration of the DSP slice. While most of the configuration is static in this application some settings are derived from the control sequence, namely the mathematical operation to be performed and when it shall be performed such that all inputs are ready.

3.2.2. Modifications

A number of modifications to the PFU architecture had to be implemented to allow for information redundancy. As will be described in the following, the unconventional usage of the DSP slice required a reworking of the circuitry around the slice as well as its operation.

Starting with the DSP48E2 slice, AMD supports easy single instruction multiple data (SIMD) usage. In the baseline PFU architecture, this is exploited to perform instructions on four numbers at once, by splitting the 48-bit ALU into four 12-bit ALUs, denoted FOUR12 by AMD [72]. Only 8 out of the available 12 bits are actually used, as the default quantization is 8-bit in this project. This under-utilization provides the basis for effectively increasing the data throughput in the fault-tolerant implementation. Although not officially supported by AMD, each 12-bit ALU is further subdivided to process two numbers. With a five bit quantization, this still leaves one bit for overflow detection. This is crucial since the built-in overflow detection is only available for the the officially supported SIMD modes such as FOUR12. The packing of data into the available 48 bit ALU is also visualized in Figure 3.4, where input bits are white

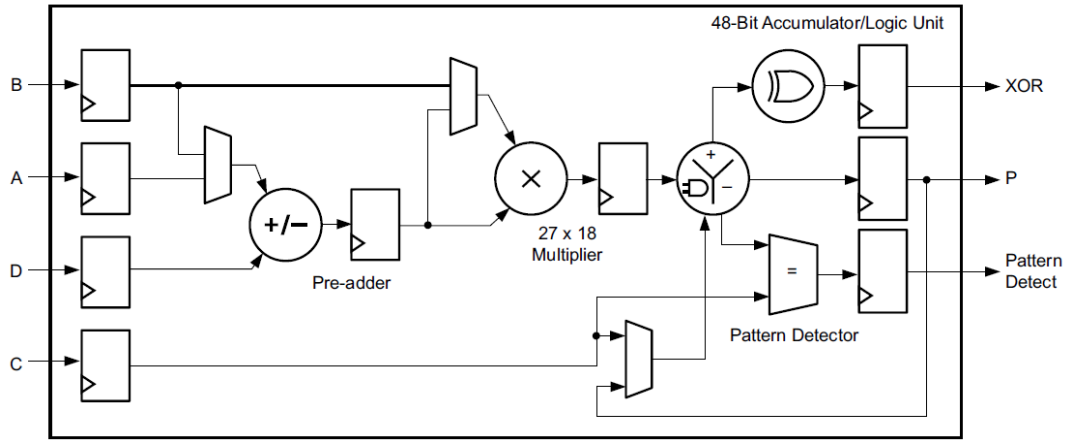


Figure 3.2: Basic DSP48E2 slice functionality [7].

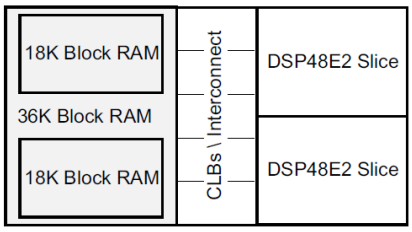


Figure 3.3: Physical layout of block RAM and DSP slices, linked via CLBs and interconnect fabric [7].

and denoted with letters, unused bits are black and bits to be used for overflow detection are grey. Although the approach to detect overflow is conventional, it will be briefly presented. Some custom circuitry is necessary for overflow detection, however the logic required is relatively little. Remembering the numbers are in two's complement format, the most significant bit indicates the sign. The fact that this bit might flip wrongly is exploited to detect overflow. If two positive numbers are added and the result is negative, the variable overflowed. The opposite condition is true for adding negative numbers. Thus, by checking the sign bits of both inputs and the output, as well as the overflow bit, positive or negative overflow can be detected in a straightforward manner. When detected, the output is simply set to the maximum or minimum value respectively.

Furthermore, the operation performed in the DSP slice had to be adjusted. This is a direct consequence of the existing architecture, which primarily performed subtraction of input B from input A. In circuitry, this is realized within the DSP slice by AMD flipping the sign of input B and then performing addition. Flipping the sign of a two's complement number is done by inverting all bits and then adding one. This works flawlessly in the supported FOUR12 mode, however two 5-bit numbers inserted into the 12 bit input will not be inverted correctly. As such, it was decided to move the inversion step out of the DSP slice and employ regular configurable logic to perform the inversion. This does not represent an ideal solution, since the circuitry used previously is part of a hard block and thus "free" in terms of area, while custom circuitry does cost area. Unfortunately, this had to be accepted since any other approach would have implied significant changes in the data flow architecture.

As described in greater detail in section 5.1.1, it was necessary to change the Min-Sum algorithm to a Normalized Min-Sum algorithm to allow for successful decoding over a broader E_b/N_0 regime. Normalization is achieved through an additional division step for check-to-variable messages. From an implementation perspective, two approaches present themselves. The DSP hard block is capable to perform multiplication or division and thus able to perform the normalization step. Alternatively, the flexibility of the FPGA can be leveraged to create custom logic for the normalization step. Considering both options, it is clear that the former can be generalized more easily for different normalization fac-

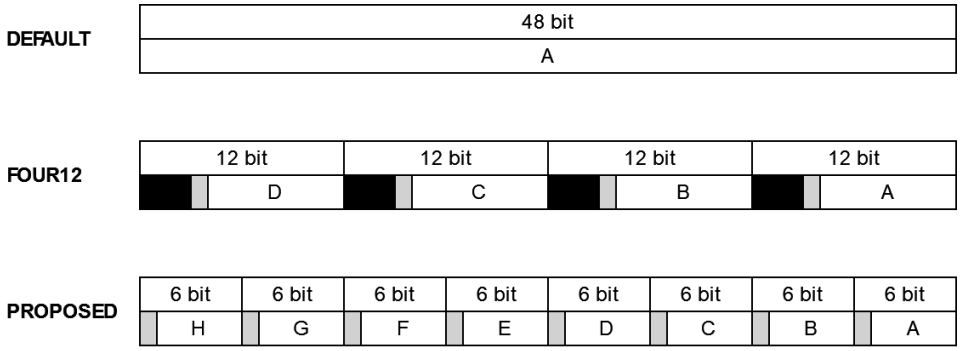


Figure 3.4: Smaller quantization enabling a doubling of the effective throughput of DSP slices.

tors and will likely take less area on chip. Nonetheless, additional computation steps will be necessary, implying a reduction in PFU throughput and increase in latency. This in turn would require a redesign of the controlling state machines and multiplexers to ensure compatibility with the rest of the system. Creating a universal multiplier/divider unit in regular FPGA fabric is not attractive as it would require significant resources. In fact the DSP slices exist mostly for the reason that the required circuitry to perform arithmetic operations cannot be implemented efficiently using regular configurable logic. This changes however when the unit is not universal anymore, but reduced to a subset of operands. Multiplication and division by two, or multiples thereof, can be realized as simple left/right shift operations on the binary string. Thus these operations can in theory be implemented purely in routing and without the use of LUTs or other logic resources available on the FPGA.

3.3. Fault Detection

Finally, after implementing some necessary modifications in the PFU, the actual fault detection mechanism was implemented. Fault detection forms one of the pillars of fault tolerance since it enables a targeted recovery of the defunct modules. In the case of EuFRATE, a fast fault detection allows for dynamic rescheduling of workloads, ultimately increasing the availability of the cluster.

3.3.1. Decoder

The purpose of the aforementioned modifications to the PFU is to save compute resources in order to allow for information redundancy and subsequent fault detection. By reducing the quantization of the LDPC decoder from 8 to 5 bit, the 48-bit ALU in the DSP slice can be used to perform eight operations simultaneously instead of the previous four. This provides the basis for the proposed duplication with compare (DWC) approach to deliver fast and reliable fault detection. In initial investigations, a duplication appeared to have the highest potential to result in a low-overhead mechanism as underutilized components could be fully exploited this way. Going for higher degrees of information redundancy, such as triple modular redundancy (TMR), showed to introduce significant overhead in initial design space exploration.

The question that presents itself is at what stage in the data flow architecture the data should be duplicated and at what stage the comparison is performed. This naturally has a strong influence on the type of fault that can be detected and the latency of detection. At first it was considered to follow a very localized approach and keep the duplication and compare operations close to the DSP slice. Following this approach, the resource overhead would be limited since most data buses can remain unchanged or decreased in size. Fault detection would also occur with very low latency, as all DSP outputs would be compared immediately and any deviation would be known to the rest of the system within a few clock cycles at most. Nonetheless, there are also significant drawbacks to this approach. First and foremost, any issues with components of the PFU other than the DSP unit will likely remain undetected. To give an example, if the tightly coupled memory exhibits a malfunction and sends wrong data to the DSP wrapper, it would be processed as usual, pass the comparison and thus not trigger any response. Furthermore, having both domains run through the same DSP and comparing after does not allow for detection of problems with the DSP itself. Those problems could be performing the wrong

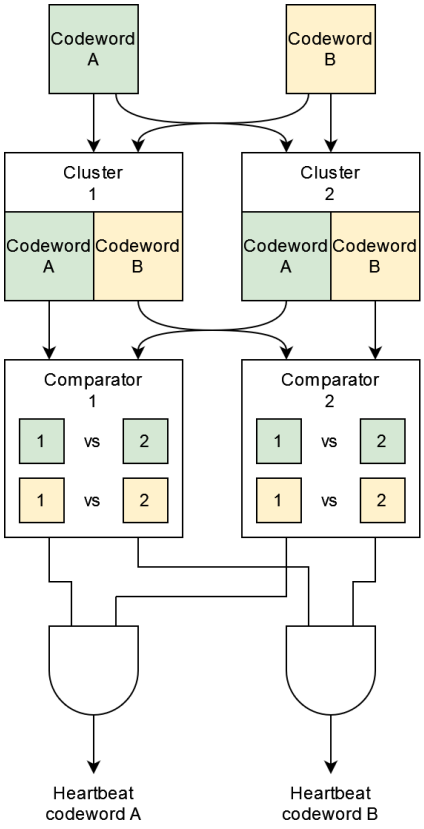


Figure 3.5: Duplication & compare data flow architecture where both clusters decode two codewords simultaneously.

operation or using a wrong input lane, since many of the DSP settings can be adjusted at run time. In conclusion, applying the DWC approach only in direct vicinity of the DSP slice would only detect a minority of potential issues resulting from single event upsets.

For the reasons described above, it was decided to apply the duplication with compare methodology at cluster level. This is also coherent with overall EuFRATE design philospohy, where a cluster of eight PFUs represents the smallest independent entity. All components previously sized for 8-bit values were redesigned to allow for 10 bit concatenations of domain A and B. Memory elements with 32 bit data width were thus modified to 40 bit data width. In principle, this allows one cluster of eight PFUs to decode the same message twice simultaneously and compare by itself. However to further improve dependability, by allowing for a larger range of failure modes to be detected, two clusters are considered a redundant group. This way, each cluster decodes two different messages simultaneously, but the two cluster perform the same computation. The comparison operation is placed at the output of the clusters. This implies that the final decoded message is compared rather than intermediate results. While this increases the latency of detection from the order of a few clock cycles to the order of 1 to 2 ms, this is still considered acceptable and the chance of a false positive is reduced. This has to do with the fact that not all upsets affect configuration memory, one has differentiate failures that affect circuit functionality and failures that affect circuit state. If a value stored in memory gets corrupted, it may not even cause the decoding to fail. Ultimately, the purpose of the LDPC decoder is to iteratively correct wrong values, whether this wrong value was introduced during transmission or in memory does not really matter. In case the comparison would occur after each and every computation step, the health signal can be expected to trigger more frequently but also likelihood that it triggered due to a correctable mistake is increased.

3.3.2. Health Signal & Watchdog

For the design of the improved health signal, a standard duplication with compare methodology is followed [59, 42]. Each cluster of 8 PFUs will decode two codewords simultaneously, denoted A & B. In the output signal both codewords are concatenated on a single 40 bit bus. To avoid single points of failure, two identical comparator modules are added, which get fed with the outputs of clusters 1 & 2. Once synchronized, the comparators will simply XOR its inputs and check for non-zero values. In case a non-zero value is detected, the outgoing heartbeat signal will seize to pulse. This is done separately for codewords A and B, leading two an output signal of width 2. The signals coming from the two comparators are then fed to an AND gate, creating a single output signal that will once again fail to pulse should the signal coming from the two modules differ. The logic to generate the health signal is thus combinatorial and instant, there is no grace period implemented that would allow for small deviations to be ignored. This was decided since no scenario could be identified in which a divergence would not indicate a fault. The duplication with compare infrastructure is also shown in Figure 3.5, where it can be seen that there are several cross-overs between the two decoding clusters and the fault detection modules. In summary, the health signal is generated in two steps, where firstly the outputs of the redundant clusters are compared in two modules and secondly the agreement of both comparisons is checked. Following this approach, the health signals generation is linked to the decoding load rather than it being mostly independent circuitry.

The health signal is monitored by a custom watchdog module, whose purpose is to inform the MicroBlaze soft processor in case of any irregularity. A watchdog is very common in systems that strive for high dependability or face threats such as radiation effects. A watchdog is typically ”kicked” regularly to inform it that the monitored systems are still available. The module is designed to monitor several health signals at once, thus only one watchdog is required per type of health signal. To realize this simultaneous monitoring, the health signals are simply concatenated. Since no guarantee can be given on the synchronization of the health signals coming from different clusters, the watchdog can be configured to account for offsets between the incoming signals. While the signals might not be in sync, they will repeat the same pattern after a number of clock cycles. Thus instead of an all-zero or all-one signal, the pattern of the signal is recorded and taken as reference. Should a deviation of the pattern be detected, the watchdog will trigger an interrupt signal to the processor and also save an identifier for the affected signal, which can then be read by the processor from the respective register. Note that a single watchdog module can quickly become a single point of failure, however in this research the focus was put on health signal generation rather than the watchdog itself. The function of monitoring health signals will be fulfilled by a more complex beacon controller in the EuFRATE architecture [21].

4

Experimental Test Setup

In this chapter, the experimental test campaign will be presented. Three key aspects will be discussed, starting with the radiation source. The facility, physical test setup and applicability of proton radiation will be elaborated. Next, the test setup will be given, where the device under test and its workload are to be presented first. Then the auxiliary systems will be presented. These on-chip modules were necessary to facilitate data extraction from the FPGA and to ensure a larger area on chip will be occupied by the device under test. Lastly, the test procedure will be given. The scope of each of the four experiments will be explained and the procedure to be followed during the experiment is also presented. The experimental campaign was the primary data source with respect to dependability attributes of the system and a core part of this research.

