Politecnico di Torino

Master's Degree in Aerospace Engineering — Aeromechanics and Systems

A.Y. 2025/2026

Graduation Session December 2025

On the Development of Radiation-Hardened High Performance Computing Nodes for Satellite Systems

Supervisor:

Prof. Luca STERPONE

Ph.D. Corrado DE SIO

Ph.D. Sarah AZIMI

Eng. Giorgio CORA

Candidate:

Morgana DUNI

Abstract

In recent years, there has been a growing demand for High-Performance Spaceflight Computing (HPSC) to support next-generation space missions characterized by increasing autonomy, heterogeneous sensor networks, and the integration of onboard artificial-intelligence algorithms. The aim is to develop autonomous and efficient systems that are able to support remote autonomous operations, onboard data processing for deep space missions. However, adopting such systems in aerospace and avionics requires significant safety and reliability challenges as these safety-critical applications must ensure continuous operation even in the presence of radiation-induced effects that can affect electronic devices and alter system behavior. This work proposes an efficient, reliable, and scalable computing architecture based on a two-node core implemented on ZCU102 FPGA boards connected by a single optical-fiber link. The system is organized around three fundamental components: the processing unit, the communication node, and the optical channel connecting the boards. Each node integrates a 100 MHz NEORV32 soft-core—an open-source, RISC-V International–compliant processor configurable at the microarchitectural level—which is used both to manage inter-board communication and to manage node-level compute resources. Inter-node data movement employs the Aurora protocol over optical fiber to provide high bandwidth, immunity to electromagnetic interference, and stable long-distance signaling while reducing cabling complexity and failure points. The NEORV32–Aurora interface is implemented with AXI4-Stream, an interconnect standard defined by ARM and adopted by AMD/Xilinx for high-speed data transmission between hardware modules. AXI-Stream enables efficient point-to-point dataflow via a ready/valid handshake, making it particularly suitable for FPGA implementations that require continuous transfers, such as optical communication systems or distributed processing architectures. Finally, the optical-fiber channel has been analyzed for efficiency and resilience by evaluating parameters such as data rate, bit-error rate (BER), and link stability. To enhance system resilience, the Aurora module supports partial reconfiguration, allowing rapid recovery from communication faults with minimal downtime. Given its small area footprint, the NEORV32 processor is protected with Triple Modular Redundancy (TMR) and majority voting. The architecture is evaluated through emulation-based fault-injection campaigns that assess link integrity and continuity of operation under adverse conditions. Experimental results show that the system maintains data integrity and service continuity, demonstrating strong fault tolerance and the reliability of the optical interconnect. The proposed platform advances the design and validation of high-performance, fault-tolerant space computing using commercial reconfigurable devices and high-efficiency optical

links. It lays the groundwork for modular multi-node deployments and integration into CubeSat platforms. Future works will scale the system to four or more nodes, integrate it into CubeSat satellites, and adopt improved mitigation techniques to increase system availability and resilience.

Acknowledgements

First, I would like to express my sincere gratitude to my supervisor, Prof. Luca Sterpone, for his guidance, availability, and trust throughout this work. His expertise and scientific rigor have been fundamental pillars during these intense months of research and development.

I am deeply grateful to Giorgio Cora, who has followed me with exceptional dedication and patience, teaching me everything I know up to this point. His support, both technical and personal, has represented a constant reference and has played a crucial role in my growth.

A heartfelt thanks goes to my parents, whose encouragement, sacrifices, and unwavering moral support have accompanied me throughout all the years of study that led to this moment. I am also profoundly thankful to Zia Lidia, a central and influential figure in my life, whose presence and affection have always guided me.

My gratitude extends to my brother Moreno, for being a steady and reassuring presence, even in the most demanding periods.

Finally, I would like to thank Agnese, for her closeness, understanding, and constant encouragement, which have made the most challenging phases of this journey lighter and more manageable.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ADC Analog-to-Digital Converter.

AI Artificial Intelligence.

ALU Arithmetic Logic Unit.

Aurora Aurora 64b/66b High-Speed Serial Protocol.

AXI Advanced eXtensible Interface.

AXI4-Stream AXI4 Streaming Interface.

BER Bit Error Rate.

bps Bits Per Second.

BRAM Block Random Access Memory.

CCSDS Consultative Committee for Space Data Systems.

CDR Clock and Data Recovery.

CMOS Complementary Metal-Oxide-Semiconductor.

CNN Convolutional Neural Network.

COTS Commercial Off-The-Shelf.

CRC Cyclic Redundancy Check.

CSR Control and Status Register.

DAC Digital-to-Analog Converter.

DD Displacement Damage.

DFF D-type Flip-Flop.

DMA Direct Memory Access.

DPR Dynamic Partial Reconfiguration.

DRPM Dynamically Reconfigurable Processing Module.

DSP Digital Signal Processing.

ECC Error Correction Code.

ECSS European Cooperation for Space Standardization.

ESA European Space Agency.

EuFRATE European FPGA Radiation-Hardened Architecture for Telecommunications.

FEC Forward Error Correction.

FIFO First-In First Out.

FMC FPGA Mezzanine Card.

FPGA Field-Programmable Gate Array.

FPU Floating Point Unit.

FSM Finite-State Machine.

GRAS GEANT$_4$ Radiation Analysis for Space.

GTH Gigabit Transceiver High-performance.

GTWIZ Xilinx Transceiver Wizard.

GTX GigaBit Transceiver.

HDL Hardware Description Language.

HIS Hyperspectral Imaging System.

HPC High-Pin Count FMC Connector.

HPSC High Performance Spaceflight Computing.

ILA	Integrated Logic Analyzer.
IP	Intellectual Property Core.
ISA	Instruction Set Architecture.
JPL	Jet Propulsion Laboratory.
JTAG	Joint Test Action Group Interface.
LDPC	Low-Density Parity Check Code.
LET	Linear Energy Transfer.
LUT	Look-Up Table.
MCU	Microcontroller Unit.
MGT	Multi-Gigabit Transceiver.
MMCM	Mixed-Mode Clock Manager.
MPSoC	Multiprocessor System-on-Chip.
MULASSIS	Multi-Layered Shielding Simulation Software.
NASA	National Aeronautics and Space Administration.
NEORV32	Open-Source RISC-V Soft-Core Microcontroller.
NFC	Native Flow Control.
NIEL	Non-Ionizing Energy Loss.
NMR	N-Modular Redundancy.
NoC	Network-on-Chip.
OBC	On-Board Computer.
OBDH	On-Board Data Handling.
PCS	Physical Coding Sublayer.
PL	Programmable Logic.

PLL	Phase-Locked Loop.
PS	Processing System.
QoS	Quality of Service.
RH	Radiation-Hardened.
RISC-V	Reduced Instruction Set Computer - Version Five.
RS	Reed-Solomon Code.
RT	Radiation-Tolerant.
RTL	Register-Transfer Level.
RTOS	Real