4.1. Radiation Testing

The test setup was irradiated by a proton source to investigate radiation effects and the effectiveness of the fault tolerance measures.

4.1.1. Facility & Physical Setup

All tests were conducted at HollandPTC in Delft, The Netherlands. The facility is primarily used for the medical treatment of cancer patients, however a versatile lab room is also available and frequently used for medical and embedded systems research.

The proton beam can be shaped according to requirements. Two options were relevant to this test campaign: the 5x5 cm collimated beam and the so called pencil beam, which irradiates an area of a few square millimeters [39]. While the pencil beam is attractive for its precise targeting, using it would require information about the physical location of certain components. One would want to only target the device under test, the decoding cluster(s), however no tool is available to extract this information and AMD is likely not interested in making this information publicly available. As such the facility was set up for the 5x5 cm square field and the FPGA was entirely located in the beam.

Several pieces of equipment are necessary to produce a proton beam that is uniform in flux and particle energy level. The beam arrives in the laboratory room at the so called beam flange. This beam flange, as well as the cyclotron particle accelerator itself, is kept under vacuum. The beam itself has a small cross section at this point, thus a scattering foil is placed behind the beam flange. This foil is made of tantalum, a highly inert metal. Behind the scattering foil is the beam monitor, which measures the flux during irradition. This device is an ionization chamber that generates a current proportional to the number of ionizied particles travelling through it. Behind the monitor is a dual ring setup that further ensures uniform energy levels in the beam. Two large brass collimators are used to reduce the size of the beam again to a 5 cm square field, absorbing the other particles. Finally, 12 cm behind the last collimator is the isocenter. The isocenter is the point along the beamline used for calibration, as such the beam properties are defined for said point [26]. Far behind the table used for experiments an absorber is placed to no irradiate the building itself.

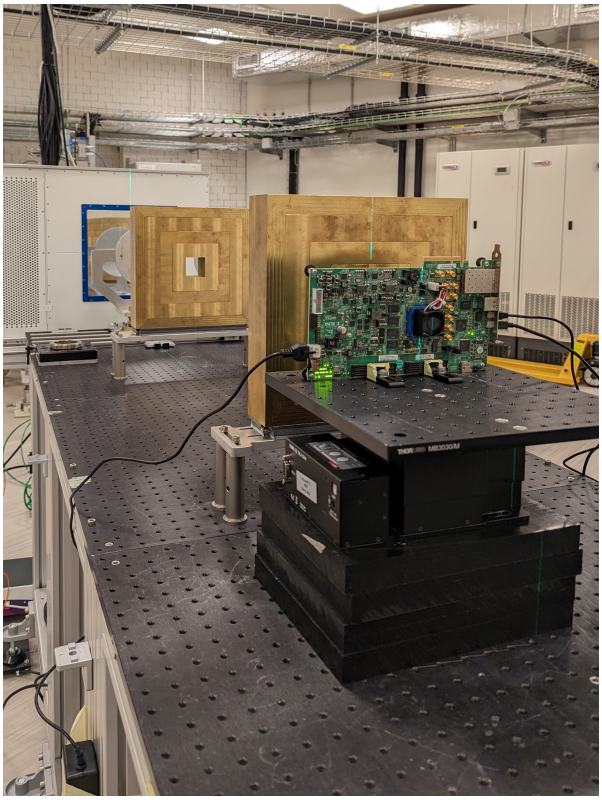


Figure 4.1: Mounting of the development board & beam line components.

The development board hosting the FPGA was mounted vertically, with its backside oriented toward the radiation source. This was done to eliminate any interference that the fan assembly on top of the chip could cause. The chip was centered in the field using two laser levels. The board power supply and the lab computer could be placed next to the experiment table and were thus not subject to significant radiation.

Regarding the proton energy level, a few fixed options are available at HollandPTC. The options are namely 70 MeV, 120 MeV and 200 MeV at target. These energy levels are representative of particles encountered in space although even higher energy particles may be encountered [65]. Generally speaking, it can be expected that higher energy protons are more likely to cause soft errors. Since the upset rate for certain energy levels is more of a device property, and not related to the configured circuit per se, it was decided not to perform extensive testing at all available energy levels. Instead it was preferred to collect a larger data set at one energy, namely 120 MeV. A small number of irradiations were performed at 200 MeV nonetheless. The primary parameter of interest when it comes to quantifying the sensitivity of the device's configuration memory to radiation effects is the cross-section. This value captures the number of single event upsets normalized by the fluence as well as the total number of bits considered. The normalization ensures that the resulting value is a device property and decoupled from the configured design and test conditions. Cross-section values are calculated according to Equation 5.1 and will be given for both particle energies in section 5.2.1.

4.1.2. Applicability

Since the choice of radiation source and facility was primarily driven by practical considerations, applicability needs to be discussed for this method of testing. As introduced in section 2.4, the space radiation environment is comprised of three types of particles: electrons, protons and heavy ions. As also described in the background chapter, the particles can vary greatly in energy, ranging from a few keV to several GeV for heavy ions. Not only the particle's energy is important to consider however, but also a property called Linear Energy Transfer (LET). The exact definitions of LET may vary, but the different models all relate to stopping power, the energy that the particle loses by travelling a certain

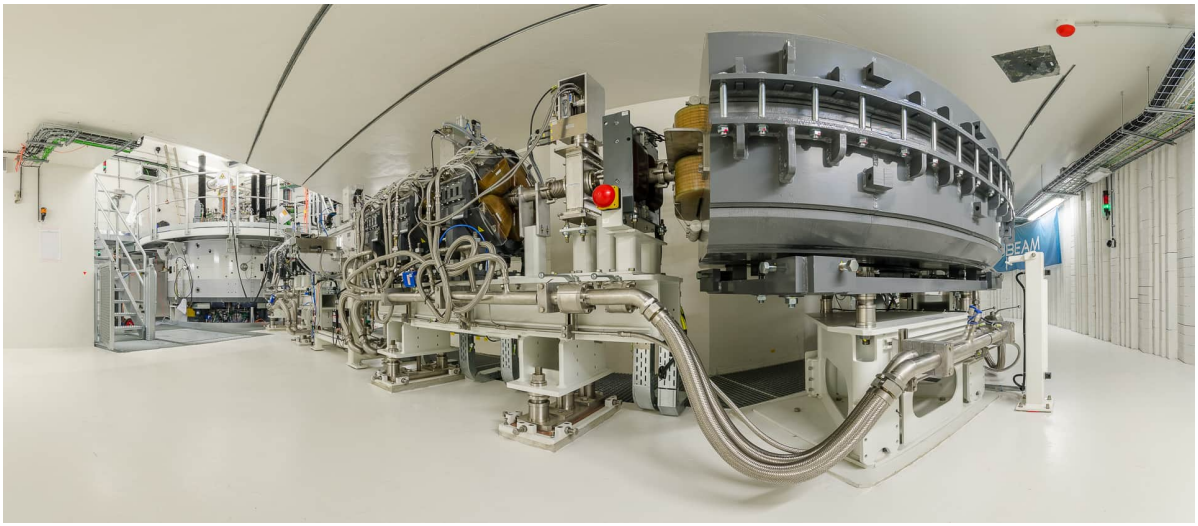


Figure 4.2: Superconducting cyclotron accelerator for proton testing, located in Delft [28].

distance in matter [70]. Note that this is not a single value, but rather a distribution over the penetration path length. Nonetheless, for practical considerations an average value is often used and for the remainder of this discussion a single LET value per particle type & energy is assumed.

Generally speaking, higher energy particles will exhibit higher LET values and particles with higher LET will be more detrimental to the semiconductor material. This is because the deposited charge will be larger, as well as the affected area on chip. Thus higher LET particles are more likely to cause MBUs and other complex failure modes. It should also be noted that most semiconductor devices have a threshold LET below which no single event effects occur, although determining the exact value with high statistical certainty is challenging [41]. FPGAs based on SRAM technology are characterized by a very low threshold value however, making them vulnerable to heavy ions, protons and electrons above certain energies [34]. This implies that proton radiation at the available energy levels is sufficient to cause single event effects and thus the system's response to faults can be studied using a proton source.

From the above context it is clear that exhaustive testing, validating the system against various radiation effects and complex failure modes, cannot be done using just one type of facility. Nevertheless, heavy ion testing can generate valuable insight about the necessity of additional testing using other sources. For radiation-hardened devices, if no susceptibility is found for heavy ions with LET less than approximately $15\,\mathrm{MeVcm}^2/\mathrm{mg}$, the usefulness of additional proton or neutron testing can be contested [53]. For non-radiation-hardened ICs, susceptibility is to be expected and the same does not apply. Furthermore, since protons do not exhibit high LET values, they cannot substitute for heavy ions. This leads to the conclusion that heavy ion testing is required for a thorough validation of most systems. This induces a high demand for this method of testing, while on the other hand the number of facilities is limited, a 2021 report by ESA reported only six suitable facilities in Europe [54]. This scarcity of beam time leads to an increase in cost, both in financial and scheduling terms [53]. The same cannot be said for proton beams, where the availability is rapidly increasing with the increased usage in cancer therapy. In Europe alone, the number has doubled to a total of 31 facilities between 2017 and 2020 [36]. Although the primary purpose of these facilities is not the study of embedded systems, many of the facilities can be used for this purpose at times where patients are not treated, namely in the evening or on the weekend.

It should be evident that proton testing is an essential step when developing reliable space systems. While it cannot validate a component entirely, heavy ions may cause different and more severe failure modes, it does provide valuable insight. Depending on the exact mission profile, it can be expected that SEEs due to protons represent the majority of events. Due to the financial and scheduling cost

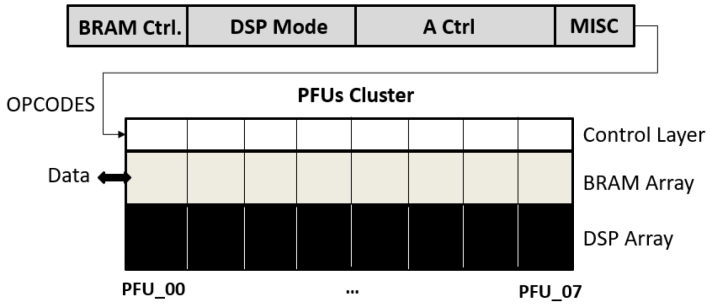


Figure 4.3: Constituents of the opcode control sequence, representing the low level control layer [21].

of heavy ion testing, this method of testing should represent the very last validation step for any design. It is fair to say that for more exploratory research as conducted in this project, heavy ion beam time would be too valuable. In the context of the EuFRATE project more testing is required anyhow, specifically targeting the other subsystems such as Babylon that were not included in this campaign. The fully integrated design also needs testing, as testing of individual components does not guarantee dependable interfaces.

4.2. Test Setup

In the following section, the test setup will be given and motivated. Furthermore the procedure followed during testing will be explained.

To test the radiation tolerance attributes of the accelerator array, a test setup had to be developed that meets a number of requirements. First and foremost, a realistic workload has to be computed by the device under test while irradiated. The cluster must thus be configured appropriately and supplied with input data. Furthermore the output data must be captured and made available for evaluation after the irradiations are complete. It shall also be extracted how many configuration upsets occurred during irradiation.

4.2.1. Device Under Test & Workload

The workload emulation was supplied by Argotec in the form of a base functional model of the cluster. Said model takes a 2 kB input message generated by a separate data model run externally. The message is pre-processed with each message bit already represented as a log likelihood ratio. Depending on the channel noise settings in the data model, a certain number of wrong bits are present in the message. As such the input message can be considered realistic. While it would be possible to generate a large number of randomly generated input messages, this would not lead to any more insight into the dependability performance of the cluster. As such, the same input message is fed to the system repeatedly, which also simplifies the checking for errors in the output message after irradiation.
To supply the cluster with instructions, the base functional model also contains a complete state machine that generates the so called opcode that tells the PFUs what data to load, store or process a certain way. This opcode is also shown in Figure 4.3, where some of the constituents of the code are listed and the conceptual control layer is illustrated. The model also has a few registers for settings, allowing to configure the number of decoding iterations among other process variables. These settings will be represented in the generated opcodes. Regarding the LDPC code itself, the H matrix is deeply embedded in this model and not easily changed. Once again, this does not have any major implications for an analysis of fault tolerance attributes and is not considered a limitation.

In summary, the device under test is self-contained to a certain degree. In- and outputs to the device are shown in Figure 4.4. Catering to the interfaces visible and writing to the necessary settings registers is sufficient to get the unit to run a realistic workload in hardware. On the input side there are the typical clock and reset inputs CLK and RST. On the output side there is the DONE signal going high when the cluster has completed its decoding task, as well as a health signal called CL_ALIVE and some optional debug signals. Three of the interfaces follow the AXI communication bus protocol: CONFIG, DATA_IN

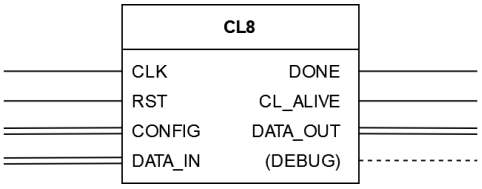


Figure 4.4: Interfaces of the base functional model of the cluster containing 8 PFUs.

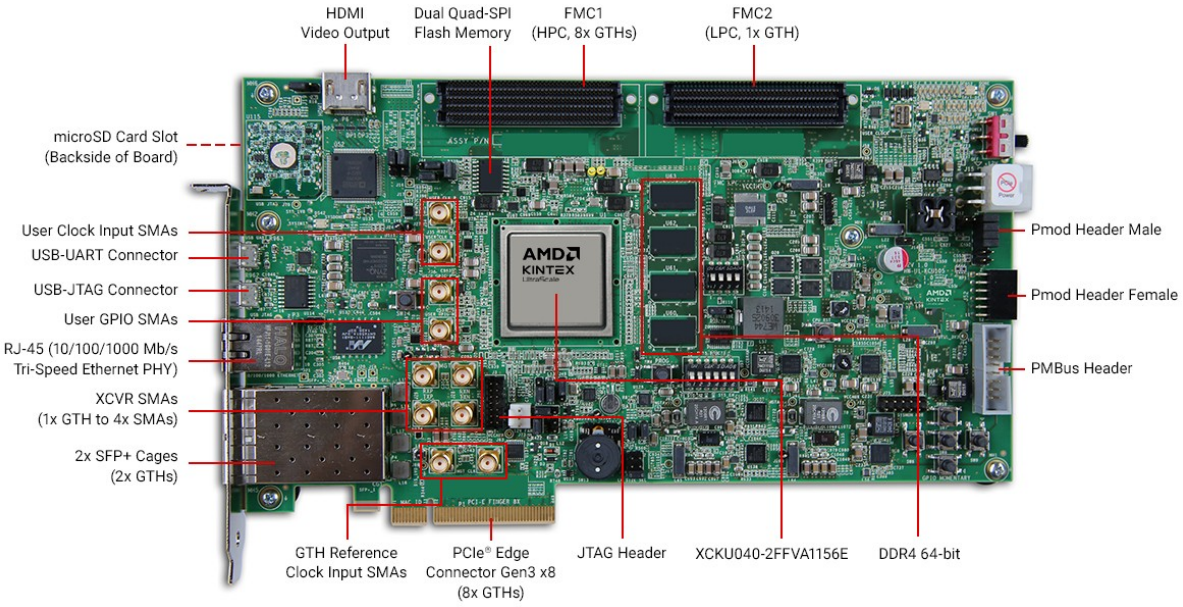


Figure 4.5: The KCU105 development board used in the experimental campaign [11].

and DATA_OUT. More precisely, CONFIG is a AXI4 Lite slave, while DATA_IN and DATA_OUT are AXI4 Stream slave and master respectively. More information on this protocol and the difference between the Lite and Stream versions can be found in section 3.1.

4.2.2. Auxiliary Systems

Specifically to extract data from the device under test, some additional components had to be developed and commissioned. These components are collectively called auxiliary systems and will be presented here. The systems are partially provided by AMD as intellectual property (IP) licensed together with the Vivado development environment. In this case the components only had to be configured and their interfaces carefully matched to the rest of the architecture. In other cases, custom components were developed from scratch to address a specific task. Naturally, the advantage of an FPGA platform is that all auxiliary components can be hosted on the same chip.

The bridge between the memory-mapped and stream domain is typically a Direct Memory Access (DMA) engine, which is also used in this experimental setup. The DMA engine thus takes address inputs and generates the stream of data to be sent to the cluster and is also responsible for receiving the output data and storing it back in memory, to make it available to the rest of the system. The IP used as DMA engine is provided free-to-use in Vivado and only had to be configured appropriately. The KCU105 development board used in this project has 2GB of DDR4 memory available, which is used to store all data related to decoding. As can be seen in Figure 4.5, the memory bank is a set of four chips approximately 1 cm removed from the FPGA. This has the advantage that the memory bank can be kept outside the 5 cm square collimated beam and will not be irradiated, guaranteeing that any errors have been introduced in the FPGA and not in memory after processing.

In order to retrieve the processed data after irradiation, an interface with a lab computer must be realized. The KCU105 offers several options for this, of which the following were considered: UART, SD Card, Ethernet and PCIe. While the latter two would be preferred due to their high-throughput, the complexity of implementation was considered too high. Using the Ethernet interface would have required a network architecture and additional hardware, packaging the data on the FPGA and guaranteeing compliance with some networking protocol such as UDP or TCP/IP. Leveraging the PCIe connector would have required an additional host board with operating system and significant development of drivers to extract data. As such, both options were discarded as they would have entailed significant development and testing effort outside of the FPGA. Using the SD card slot to store data was considered, however the slot is intended to store configuration bitstream files for the FPGA, allowing the designer to load from there instead of the flash memory. Once again, using the port would have entailed significant effort to program a memory controller that can write to the card from the FPGA, shifting the scope of the project.
The only connection that is relatively straightforward to configure is the UART via USB interface. UART is a simple serial link protocol that is widely used in the world of embedded systems and digital systems in general. Downsides are that the protocol efficiency is low at 80% in the 8N1 setup, meaning only 80% of transmitted bits are actual information bits. It is also relatively slow, supporting baud rates of up to 921600 Hz, putting its maximum bit rate at less than 800 kb/s. As opposed to the other interfaces, an IP core is provided with Vivado to set up the serial link. UART terminals for the lab computer are also readily available.

Considering the described range of components to be included in the design, it becomes clear that an overarching controller entity is required. This controller needs to configure the other components and orchestrate the data flow during testing, partially fulfilling the role of the Babylon subsystem in the EuFRATE architecture. Two main options present themselves: a custom bare-metal controller or a soft processor running an operating system and test script. With respect to complexity, both options score similarly for different reasons. A bare-metal controller would be written in VHDL and become part of the circuitry, implying a limited observability of state and adjustability during testing. Using a soft processor comes at the cost of a higher resource footprint and adds another step to the tool chain. On the other hand it allows for a more typical software debugging experience when developing the script. Furthermore the test procedure can be adjusted on the fly as long as there are no changes in hardware associated.

The natural choice for a soft processor on AMD hardware is a MicroBlaze processor, as offered as part of Vivado. While the processor is more powerful than would be required for this application, it does offer several key options for customization. Furthermore the extensive documentation made available by AMD facilitates incremental development. One MicroBlaze feature that was exploited for this project is the interrupt interface support. Interrupt signals allow entities outside the processor to get the processors immediate or as-soon-as-possible attention, and thus adds responsiveness and a real-time element to the system. For each interrupt, an interrupt handler is created that will execute on the soft processor and handle what the situation demands. Several interrupt signals are used in the system. The DMA engine notifies the MicroBlaze of completed transfers to or from memory via interrupt. The same is true for the UART module communicating with the lab computer. An interrupt signal is also the natural choice for the interface between a watchdog module and the processor. The MicroBlaze IP was configured in microcontroller preset and optimized for area, as it was known that the computational load would be relatively low. A selection of the configuration settings is shown in Table 4.1.

Lastly, a clock module is present to stimulate all other components. While clock generation and distribution in the FPGA fabric are complex matters, this is mostly automated and implemented by Vivado without user intervention. The clock speed chosen for this project is $100\,\mathrm{MHz}$, as it was for the incremental test benches developed by Argotec [21].

4.2.3. Utilization

The utilization of the FPGA is an important metric for any design. For this study, a high utilization is desirable as more data on faults can be collected, since the device will be irradiated uniformly. Another

Table 4.1: MicroBlaze soft processor settings.

Setting	Value
Configuration	Microcontroller Preset
Implementation	32 bit
Implementation Optimization	Area
Barrel Shifter	Yes
Floating Point Unit	None
Integer Multiplier	32 bit
Local Memory Bus	Instruction & Data
Peripheral AXI Interface	Data

Table 4.2: Utilization of the device under test (DUT) and auxiliary systems (AUX) during baseline version experiments.

	Total	Utilization	Utilization %	DUT	AUX	Ratio
Logic LUT	242400	179318	73.98%	59.78%	14.19%	4.21
FF	484800	190502	39.29%	29.90%	9.40%	3.18
BRAM	1200	474	39.50%	29.33%	10.17%	2.89
DSP	1920	134	6.98%	6.67%	0.31%	21.33

requirement is that the ratio between resources allocated to PFUs and resources allocated to auxiliary systems is maximized. This is to increase the likelihood that the auxiliary systems remain functional and can be used to extract data on the failing or failed device under test, namely the PFU arrays. A hierarchical breakdown of utilization per subsystem and individual component can be extracted from Vivado. The data becomes available post-implementation and thus directly corresponds to the actual resources configured on the device.

Given the fact that strictly necessary modules such as the DMA engine or the soft processor are lower-bound in resource consumption, the number of decoding clusters on chip was increased to improve the ratio between the two. A total of 16 clusters hosting 8 PFUs each was implemented. The AXI Stream input signal to the PFU was simply split before and the output signals were combined after being processed by the clusters through an XOR operation. This approach is visualized in Figure 4.6. Replicating the cluster 16 times led to a larger amount of resources associated to said components, as shown in Table 4.2 and Table 4.3. Some variation based on the type of resource can be observed, but the overall ratio of DUT to auxiliary resources is around 2.8 to 4.2. This implies that the number of recorded configuration upsets by component will be skewed similarly.

4.2.4. Data Compression

Beam time at HollandPTC is valuable, in the sense that there is a big demand for the experimental time from both medicine and space researchers. As such, the value and data extracted in each experiment should be maximized.

It was recognized in early testing that the UART serial connection poses a significant bottleneck in this test setup. In order to not throttle the PFU clusters, they load their input and store their output

Table 4.3: Utilization of the device under test (DUT) and auxiliary systems (AUX) during improved version experiments.

	Total	Utilization	Utilization %	DUT	AUX	Ratio
Logic LUT	242400	182958	75.48%	58.35%	17.13%	3.41
FF	484800	204327	42.15%	31.28%	10.86%	2.88
BRAM	1200	555	46.25%	36.00%	10.25%	3.51
DSP	1920	134	6.98%	6.67%	0.31%	21.33

data directly in the DDR4 memory, which is sufficiently fast. The slow UART connection thus does not impact the design under load. Nevertheless, the cluster output data needs to be recovered from DDR4 memory and relayed to the lab computer after the irradiation is complete. Considering that a CL8 unit produces about 2.5 kB of data in as little as 1 ms, the data rate is about 2.5 MB/s during operation depending on the number of decoding iterations selected. In contrast, the UART connection in the given configuration stays below 100 kB/s. As such, if all output data is to be downloaded, each second of irradiation would require some 25+ seconds of data download. This imbalance would imply that out of a five hour experiment at HollandPTC, more than 4:45 would be spent downloading data from the development board to the lab computer.

To circumvent the above, and improve the ratio between irradiation and data download time, some of the data processing was moved to the FPGA. Instead of transmitting all raw data, the script executed on the MicroBlaze soft processor handles some pre-processing of the data. Due to the XOR operation described in the previous section, a fault-free test cycle will produce an all zero output. This is checked by the processor, which will keep count of correct and incorrect bytes. Instead of transmitting correct output data, the number of correct bytes will be transmitted just as the first faulty byte is detected. A specific binary phrase is adopted to distinguish this meta information from raw data in the log. For an error-free test cycle, this measure reduces the amount of data to be sent via UART from 2.5 kB to just 8 bytes. In the worst case, where all bytes are faulty, the amount of data stays the same, while anything in between is naturally also possible for partially corrupted data.
To further reduce the amount of data, a rudimentary error signature detection was also added to the test script. While going over the output data of various test cycles, the first few bytes of the cycle are saved temporarily, given that the the output was faulty. For subsequent data, it is first checked whether this signature is identical for both cycles. If so, once again a specific binary phrase will be transmitted via UART instead of the faulty data, telling the user that the output data of this cycle is identical to the previous cycle. This failure mode is relatively common in FPGAs and related to faults that affect circuit functionality. One can imagine a DSP slice that subtracts instead of adds because of a configuration error. Until another fault is introduced, this DSP slice will perform the same (wrong) operation for the next cycles, producing the same (wrong) data.

With the data compression measures outlined above, it was possible to restore balance between irradiation and data download time, although the download typically still takes longer. The beam time at HollandPTC was used more effectively and several dozen irradiation runs were possible per experiment. A downside of this approach is that some of the data processing is run on the FPGA instead of the lab computer. This is namely an issue since everything on the FPGA is being irradiated, and as such no guarantee can be given that the MicroBlaze will correctly perform its data processing task. Initial observations showed however, that the processor is likely to be either working correctly or entirely unavailable, and unlikely to fall in between those two extremes.

4.3. Test Procedure

Apart from developing the necessary auxiliary systems for the test campaign, the procedure during testing also needs to be considered. Once again, the goal for the procedure is to effectively use the beam time and to maximize the data output. In the following section, the nature of the available data will be given first. Then the procedure to be followed during testing will be elaborated. Lastly, the incremental approach to the scope of each of the four tests will be motivated.

Two primary sources of data can be distinguished. As described earlier, all output data generated by the clusters is first saved in DDR4 memory and then selectively transmitted to the lab computer. This data is saved in the form of a raw binary log file, which can then be evaluated after the experiment to exactly determine the point in time where the decoding failed. By closer inspection, these log files can also be used to distinguish some failure modes, for example whether the functionality of the cluster was affected or whether only a single value was affected.
The second important data point is the total number of configuration upsets that occurred during irradiation. For this, the Hardware Manager available in the Vivado IDE can be used. Through the ”Verify

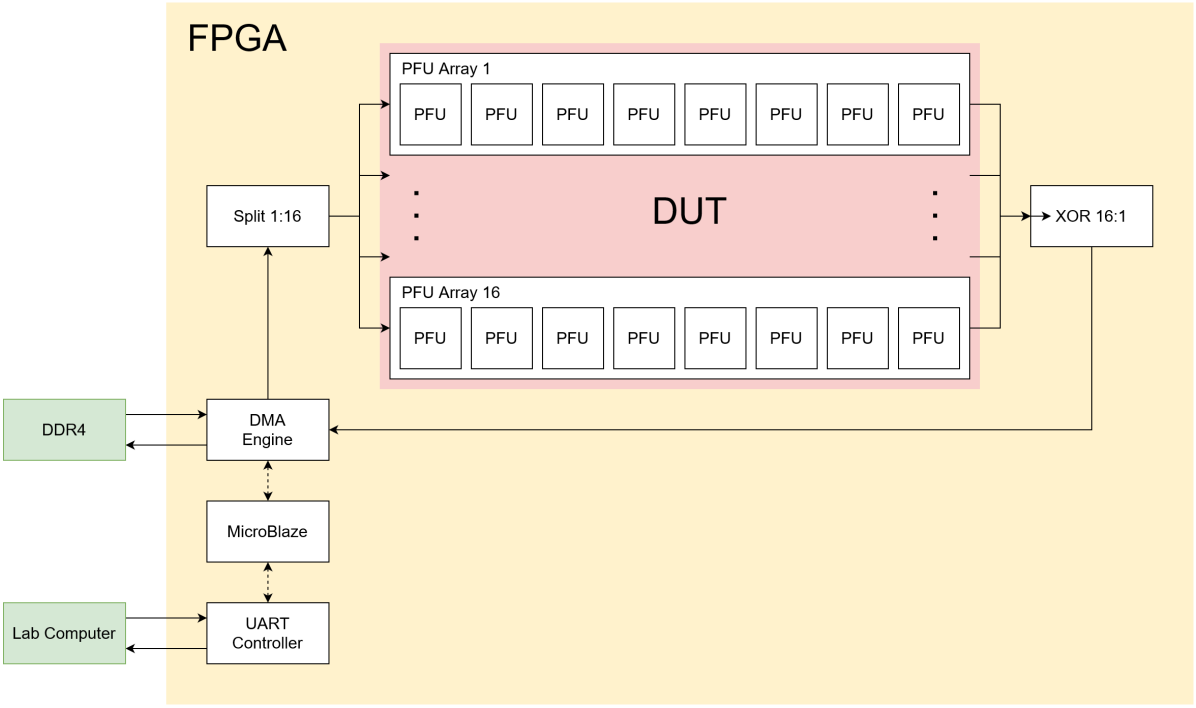


Figure 4.6: Test setup for the baseline, non-radiation-tolerant cluster.

device" function, the bitstream will be recovered as-is from the device and compared to the initial, upset-free bitstream. Furthermore a mask file is used to filter out unused configuration bits and dynamic bits e.g. used for memory [9]. AMD denotes this essential bits, bits that are used in a design and as such *may* lead to failure when flipped. There is an even smaller subset of critical bits, which are *guaranteed* to lead to failure when flipped. Determining the latter is not straightforward however, and typically done by systematic and exhaustive fault injection. More precise information about the location and time of occurrence of the upset is unfortunately also not available as it would require the use of AMD's Soft Error Mitigation (SEM) IP or similar. Nonetheless, extracting the total number of bitstream errors after radiation provides a valuable data point to judge the overall dependability of the system and its ability to function in the presence of faults.

During each experiment, a comparable test procedure was followed although the objectives of the experiments may vary. This procedure is shown in Figure 4.7, where three main phases can be distinguished. At first the FPGA needs to be configured and known to be good input data is received via UART. This is too ensure that each irradiation run has the exact same starting conditions. In the second phase, the chip is being irradiated while under load. As described earlier, the workload is imposed by repeatedly loading the same input data from memory and storing the output data there as well. Since one decoding cycle may only take 1 to 2 ms, the cluster typically computes thousands of decoding cycles while irradiated. A common value during testing was a radiation exposure time of $20\,\mathrm{s}$ and $10\,000$ to $20\,000$ test cycles, although this was varied regularly too. The third phase of the test procedure begins after irradiation is complete and concerns data recovery. Via the UART connection, the user is able to determine whether the MicroBlaze processor is still available. If so, the cluster's output data is pre-processed as described in section 4.2.4 and then transmitted via UART. If the processor become unavailable, this step is not possible. In both cases, the test procedure is concluded by extracting the number of upsets in the bitstream via readback & verify. After storing the log files and noting any other observations made, the procedure is repeated.

In total, four radiation experiments were conducted at HollandPTC over a period of several months. The methodology was to incrementally increase the scope of the tests and adjust according to lessons learnt during the process.

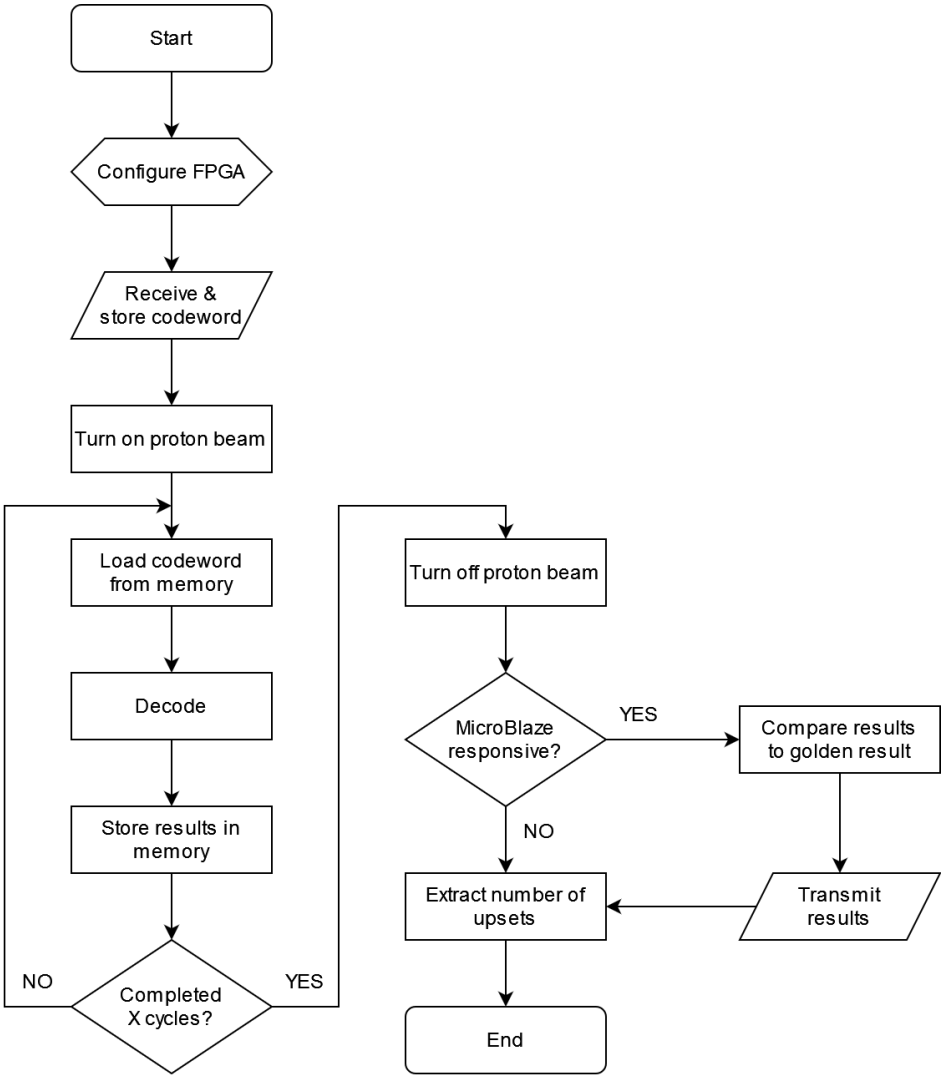


Figure 4.7: Test procedure to extract upsets and output data after irradiation.

Table 4.4: Overview of the objectives for the four radiation experiments.

Experiment	Objectives
1	- Establish physical test setup - Familiarize with proton beam settings - Validate test setup - Explore effects of flux/fluence
2	- Validate test setup - Explore effects of energy level - Collect data on baseline
3	- Collect data on baseline cluster - Analyze baseline health signal
4	- Collect data on fault-tolerant cluster - Analyze improved health signals

The primary objectives of the first experiment were to establish and validate the test setup. This encompassed setting up the beam shaping devices and mounting the development board at the exact position for which the flux is calibrated at the facility. An introduction to the proton beam and its adjustable parameters was also necessary. After establishing this physical setup, which would remain unchanged for the rest of the experimental campaign, the test setup can be validated. The important question to answer is whether beam settings can be identified for which the on-chip modules of the test setup survive such that data can be extracted reliably.
This validation process was continued in the second experiment and expanded to include new developments with respect to data compression. Furthermore the possible effects of higher energy particles were investigated and the data set on the un-mitigated baseline cluster was grown.
The third experiment put the focus on health signals and included a custom watchdog module for the first time. More data about the baseline cluster was collected and some high fluence stress tests were conducted to investigate the systems response as a whole and particularly the health signals.
In the fourth and final experiment the fault-tolerant version was put to test including the improved health signal. Thus the last experiment served to test the improvements proposed in this work.

Overall, some 140 irradiation where conducted. Between the four experiments, the test setup was continuously improved and performed reasonably well. This is considering that the modules necessary on chip were not fault-tolerant and expected to behave unreliably. Further detail on this will be given in the results chapter.

5 Results

In this chapter, the results of the research project will be presented. Results need to be collected in three domains and synthesized before overall results and conclusions can be derived. These three domains are the LDPC decoding performance, the dependability of the system and the associated resource overhead. Data for the three domains can be collected independently, decoding performance was mostly analyzed using a dedicated data model. The dependability was tested in the radiation facility as described in the previous chapter. The footprint of the design was extracted from the integrated development environment. The results across all domains will be presented in the following sections.

5.1. LDCP Decoding

The decoding performance of LDPC codes can be quantified as the bit error rate (BER) after decoding has been completed. Naturally, the difficulty of the decoding task is directly correlated to the error rate in the received message, before decoding begins. This in turn is purely dependent on the signal-to-noise ratio for many channels. In this case, a binary additive white gaussian noise channel is modelled. The decoding performance can then be plotted as the BER against the signal-to-noise ratio per bit, as shown in Figure 2.8. Three regions can be distinguished and are characteristic for LDPC codes: At low signal-to-noise ratios, decoding practically fails and the BER is too high for all intents and purposes. After a threshold SNR, the waterfall region is entered where the BER rapidly decreases with increasing SNR. Lastly, the code will enter the so called error floor, where increases in SNR only lead to marginal reductions in error rate.

As presented in section 2.3, there are many parameters that influence the performance of a given LDPC code and decoder. The construction of the H-matrix, variable and check node degrees as well as word length and code rate can be varied with significant effect and present a complex optimization problem on their own. Since the primary objective of this study is to re-allocate underutilized compute resources to improve fault tolerance, the underlying LDCP code was left mostly untouched to not skew results. The analysis was limited to the quantization of the implementation, as a lower data width will consume less resources on-chip. The Min-Sum algorithm was changed to a Normalized Min-Sum algorithm however, as the non-normalized code did not perform well in the base functional model. A non-normalized code was deemed acceptable previously as the decoding load purely represented a dummy load, however when investigating the effect of quantization, the baseline should perform reasonably well.

5.1.1. Normalization

Before considering the effect of different quantization schemes, the results of the normalization will be presented. As typical for the evaluation of digital modulation and encoding schemes, the BER after decoding will be evaluated against the signal to noise ratio per bit, denoted E_b/N_0. The two implementations to be compared are the floating point decoder, using 32 bit per LLR, and the baseline fixed point decoder using 8 bit data width. The exact fixed point format is Q4.3, implying a leading sign bit, four integer bits and three fractional bits. Thus the range that can be represented is -16 to 15.875

with a resolution of 0.125. None of the initial LLRs is clipped, all received values fall within the range. The performance of the regular, non-normalized Min-Sum decoders is shown in Figure 5.1, with the BER after decoding on the y axis and the E_b/N_0 on the x axis. Clearly the fixed point implementation does not behave as expected, the BER should steadily decrease with increasing E_b/N_0. Instead it can be observed that the BER increases after E_b/N_0 of higher than 3.5 dB. A similar behavior is not present when using only three decoding iterations. This led to the conclusion that the non-normalized fixed point implementation present in the base functional model is unstable and will fail at higher iterations.

Due to considerations regarding the complexity of the multiplication in hardware, only a small selection of normalization factors was investigated. These factors were 0.25, 0.5 and 0.75. The resulting performance over a range of E_b/N_0 inputs is shown in Figure 5.2. It can be observed that all three normalization factors alleviate the instability issue. However it can also be seen that a sub-optimal normalization factor does not lead to better decoding performance. Using a factor of 0.25 decreases the magnitude of the messages to a degree where the decoder becomes sluggish and cannot correct errors in the message within six iterations. Using a factor of 0.75 leads to very good results, it is clear from the figure that this version quickly enters a steep waterfall region at relatively low E_b/N_0 values. Using a normalization factor of 0.5 leads to successful decoding within six iterations, however the performance with respect to noise is about 0.5-1 dB worse than in the case of 0.75.

Ultimately, a normalization factor of 0.5 was selected as a compromise and implemented on the FPGA. This factor might not be ideal, but at this point it should be highlighted again that investigations into optimal LDPC decoder architectures are not a core part of this study. A factor $1/2$ over $3/4$ is significantly easier to implement in hardware as discussed previously. The right shift and zero padding operations for divisions by multiples of two are cheap, however a factor $3/4$ would have also required addition, which in turn would have led to custom logic to realize the adder or a change in the data flow architecture to run an additional addition in the DSP slice. Throughout the project, the latter was avoided as many subsystems need to be data synchronized, so modifying the data flow architecture easily introduces errors which in turn are very difficult to trace back to the original timing issue.

In summary, due to the baseline Min-Sum decoder failing at higher iterations, normalization was introduced to solve this instability issue. This was considered a necessity to allow for a sensible analysis of the effects of quantization in a second step. After the evaluation of some candidate factors, a normalization by 0.5 was selected. While a factor of 0.75 would have led to better decoding performance, going from Min-Sum to Normalized Min-Sum as an afterthought is challenging, thus the advantage of much simpler circuitry to implement a division by two was recognized and exploited.

5.1.2. Quantization

After discussing the normalization modification to the Min-Sum decoder, the effects of quantization will be presented next. The base functional model used as a baseline here utilizes an Q4.3 fixed point data format. Thus, each message and each LLR in the system is represented by an 8 bit number, with the exception of the immediate result of certain operations that may overflow. In that case the the overflow is caught, and variables are saturated to their maximum or minimum value before the data is saved in memory. Naturally, the biggest savings in compute resources are achieved with the smallest quantization. In this study, fixed point formats ranging from 3 to 6 bit data width were investigated. It was found that no performance degradation occurs for quantizations down to 5 bit. When examining Figure 5.3, in fact the 'Fixed point (8 bit)' and the 'Fixed point (5 bit)' lines are coincident, as the BER after decoding was identical in the given scenario. This is a very good result and confirms the initial presumption that 8 bit are not required for good decoding performance. Figure 5.3 also shows that neither of the fixed point implementations reaches the performance of a floating point implementation. However the reader should be reminded that this implementation only serves as context and was never intended or attractive to be realized on-chip.

Unfortunately the good performance of 5 bit quantization could not be achieved for a 4 bit scheme. While the literature suggests that this should be possible without significant performance losses, the author could not recreate this in the given time frame. As also shown in Figure 5.3, the decoder once

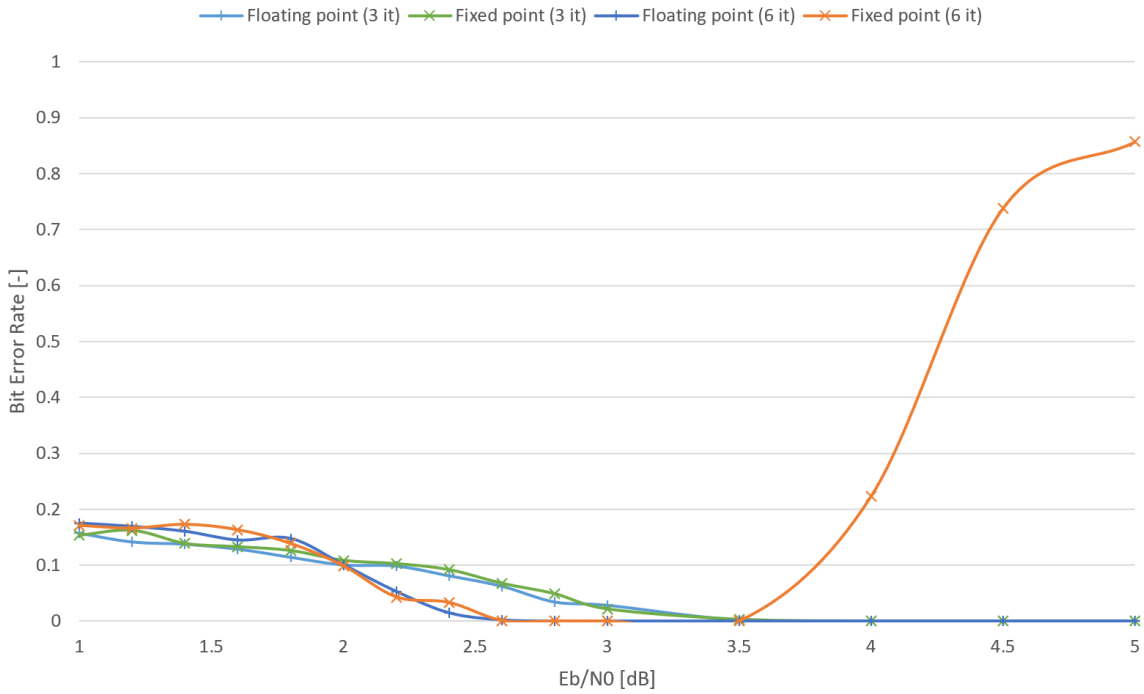


Figure 5.1: Bit error rate as function of signal to noise ratio per bit (E_b/N_0), indicating the decoding performance of default floating and fixed point implementations.

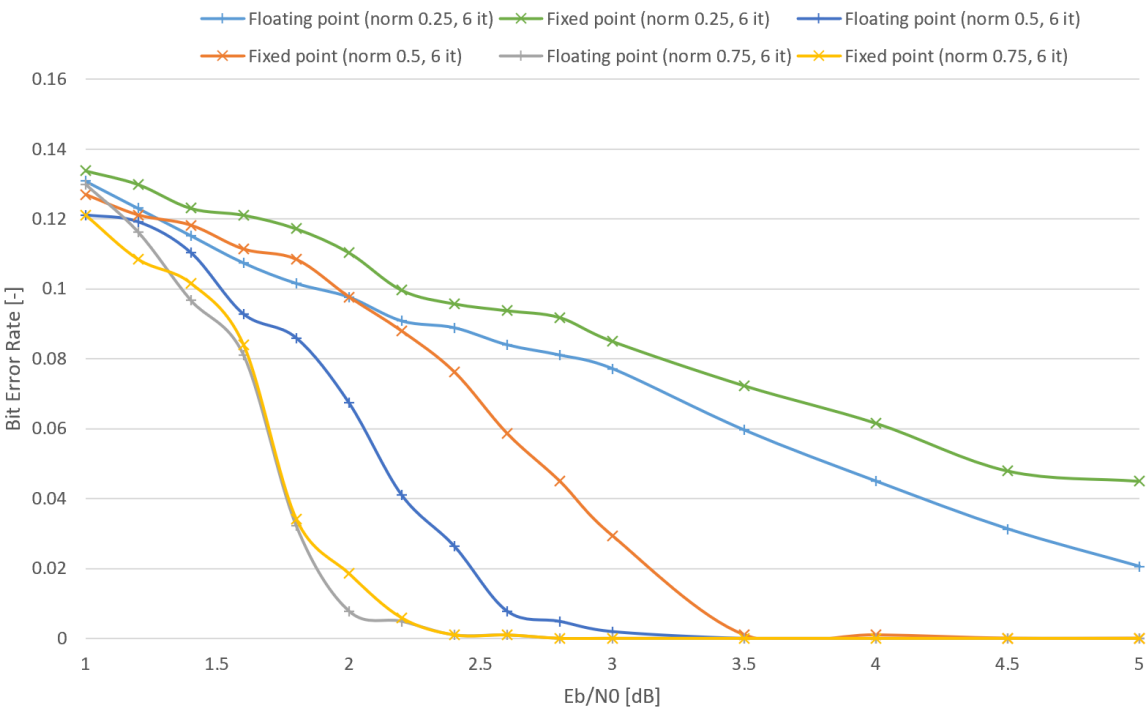


Figure 5.2: Bit error rate as function of signal to noise ratio per bit (E_b/N_0), indicating the decoding performance for a range of normalization factors.

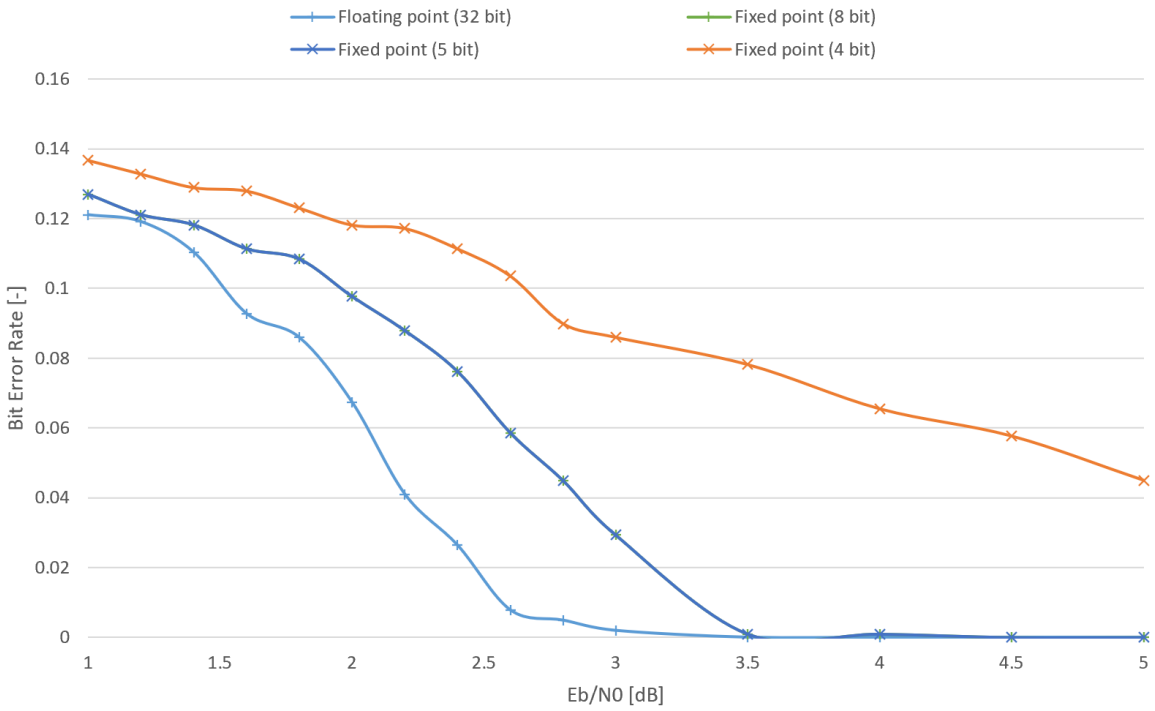


Figure 5.3: Bit error rate as function of signal to noise ratio per bit (E_b/N_0), indicating the decoding performance of the 1/2 normalized algorithm for a range of quantizations.

again fails to decode after six iterations even for a low-noise scenario. Whether the 4 bit implementation's waterfall region is just more shallow, or whether it has not been entered yet, was not investigated further.

5.2. Dependability

Investigations into the dependability attributes of the system and improvements thereof represent the core of this research. A dependable system is a system able to deliver a service that justifiably trusted by its users. Five properties contribute to dependability according to Avižienis et al [14]:

- Availability = Readiness for correct service
- Reliability = Continuity of correct service
- Safety = Absence of catastrophic consequences outside the system boundary
- Integrity = Absence of improper system alterations
- Maintainability = Ability to undergo modifications and repairs

The experimental campaign described in the previous chapter produced a data set capturing radiation effects in the device. From said data set, results can be derived primarily for availability, reliability and safety. The other two properties would need to be analyzed following a different approach.

5.2.1. Radiation effects

Studying radiation effects on the FPGA in general and analyzing the behavior of the baseline, non-radiation-hardened cluster under radiation were the objective of the first three experiments. Those experiments also served to test and validate the experimental setup and the systems developed in that context.

As described in section 4, the entirety of the FPGA is exposed to radiation during testing and will be subject to configuration memory upsets. Since the auxiliary system such as the MicroBlaze soft processor or the DMA engine are not explicitly fault tolerant, an initial exploration of a suitable fluence

was conducted. The goal herein is to find a balance to maximize the data extracted. On the one hand it is necessary to have the decoding cluster fail during irradiation, so that its failure modes can be analysed. On the other hand there is the risk that data can only be recovered partially or not at all if the auxiliary systems fail. Several factors influence this balance, such as the footprint of both parts of the system on the chip or the inherent vulnerability that allows faults to cause failure. But naturally the total fluence also has a strong influence as it is proportional to the total number of upsets, and each upset has a certain probability to cause failure. Thus the first irradiation runs are used to optimize the fluence according to the above goal.

Figure 5.6 shows the number of upsets plotted against the fluence. All data points were collected at a proton energy at target of 120 MeV. Note that the data from the initial exploration was expanded with data from later tests. Three outcomes are distinguished: failure of the test setup (meaning no data recovered), no failure (meaning decoding completed without errors) and cluster failed (meaning the cluster malfunctioned and data was recovered). From the figure it is evident that a higher fluence causes more upsets as is to be expected, however a large variation in upsets for each fluence level can also be observed. The system appears to be able to tolerate up to 5 configuration upsets for this workload and thus 'no failure' occurred mostly for low fluences of less than 2×10^8 protons.cm^{-2}. While this is good to know, the no failure condition is not interesting for future analysis. Regarding the test setup failing, it can be seen that this can happen at any fluence. What is not plotted in the figure is a number of data points with very high fluence of up to 7.27×10^9 protons.cm^{-2}, causing more than 250 upsets. For all these data points the test setup failed before data could be recovered. The desirable outcome is 'cluster failed', which represented the majority of data points. It was found that a fluence of approximately 7×10^8 protons.cm^{-2}, causing some 15 to 35 configuration upsets, represents a good value, where the test setup generally survives and data can be extracted, but the cluster fails in some way.

An important metric when testing FPGAs under radiation is the configuration memory cross section. It represents the ratio of configuration errors to total utilized configuration bits, normalized by the fluence the chip was exposed to as given in Equation 5.1. Correction by fluence and number of bits is crucial to enable comparability across different devices and varying radiation test setups. In section 4.3, the method for extracting the total faults from the device was presented. Since a mask file was used during verification, the total number of bits is not the length of the bitstream but rather only the unmasked bits, represented as zeros in said file [9]. The method to determine the cross section of the entire design is thus straightforward and a value can be calculated for each irradiation. Due to the large number of irradiation runs, mean value and standard deviation are derived. Determining cross section metrics for individual components or subsystems was not possible as it would have required exact localization of the fault and information about the number of configuration bits used for the component only.

$$\theta = \frac{\text{\# errors}}{\text{fluence} \cdot \text{\# bits}} \tag{5.1}$$

Figure 5.4 shows the cross section values as a box and whisker plot. The multiple irradiations conducted at 120 MeV are grouped by experiment. It can be seen that the results are consistent with each other, exhibiting a median value of about 8×10^{-16} cm^2/bit. Larger variation can be observed for the first two experiments, with recorded minima of about 2.5×10^{-16} cm^2/bit and maxima of about 2×10^{-15} cm^2/bit. While the beam settings were varied quite often during the first experiment, the same cannot be said for the second experiment. As the test setup was practically identical no explanation can be given at this point why the variation in cross section is larger.

As mentioned earlier, the energy of the beam was varied during the second experiment. Namely, 27 irradiations were performed at 200 MeV at target as opposed to the usual 120 Mev level used in the other tests. The resulting cross section data is plotted in Figure 5.5. It can be seen that the mean value decreased slightly to 5.34×10^{-16} cm^2/bit while the standard deviation also decreased to 1.52×10^{-16} cm^2/bit. Note that the 120 MeV data used for comparison is from the same experiment only and does not include the data collected on other days.

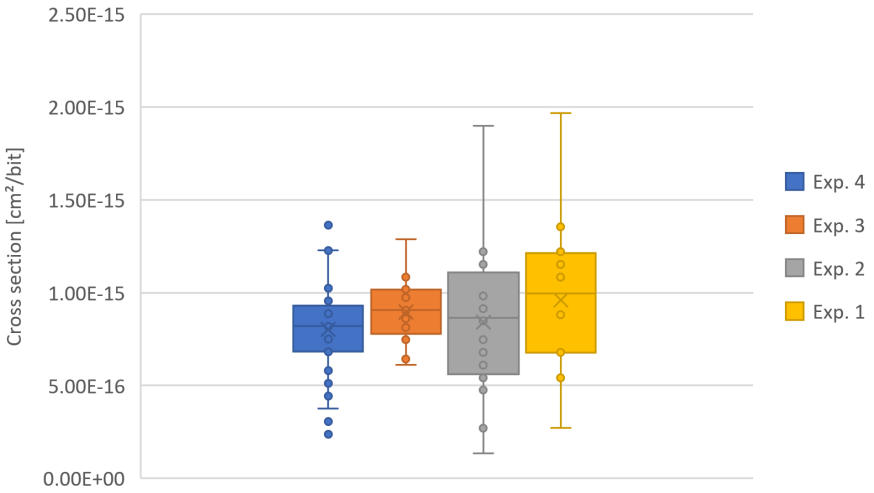


Figure 5.4: Cross section values and spread measured in four experiments at 120 MeV.

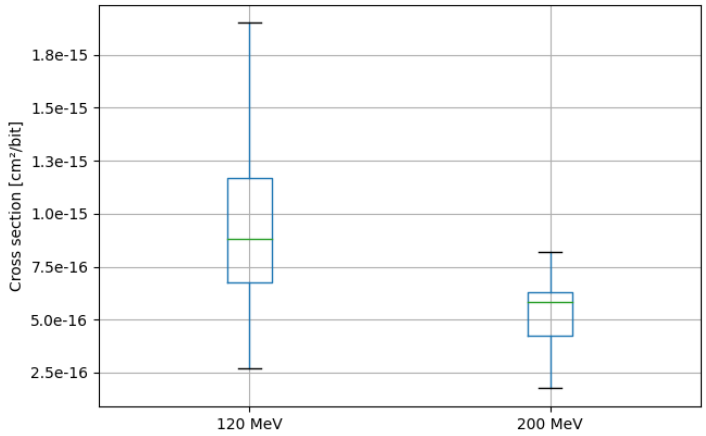


Figure 5.5: Cross section of the design at 120 & 200 MeV.

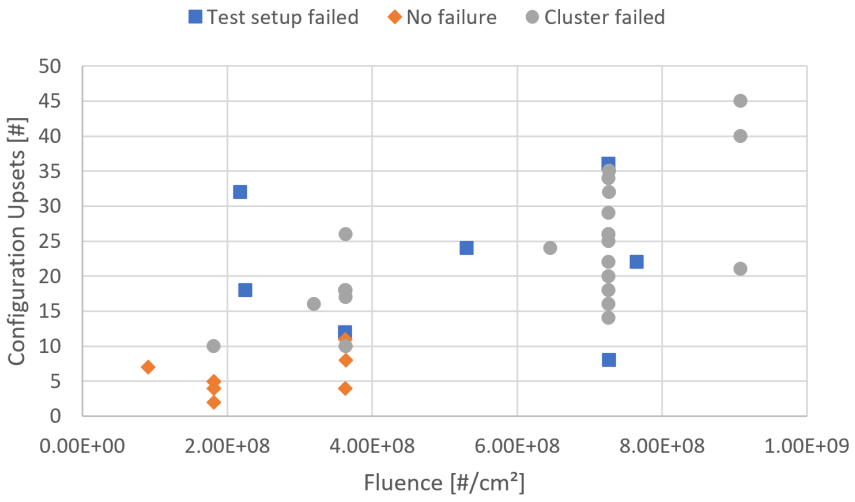


Figure 5.6: Number of upsets versus fluence, categorized by test outcome.

In comparison with other data available in literature, these cross section values are mostly consistent. Hiemstra et al [38] found an overall configuration memory cross-section of 1.89×10^{-15} cm^2/bit for this device. The tests were conduced with proton energy 105 MeV, thus slightly lower, at the TRIUMF facility. Another data point is provided by AMD engineers Maillard et al [50], who subjected the device to a 64 MeV proton beam. The cross section was quantified at 2.5×10^{-15} cm^2/bit in their work. In a more recent publication, a value of 2.16×10^{-15} cm^2/bit was found while testing a specific design also at 64 MeV [51].
Two observations shall be addressed: Firstly, the measured cross section value is slightly smaller than other reported values. This is likely due to variations in either the physical test setup or the method to extract upsets. Namely, it is not know for the other data points whether the chip was irradiated from the top or bottom. It is possible, although unlikely, that the back plate circuit board is providing more shielding than the lid of the device. The number of upsets can be extracted in different ways, in this case the readback and verify function of Vivado was used, while other setups might extract the entire bitstream via the JTAG interface.
Secondly, the cross section decreases with increasing particle energy. This seems counter intuitive, however a similar trend was reported for the Kintex-7 device [56]. More precisely, the cross section decreased from 7.5×10^{-15} cm^2/bit to 6.4×10^{-15} cm^2/bit to 5.4×10^{-15} cm^2/bit when irradiated by 80 MeV, 100 MeV and 184 MeV respectively. Although the decrease is less significant than in this study, this also shows a clear negative correlation between particle energy and device cross section. Note that the Kintex-7 device is based on a 28 nm node while this device is built on 20 nm, thus the actual values cannot be compared.

5.2.2. Base Functional Model

The health signal present in the base functional form was also tested, primarily in the third and fourth experiment. This signal is designed as a heartbeat signal on a clock divider and can be turned on and off via control registers.

As the health signal was not connected in the base functional model, a custom watchdog module was created that would notify the MicroBlaze processor of any irregularity via an interrupt signal. The MicroBlaze will then immediately output some status information via UART. In detail, the watchdog module will trigger quite easily when the signal loses its rhythm or stops beating and does grant a grace period or similar. Unfortunately, due to a software bug, the watchdog did not function as expected in the third experiment. This could not be fixed during the experiment due to the synthesis and implementation processes necessary to create a new bitstream taking several hours. As a fallback option, the signals were observed manually via the Vivado Hardware Manager and the Integrated Logic Analyzer (ILA) IP that was included in the design for debugging purposes. Although far from ideal, this still permitted the author to take random samples of the signals by viewing the waveform momentarily.

Out of the 19 data points available from the third experiment, the health signal was observed to be affected only twice. In 17 cases, the signal did not indicated any issues although the cluster did not function correctly and produced wrong outputs. Note also that the two triggers originated from intense stress testing with high fluence causing 259 and 287 configuration upsets respectively. In the first test even the ILA become unavailable toward the end of the irradiation. Nonetheless, there were also some stress test irradiations that did maintain a good health signal. All 19 runs are summarized in Table 5.1. Due to the watchdog module not working, the exact time the health signal triggered could not be collected.

When looking at the design of this health signal the above results are to be anticipated. It is created in a mostly isolated manner, and only connected to clock, reset and its own control signals and counters. It is not inter-weaved with other logic in the cluster. Therefore, the health signal's generation logic will also be physically separated on the FPGA. Instead of indicating the health of the cluster, it indicates the health of the the signal generator logic. It can of course indicate issues with the clock or reset tree, which arguably cause some of the most severe failures, but such faults will also quickly show in other ways. It is thus concluded that the signal behaves mostly as expected, but falls short of the ambition to inform the other subsystems in the EuFRATE architecture of a clusters availability.

Table 5.1: Baseline health signal performance in the presence of faults.

Run	Energy [MeV]	Fluence [#/m2]	Upsets	Health Signal
1	120	7.27E+09	269	OK
2	120	7.27E+09	267	OK
3	120	7.27E+09	259	KO
4	120	7.27E+09	254	OK
5	120	7.27E+09	287	KO
6	120	7.27E+09	305	OK
7	120	7.27E+08	24	OK
8	120	7.27E+08	22	OK
9	120	7.27E+08	38	OK
10	120	7.27E+08	18	OK
11	120	7.27E+08	32	OK
12	120	7.27E+08	22	OK
13	120	7.27E+08	27	OK
14	120	7.27E+08	23	OK
15	120	7.27E+08	30	OK
16	120	7.27E+08	33	OK
17	120	7.27E+08	19	OK
18	120	7.27E+08	27	OK
19	120	7.27E+08	23	OK

5.2.3. Radiation-hardened

The dependability attributes of the proposed radiation-hardened were the primary subject of the fourth experiment. Since no fault masking or partial reconfiguration mechanisms were included, the analysis is limited to the performance of the new and improved health signal described in detail in section 3.3.2. Once again, the custom watchdog module is employed to monitor the signal.

Table 5.2 shows the irradiation runs for which data could be extracted. For each run, the cycle in which data corruption initially occurred is given in the *Failed at* column and the first cycle in which the health signal was anomalous in the *Watchdog trigger* column. The *Timing* column indicates whether the watchdog trigger was early, late or aligned with the failure of one of the clusters. Lastly the *Aff. Cluster* column tells which cluster was affected. The energy level, fluence and number of upsets are given for completeness.

In summary, 13 cases were recorded where the watchdog triggered although the log file did not indicate any data corruption yet. In 15 cases the trigger was perfectly synchronized with the failure of one of the clusters. In only 4 cases was the watchdog unable to detect failure although data corruption was present in the log. This is considered a good result although there is clearly room for improvement. The most problematic scenario is naturally when the failure is not reflected in the health signal, thus a share of only 4 of 32 runs or 12.5 % is decent. Regarding the early triggers, a straightforward explanation will be given in the next paragraph.

At least part of the early triggers can be related back to the test setup. The mistake here is that after comparison, one of the data streams is discarded. This is in line with the usual DWC approach, where the generation of a health signal is the second data stream's only purpose, however better data would have been acquired had both streams left the FPGA and been reflected in the log. In order to cross-check for discrepancy in the streams and validate the health signals timing, more extensive logging should have been applied. Nonetheless, this does explain the high number of early triggers. Since both redundant clusters take up the same area on chip, and are exposed to the same flux, it is reasonable to assume that their upset rates will be equal in a large enough data set. Thus, the health signal is likely to have indicated a discrepancy correctly, however the faulty data stream was not the one being logged. With a ratio of 13 early triggers to 15 correct triggers, the 1:1 ratio is almost met.

It is left up to the reader to ultimately judge this anomaly, but the author believes that a large number of early triggers are in fact the redundant data stream being faulty. In the best possible scenario, this would lead to a accurate detection rate of 28 out of 32 or 87.5 %.

Once again, it should be pointed out that the lack of observability in the system can lead to distortions in the data. For the scenario of late detection, it is possible that the data corruption was introduced after the comparator modules have processed the data. For example by an upset affecting routing to the DMA engine or its operation. There are several auxiliary systems downstream in the data flow that could corrupt the data in this way and the possibility cannot be excluded based on the collected data.

Although the performance of the new health signal and watchdog could not be quantified as accurately as hoped, it can be safely stated that the improvement over the baseline is significant. Using the duplication with compare approach makes the health signal generation more reliable, as it is interlinked with the data processing task. In a (small) majority of cases, the new signal correctly indicated the failure of a cluster. The fault could also be localized, facilitating a subsequent reconfiguration. In comparison, as described in section 5.2.2, the health signal in the baseline did not appear to indicate issues in any of the irradiation runs. This was confirmed again in the fourth experiment, where the old health signal was also monitored by the watchdog. No trigger could be identified in any of the log files.

With respect to the passive scanning fault detection provided by AMD Soft Error Mitigation (SEM) Controller, a significant improvement in latency was achieved. A typical detection time of 22 ms is claimed for the XCKU040 chip used as testbed in this study, while a detection time of 30 ms is claimed for the targed XCKU060 device [5]. Given that the implemented mechanism detects faults within 1 to 2 ms depending on the number of iterations chosen, the proposed method is at least an order of magnitude faster. More importantly, the proposed mechanism can be used in conjunction with the processed data to flag potentially faulty results. The passive fault detection implemented by SEM can localize and subsequently remove faults, however identifying corrupted data would require additional development to link the two domains.

5.3. Resource Overhead

Whenever a system is modified to improve its dependability attributes, there is an associated cost that the designer needs to pay. Typically the improved version will be larger and more complex and will thus consume more resources. Thus the term overhead is often used to describe the amount of compute resources used over an unprotected baseline taken as reference. Determining overhead is important to classify the effectiveness of certain measures and to enable comparison between different approaches.

In this project, an opportunity was recognized to improve dependability for a relatively small overhead. This is because the baseline design used a high quantization scheme for an algorithm that does not directly benefit of a high quantization. Furthermore some underutilized components were present in the design. By reducing the quantization and fully utilizing said components, it was anticipated that a duplication-with-compare could be implemented for less than the typical overhead of at least 100%. The resource consumption of both the baseline and the radiation-hardened cluster is shown in Table 5.3. The rows represent the various components present on the FPGA, such as lookup-tables (Logic LUT), flip-flops (FF), block RAM (RAMB18 & RAMB36) and digital signal processor blocks (DSP blocks). For completeness the components available but not utilized are also given, namely lookup tables used as memory (LUTRAM), shift register lookup tables (SRL) and ultra RAM (URAM).

The resource consumption, also denoted utilization by AMD, was extracted for a cluster with 16 PFUs. This represents the smallest functional cluster in the radiation-hardened version, which decodes two codewords simultaneously and redundant using 16 PFUs. To enable comparison, the baseline values were also taken for 16 PFUs, thus two clusters of 8 units each. It can be seen that both the Logic LUT and the flip-flop utilization only change slightly by less than 5%. The most significant change can be observed in the block RAM usage, where the baseline uses more RAMB18 primitives while the radiation-hardened version uses more RAMB36 primitives. It should be noted that, as the name suggests, a RAMB36 unit is comprised of two RAMB18 units. Table 5.4 shows the same results in a

Table 5.2: Improved health signal testing.

Run	Energy	Fluence	Upsets	Failed at	Watchdog trigger	Timing	Aff. Cluster
4	120	4.87E+08	27	3073	1284	Early	1B
7	120	4.87E+08	24	358	358	Good	2A
9	120	4.87E+08	22	724	724	Good	1B
10	120	4.87E+08	22	2427	1074	Early	2B
11	120	4.87E+08	16	2844	5400	Late	5B
12	120	4.87E+08	40	2588	2588	Good	5B
16	120	4.87E+08	25	1600	1600	Good	1B
17	120	4.87E+08	17	0	999	Late	6B
21	120	4.87E+08	23	1727	1727	Good	5A
22	120	4.87E+08	31	4768	1995	Early	6B
23	120	4.87E+08	20	4312	1576	Early	6A & 6B
25	120	4.87E+08	28	1087	1087	Good	4A
26	120	4.87E+08	13	9998	4842	Early	2A
27	120	4.87E+08	40	1209	1209	Good	3A
28	120	4.87E+08	15	4469	1861	Early	4A & 4B
29	120	4.87E+08	27	5173	5173	Good	3A
32	120	4.87E+08	25	10025	10025	Good	5B
33	120	4.87E+08	16	1191	1191	Good	5A
34	120	4.87E+08	24	2200	1355	Early	8B
35	120	4.87E+08	11	5800	2487	Early	4B
36	120	4.87E+08	27	2968	2968	Good	2B
37	120	4.87E+08	24	6357	5733	Early	3A
38	120	4.87E+08	15	5977	990	Early	2A
40	120	4.87E+08	20	2066	2066	Good	8A
41	120	4.87E+08	20	10858	7233	Early	2A
42	120	4.87E+08	21	2677	2677	Good	8B
44	120	4.87E+08	25	4494	2847	Early	8A
45	120	4.87E+08	31	1233	1233	Good	1A & 1B
46	120	4.87E+08	24	1664	1857	Late	7B
47	120	4.87E+08	25	1639	2457	Late	7A & 7B
48	120	4.87E+08	31	3472	854	Early	7B
50	120	2.43E+08	18	17179	14961	Early	1B

more concise manner, where this aspect has been considered to have only one block RAM statistic. Furthermore the unused components were removed from the table.

Given the previously discussed improvements on dependability of the radiation hardened, this can be considered a good result. The resource overhead is small for flip-flops at $4.72\,\%$ and the radiation-hardened IP actually uses $2.37\,\%$ less logic LUTs. While it is difficult to correlate these changes to individual design decisions, a few drivers can be identified. The radiation-hardened version uses one 64-bit bus to feed 16 PFUs, while the baseline uses two 32-bit buses to feed two times 8 PFUs. This saves compute resources required to realize and operate the AXI4 interfaces. On the other hand, additional logic had to be implemented to perform the normalization step in the Min-Sum algorithm and to feed 8 values to the DSP block instead of 4. Additionally some logic is required to generate the improved health signals leaving the cluster. This contributes to the rise in flip-flop utilization. It should also be mentioned that the optimization processes that Vivado performs during synthesis and implementation can skew these numbers.

The largest change can be seen in block RAM utilization, where the radiation-hardened version requires the equivalent of 54 RAMB18 blocks over 44 in the baseline, signifying a $22.73\,\%$ increase. As opposed to LUT and FF utilization, the reason could clearly be identified after further investigation. Remembering that each PFU deals with two sets of four values of 5 bit each, the concatenated data width is 40 bit. The RAMB18E2 primitive supports a maximum data width of 36 bit at a depth of 2048 when in simple dual port memory configuration [6]. Thus it is not possible to realize the scratchpad memory using only one of these primitives. During development, Vivado recognizes this and automatically employs a RAMB36E2 primitive instead. As mentioned earlier, this block simply consists of two RAMB18E2 blocks. In simple dual port memory mode, a maximum data width of 72 bit is achievable with a depth of 4096. This leads to the unfortunate situation where a RAMB18 is slightly too small to meet requirements, while the RAMB36 will only utilize 40 bit of the available 72 bit data width.
It is fair to assume that some further optimization could decrease the block RAM overhead. Since the deficit in data width is only four bits, the data of all 16 PFUs present in this implementation could in theory be stored in one RAMB36 block, equivalent to two RAMB18 blocks. This would allow the tightly coupled memory to use the smaller primitive again. Thus, the total RAMB18 utilization would decrease by 8 and increase by 2, leading to a final utilization of 48 blocks. This would decrease the overhead from 22.73 % to just 9.1 %. Note however, that a shared use of memory by several PFUs would require a larger degree of synchronization between the units, or alternatively some additional buffer registers in the PFU.

Overall, it can be concluded that the resource overhead is kept at a reasonable level. LUT and FF utilization only change by a few percent, and could likely be reduced further with another design iteration. The increase in block memory consumption is significant however, and not in line with EuFRATE's tightly-coupled memory approach as of this writing. As shown earlier in this report in Figure 3.3, a one-to-one correspondence exists between an 18k block memory and a DSP block. As such, using two memory blocks per one signal processor is less than ideal and cannot be sustained for larger designs. That being said, the baseline design also appears to be constrained by block RAM memory available on the device.

Table 5.3: Resource utilization and overhead by component.

Component	Baseline	DWC	Relative change
Logic LUTs	18110	17680	-2.37 %
LUTRAMs	0	0	± 0 %
SRLs	0	0	± 0 %
FFs	18104	18958	$+4.72$ %
RAMB36	12	25	$+108.33$ %
RAMB18	20	4	-80 %
URAM	0	0	± 0 %
DSP Blocks	16	16	± 0%

Table 5.4: Utilization and overhead, unused components removed and RAMB36 counted as two RAMB18.

Component	Baseline	DWC	Relative change
Logic LUTs	18110	17680	-2.37 %
FFs	18104	18958	$+4.72$ %
RAMB18	44	54	$+22.73$ %
DSP Blocks	16	16	± 0 %

6

Conclusions

Having presented the results in the decoding performance, dependability and resource overhead domain, conclusions will now be drawn. This shall happen individually but more importantly in synthesis, to judge the success and relevance of this work in anticipation of formulating recommendations.

On the topic of LDPC decoders and their FPGA implementation, it can be concluded that the modifications to the Min-Sum algorithms developed in recent years are not only beneficial but required in this case. Adding a normalization step to the algorithm proved essential for a reliable decoding process over larger signal to noise ratio per bit (E_b/N_0) ranges. It was shown that even a basic divide-by-two can stabilize the algorithm while being relatively cheap to implement in regular FPGA configurable logic. The method and compute resource cost of implementation should be considered from the beginning of any development. While it was possible to add the normalization step as an afterthought, the factor of 1/2 is certainly sub-optimal while a generalized normalization step using a hard block would give the designer more freedom to reach optimal performance.
The quantization of the messages used in the decoder was successfully reduced from 8 bit to 5 bit. This was achieved without any degradation in decoding performance and freed up valuable resources to enable better fault detection. Literature suggest that even a 4 bit quantization is capable of achieving near ideal performance as long as a carefully modified Min-Sum algorithm is used. In this work, the 4 bit quantization did show significant performance losses and was considered out of scope eventually. Using data widths that are not multiples of two did prove challenging during development but was possible, nonetheless it can be concluded that a 4 bit implementation would be significantly simpler to map to hardware components and is achievable.

The dependability of the system has been improved. This has been achieved by adding a better fault detection mechanism based on a duplication with compare methodology. By exploiting the full data width of the digital signal processor hard blocks, one decoding cluster of eight PFUs can decode two codewords simultaneously. Thus, two clusters can be run in information redundancy and any discrepancy when comparing will indicate a fault. It was decided to apply the comparison at cluster level not individual arithmetic operation to be more selective about the failure modes that would be captured, namely to exclude faults in data that do not affect circuits functionality. The resulting health signal was tested and found to indicate faults much more reliably and quickly than the pre-existing health signal. Nonetheless, there were faults that remained undetected even with this improved health signal. Thus it can be concluded that a systematic fault removal through readback of the bitstream is still necessary, to supplement any method optimized for fault detection latency.

Finally, the resource overhead remained within reasonable limits. Lookup-table, flip-flop and digital signal processor utilization is comparable or equal. This was achieved by fully utilizing previously underutilized components and some efficiency gains due to less communication bus infrastructure. On the other hand block RAM utilization went up by more than 20 %, which is problematic. Unfortunately much of this block RAM is underutilized in the proposed design, highlighting again that the hardware

implementation of any algorithm needs to be considered as early as possible to avoid such outcomes. In this specific case, it is fair to say that some more optimization or a different memory architecture would be able to alleviate this increase in memory consumption.

Tying the bridge between the three domains, it was demonstrated that fault tolerance can be improved for a low resource overhead. This is only possible however if there is optimization potential in the algorithm to be executed or its hardware implementation. Lowering the quantization is one strategy to free up resources and decrease the total overhead. This approach is not exclusive to LDPC decoding and can be evaluated for any application in principle.
Due to the ever-increasing demand for more throughput in our digital systems, the underlying algorithms processing data are continuously being improved and the state of the art is dynamic. This is true not only for communication networks, where 5G is being rolled out rapidly and 6G development has started, but also for artificial intelligence. SRAM FPGA technology offers the unique opportunity to react to the newest developments and realize efficiency gains in hardware. As such it can be expected that FPGAs will continue to be an important component in our digital systems, terrestrial or space-based.

6.1. Research questions

In the following, the earlier formulated research question will be answered.

Mitigating soft errors in FPGAs significantly increases resource consumption. How can this overhead be reduced while maintaining fault tolerance attributes?

The process of quantization, mapping mathematical formulations of algorithms to numerical compute resources, can reveal application-specific means of achieving fault tolerance. By considering dependability from an early stage, assumptions on the overhead of typical radiation-hardening methods such as duplication or triplication can be challenged.

Can unused or underused resources in an unmitigated system be exploited to achieve fault tolerance with minimal functional impact on the system?

Depending on the degree of optimization already present, unused or underused resource can be used to implement fault tolerance mechanisms. This can be done with no measurable impact on the performance of the system, as shown in this work, however a small degradation of performance may also be traded against larger gains in fault tolerance. The latter scenario is more generally applicable.

Which means to achieve fault tolerance are applicable for a system exhibiting high degrees of regularity and parrallelization?

The high degree of regularity in this design had no significant impact on the overhead associated to the fault detection mechanism. Nonetheless, a regular system is easier to recover through means of partial reconfiguration, since affected regions can be more easily separated physically and logically. Overall, a higher availability can thus be achieved for systems comprised of regular, interchangeable blocks.

7

Recommendations

In this final chapter, recommendations for future development and research will be formulated. A pragmatic approach is followed in which the recommendations are tied to a specific part of this project or an issue encountered. A small indication is given on the associated effort, where *high* implies a workload of several months or significant coordination required, *medium* implies a workload of several weeks, while *low* represents a recommendation that can be implemented easily for someone familiar with the project or platform.

1. **LDPC Decoder**
   Improving the LDPC decoder was not a key objective of this project, however some observations were made that could benefit this specific decoder.

   (a) It was found that a normalization step can stabilize the decoding performance over a larger range of E_b/N_0. As this normalization was not foreseen in the initial data flow architecture, custom combinatorial circuitry was added. The subset of normalization factors that can be implemented this way cost-effectively is very limited however. Thus, it is recommended to formally include the normalization step in the data flow architecture and exploit the DSP slice that can apply any factor efficiently. The increased latency is preferable over increased configurable logic utilization. A normalization or offset modification to the Min-Sum algorithm is strongly recommended in various publications.
   Effort: High

   (b) Although normalization appears to alleviate the instability issue, the problem might arise from the way the decoder is implemented rather than the underlying algorithm. It is recommended to further investigate this matter as any architectural flaw will otherwise persist in later stages of the EuFRATE project. Since the problem appears to worsen at higher iterations, saturated messages introducing systematic errors and eventually flipping the sign of variable nodes could be one possible explanation.
   Effort: Medium

2. **BRAM Memory Overhead**
   As discussed in the results section, the only compute resource for which the duplication introduces significant overhead is the block RAM memory.

   (a) The most elegant solution to this problem would be to further reduce the quantization to a 4 bit architecture. 4 bit quantized decoders have been demonstrated to work without significant performance losses given some modifications to the Min-Sum algorithm [77]. This change would not only reduce the BRAM overhead, by allowing a single RAMB18 primitive to serve as PFU memory, but also benefit other parts of the system. Since the concurrent development by the consortium uses a 8-bit architecture, communication interfaces are much easier to adapt to handle two 4 bit values instead.
   Effort: Medium

(b) Another approach could be to reduce the data width artificially, by separating sign bit and magnitude to once again reach 4 bit data width for memory elements. Remembering that both sign products and magnitude values are key parts of the Min-Sum algorithm, this solution may also bring some advantages. To store the sign bits several options are available, for example a dedicated BRAM serving multiple PFUs, or just regular localized registers or LUTRAM. Nevertheless, this would require a rework of large parts of the systems data flow architecture and can thus only be recommended in case none of the above is suitable.
Effort: High

3. **Test Setup**
The test setup was able to capture data for a majority of irradiations. There were several cases were the setup failed however and the setup also necessitated some compromises in system architecture, thus improvements would be adequate.

(a) Failures of the test setup could be reduced or entirely avoided if all auxiliary modules on the FPGA are fault tolerant. This primarily concerns the MicroBlaze soft processor, DMA engine and UART controller. Part of this effort is already within scope of EuFRATE, so a better test setup, aimed at extracting data close to the PFU, could become a dedicated milestone of the consortium.
Effort: Medium

(b) An alternative approach would be to try to minimize the number of systems needed on the FPGA and to have the core of the setup off-chip. A critical requirement for this approach is to have a high-throughput interface commissioned. The KCU105 development board offers several candidate interfaces, such as PCIe, Ethernet or FMC (FPGA Mezzanine Card) as shown in Figure 4.5. While those interfaces would still require some controllers on-chip, they would alleviate the bottleneck that the UART link introduces. Subsequently, additional buffer memory outside of the FPGA and a soft processor would not be necessary anymore. Note that the systems receiving data from the FPGA would become more complex, however those systems are universal and could be used for various test campaigns.
Effort: Medium

4. **Others**
Some recommendations can also be given for this specific setup and device under test, mostly stemming from items that were considered out of scope or issues that were only recognized after the last experiment had been conducted already.

(a) Regarding the LDPC decoding benchmarking, a more thorough investigation should be conducted that tests on a larger set of codewords.
Effort: Low

(b) Both output streams should be logged when testing health signal performance.
Effort: Low

(c) Enabling the discrete port MB_Error on the MicroBlaze soft processor could allow for a more detailed classification of failures.
Effort: Low

(d) AMD SEM IP can be configured to allow for real-time insight into configuration upsets during irradiation.
Effort: Medium

References

[1] TS 138 300. *5G; NR; Overall description; Stage-2*. Technical Specification. European Telecommunications Standards Institute, 2018.

[2] EN 302 307. *Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)*. Standard. European Telecommunications Standards Institute, 2009.

[3] AMD. *DS890: UltraScale Architecture and Product Data Sheet Overview*. 2023.

[4] AMD. *PG074: Aurora 64B/66B*. 2023.

[5] AMD. *PG187: UltraScale Architecture Soft Error Mitigation Controller LogiCORE IP Product Guide*. 2023.

[6] AMD. *UG573: UltraScale Architecture Memory Resources User Guide*. 2021.

[7] AMD. *UG579: Ultrascale Architecture DSP Slice User Guide*. 2018.

[8] AMD. *UG761: AXI Reference Guide*. 2012.

[9] AMD. *UG835: Vivado Design Suite Tcl Command Reference Guide*. 2021.

[10] AMD. *UG909: Dynamic Function eXchange User Guide*. 2022.

[11] AMD. *UG917: KCU105 Board User Guide*. 2019.

[12] AMD. *UG974: UltraScale Architecture Libraries Guide*. 2023.

[13] Wolfgang M Arden. *The International Technology Roadmap for Semiconductors - Perspectives and challenges for the next 15 years*. 2002, pp. 371–377.

[14] Algirdas Avižienis et al. “Basic concepts and taxonomy of dependable and secure computing”. In: *IEEE Transactions on Dependable and Secure Computing* 1 (1 Jan. 2004), pp. 11–33. ISSN: 15455971. DOI: 10.1109/TDSC.2004.2.

[15] Mark L. Ayers. “Satellite Networks”. In: *Telecommunications System Reliability Engineering, Theory, and Practice*. 2012, pp. 133–169. DOI: 10.1002/9781118423165.ch4.

[16] Melanie Berg et al. *Xilinx Kintex UltraScale Field Programmable Gate Array Single Event Effects (SEE) Heavy-ion Test Report*. 2018.

[17] Vaughn Betz, Jonathan Rose, and Alexander Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999. ISBN: 978-1-4613-7342-1. DOI: 10.1007/978-1-4615-5145-4.

[18] Safa Bouguezzi, Hassene Faiedh, and Chokri Souani. “Hardware Implementation of Tanh Exponential Activation Function using FPGA”. In: Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 1020–1025. ISBN: 9781665414937. DOI: 10.1109/SSD52085.2021.9429506.

[19] Sébastien Bourdarie and Michael Xapsos. “The near-Earth space radiation environment”. In: vol. 55. Aug. 2008, pp. 1810–1832. DOI: 10.1109/TNS.2008.2001409.

[20] Andrew Boutros, Sadegh Yazdanshenas, and Vaughn Betz. “Embracing Diversity: Enhanced DSP Blocks for Low-Precision Deep Learning on FPGAs”. In: Institute of Electrical and Electronics Engineers Inc., Nov. 2018, pp. 35–42. ISBN: 9781538685174. DOI: 10.1109/FPL.2018.00014.

[21] Ludovica Bozzoli et al. *EuFRATE: Design Description*. 2022.

[22] Ludovica Bozzoli et al. “PyXEL: An Integrated Environment for the Analysis of Fault Effects in SRAM-Based FPGA Routing”. In: *2018 International Symposium on Rapid System Prototyping (RSP)*. 2018, pp. 70–75. DOI: 10.1109/RSP.2018.8632000.

[23] Stephen P. Brown et al. “How Moore’s law is enabling a new generation of telecommunications payloads”. In: American Institute of Aeronautics and Astronautics Inc., 2014. ISBN: 9781624103070. DOI: 10.2514/6.2014-4381.

[24] Jan Budroweit and Hagen Patscheider. “Risk assessment for the use of cots devices in space systems under consideration of radiation effects”. In: *Electronics (Switzerland)* 10 (9 May 2021). ISSN: 20799292. DOI: 10.3390/electronics10091008.

[25] Mark L. Chang. “Device Architecture”. In: *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*. Ed. by Scott Hauck and André DeHon. Morgan Kaufmann, 2008, pp. 3–27. ISBN: 9780123705228.

[26] “Characterization of the HollandPTC proton therapy beamline dedicated to uveal melanoma treatment and an interinstitutional comparison”. In: *Medical Physics* 48 (8 Aug. 2021), pp. 4506–4522. ISSN: 24734209. DOI: 10.1002/mp.15024.

[27] Jinghu Chen and Marc P.C. Fossorier. “Density evolution for two improved BP-based decoding algorithms of LDPC codes”. In: *IEEE Communications Letters* 6 (5 May 2002), pp. 208–210. ISSN: 10897798. DOI: 10.1109/4234.1001666.

[28] Tomas van Dijk. *TU Delta: Meten en regelen in de protonenkliniek*. 2017. URL: https://www.delta.tudelft.nl/article/meten-en-regelen-de-protonenkliniek (visited on 09/16/2022).

[29] P E Dodd and F W Sexton. *Critical Charge Concepts for CMOS SRAMs*. 1995.

[30] Paul E. Dodd and Lloyd W. Massengill. “Basic Mechanisms and Modeling of Single-Event Upset in Digital Microelectronics”. In: *IEEE Transactions on Nuclear Science* 50 (3 2003), pp. 583–602.

[31] ESA. https://connectivity.esa.int/funding/hardening-techniques-commercialoffthesheld-fpga-digital-telecommunication-payloads-artes-5c416. Accessed: 15-11-2023.

[32] ESA. *ECSS-E-ST-10-04C Rev.1*. 2020.

[33] Umer Farooq, Zied Marrakchi, and Habib Mehrez. *FPGA Architectures: An Overview*. Springer New York, 2012, pp. 7–48. DOI: 10.1007/978-1-4614-3594-5_2.

[34] Matthew J. Gadlage et al. “Soft errors induced by high-energy electrons”. In: *IEEE Transactions on Device and Materials Reliability* 17 (1 Mar. 2017), pp. 157–162. ISSN: 15582574. DOI: 10.1109/TDMR.2016.2634626.

[35] Roger Boada Gardenyes. “Trends and patterns in ASIC and FPGA use in space missions and impact in technology roadmaps of the European Space Agency”. 2012.

[36] Cai Grau et al. “Particle therapy in Europe”. In: *Molecular Oncology* 14.7 (2020), pp. 1492–1499. DOI: https://doi.org/10.1002/1878-0261.12677. eprint: https://febs.onlinelibrary.wiley.com/doi/pdf/10.1002/1878-0261.12677. URL: https://febs.onlinelibrary.wiley.com/doi/abs/10.1002/1878-0261.12677.

[37] David M. Hiemstra and Ewart W. Blackmore. “LET Spectra of Proton Energy Levels from 50 to 500 MeV and Their Effectiveness for Single Event Effects Characterization of Microelectronics”. In: vol. 50. Dec. 2003, pp. 2245–2250. DOI: 10.1109/TNS.2003.821811.

[38] David M. Hiemstra, Valeri Kirischian, and Jakub Brelski. “Single Event Upset Characterization of the Kintex UltraScale Field Programmable Gate Array Using Proton Irradiation”. In: 2016.

[39] Atia Ibrahimi. *Characterization of the Proton Beam Line in the Experimental Room of HollandPTC*. 2020. URL: %5Curl%7Bhttps://repository.tudelft.nl/islandora/object/uuid%3A34e200d1-f744-4685-b4d0-2cfa2efce158%7D.

[40] Adam Jacobs, Grzegorz Cieslewski, and Alan D. George. “Overhead and Reliability Analysis of Algorithm-based Fault Tolerance in FPGA Systems”. In: 2012, pp. 300–306. ISBN: 9781467322560. DOI: 10.1109/FPL.2012.6339222.

[41] JEDEC. *JEDEC Dictionary: threshold LET*. [Online; accessed 7-October-2023]. 2023. URL: https://www.jedec.org/standards-documents/dictionary/terms/threshold-let.

[42] Jonathan Johnson et al. *Using Duplication with Compare for On-line Error Detection in FPGA-based Designs*. 2008. DOI: 10.1109/AERO.2008.4526470.

[43] Rinu Jose and Ameenudeen Pe. “Analysis of Hard Decision and Soft Decision Decoding Algorithms of LDPC Codes in AWGN”. In: 2015, pp. 430–435. DOI: 10.1109/IADCC.2015.7154744.

[44] Fernanda Kastensmidt and Paolo Rech. “Radiation effects and fault tolerance techniques for FPGAs and GPUs”. In: *FPGAs and parallel architectures for aerospace applications*. Springer, 2016, pp. 3–17.

[45] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. “Factor graphs and the sum-product algorithm”. In: *IEEE Transactions on information theory* 47.2 (2001), pp. 498–519.

[46] Ian Kuon and Jonathan Rose. “Measuring the gap between FPGAs and ASICs”. In: vol. 26. Feb. 2007, pp. 203–215. DOI: 10.1109/TCAD.2006.884574.

[47] F. Libano et al. “Understanding the Impact of Quantization, Accuracy, and Radiation on the Reliability of Convolutional Neural Networks on FPGAs”. In: *IEEE Transactions on Nuclear Science* 67 (7 July 2020), pp. 1478–1484. ISSN: 15581578. DOI: 10.1109/TNS.2020.2983662.

[48] Andrew Ling, Deshanand P Singh, and Stephen D Brown. “FPGA technology mapping: a study of optimality”. In: *Proceedings of the 42nd annual Design Automation Conference*. 2005, pp. 427–432.

[49] Bozzoli Ludovica et al. “EuFRATE: European FPGA Radiation-hardened Architecture for Telecommunications”. In: 2023. ISBN: 9783981926378.

[50] Pierre Maillard et al. “Neutron, 64 MeV proton, thermal neutron and alpha single-event upset characterization of Xilinx 20nm UltraScale Kintex FPGA”. In: 2015. ISBN: 9781467376419.

[51] Pierre Maillard et al. “Radiation-Tolerant Deep Learning Processor Unit (DPU)-Based Platform Using Xilinx 20-nm Kintex UltraScale FPGA”. In: *IEEE Transactions on Nuclear Science* 70 (4 Apr. 2023), pp. 714–721. ISSN: 15581578. DOI: 10.1109/TNS.2022.3216360.

[52] Keith S. Morgan et al. “A Comparison of TMR with Alternative Fault-Tolerant Design Techniques for FPGAs”. In: vol. 54. Dec. 2007, pp. 2065–2072. DOI: 10.1109/TNS.2007.910871.

[53] Engineering National Academies of Sciences, Medicine, et al. *Testing at the Speed of Light: The State of US Electronic Parts Space Radiation Testing Infrastructure*. National Academies Press, 2018.

[54] Anastasia Pesce and Alessandra Costantino. *Status of High Energy Irradiation Facilities in Europe*. Tech. rep. ESA, 2021.

[55] Brian Pratt et al. “Reduced-Precision Redundancy for Reliable FPGA Communications Systems in High-Radiation Environments”. In: *IEEE Transactions on Aerospace and Electronic Systems* 49 (2013), pp. 369–380. DOI: 10.1109/TAES.2013.6404109.

[56] Markus Preston et al. “Proton-and Neutron-Induced Single-Event Upsets in FPGAs for the PANDA Experiment”. In: *IEEE Transactions on Nuclear Science* 67 (6 June 2020), pp. 1093–1106. ISSN: 15581578. DOI: 10.1109/TNS.2020.2987173.

[57] Heather Quinn et al. “Radiation-induced multi-bit upsets in SRAM-based FPGAs”. In: vol. 52. Dec. 2005, pp. 2455–2461. DOI: 10.1109/TNS.2005.860742.

[58] Heather Marie Quinn et al. *An Introduction to Radiation-Induced Failure Modes and Related Mitigation Methods for Xilinx SRAM FGPAs*. 2008. URL: https://www.researchgate.net/publication/220844215.

[59] Nathaniel Rollins, Megan Fuller, and Michael J. Wirthlin. *A Comparison of Fault-Tolerant Memories in SRAM-Based FPGAs*. IEEE, 2010. ISBN: 9781424438884.

[60] William E. Ryan. *An Introduction to LDPC Codes*. Vol. 5.2. 2004, pp. 1–23.

[61] Laurent Schmalen. *Lecture notes in Channel Coding: Graph-Based Codes*. Sept. 2022.

[62] James R. Schwank, Marty R. Shaneyfelt, and Paul E. Dodd. “Radiation hardness assurance testing of microelectronic devices and integrated circuits: Radiation environments, physical mechanisms, and foundations for hardness assurance”. In: *IEEE Transactions on Nuclear Science* 60 (3 2013), pp. 2074–2100. ISSN: 00189499. DOI: 10.1109/TNS.2013.2254722.

[63] John Shalf. *The future of computing beyond Moore’s Law*. Mar. 2020. DOI: 10.1098/rsta.2019.0061.

[64] Felix Siegle et al. “Mitigation of radiation effects in SRAM-based FPGAs for space applications”. In: *ACM Computing Surveys* 47 (2 Jan. 2015). ISSN: 15577341. DOI: 10.1145/2671181.

[65] E. G. Stassinopoulos and James P. Raymond. “The Space Radiation Environment for Electronics”. In: *Proceedings of the IEEE* 76 (11 1988), pp. 1423–1442. ISSN: 15582256. DOI: 10.1109/5.90113.

[66] R Tanner. “A recursive approach to low complexity codes”. In: *IEEE Transactions on information theory* 27.5 (1981), pp. 533–547.

[67] Stephen M. Trimberger. “Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology”. In: *Proceedings of the IEEE* 103 (3 Mar. 2015), pp. 318–331. ISSN: 15582256. DOI: 10.1109/JPROC.2015.2392104.

[68] Allan J. Tylka et al. *CREME96: A Revision of the Cosmic Ray Effects on Micro-Electronics Code*. 1997. DOI: 10.1109/23.659030. URL: http://crsp3.nrl.navy.mil/creme96/.

[69] Wikipedia contributors. *Q (number format) — Wikipedia, The Free Encyclopedia*. [Online; accessed 3-October-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Q_(number_format)&oldid=1165986062.

[70] Jan J. Wilkens and Uwe Oelfke. “Analytical linear energy transfer calculations for proton therapy”. In: *Medical Physics* 30.5 (2003), pp. 806–815. DOI: https://doi.org/10.1118/1.1567852. eprint: https://aapm.onlinelibrary.wiley.com/doi/pdf/10.1118/1.1567852. URL: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.1567852.

[71] Michael Wirthlin. “High-Reliability FPGA-Based Systems: Space, High-Energy Physics, and Beyond”. In: *Proceedings of the IEEE* 103 (3 Mar. 2015), pp. 379–389. ISSN: 15582256. DOI: 10.1109/JPROC.2015.2404212.

[72] Xilinx. *SDAccel Development Environment Help: DSP48 Block*. 2018. URL: https://www.xilinx.com/htmldocs/xilinx2017_4/sdaccel_doc/uwa1504034294196.html (visited on 09/19/2022).

[73] Xilinx. *Space-Grade Virtex-4QV Family Overview*. 2008. URL: https://docs.xilinx.com/v/u/en-US/ds653.

[74] B Yost et al. “An Overview of the Current State of the Art on Small Spacecraft Avionics Systems”. In: 2022. DOI: 10.2514/6.2022-0521.

[75] In Woo Yun, Hee Ran Lee, and Joon Tae Kim. “An Alternative Approach Obtaining a Normalization Factor in Normalized Min-Sum Algorithm for Low-Density Parity-Check Code”. In: *Wireless Communications and Mobile Computing* 2018 (2018). ISSN: 15308677. DOI: 10.1155/2018/1398191.

[76] Radivoje Zarubica et al. “Efficient Quantization Schemes for LDPC Decoders”. In: I E E E, 2008. ISBN: 9781424426775.

[77] Jianguang Zhao, Farhad Zarkeshvari, and Amir H. Banihashemi. “On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes”. In: *IEEE Transactions on Communications* 53 (4 Apr. 2005), pp. 549–554. ISSN: 00906778. DOI: 10.1109/TCOMM.2004.836563.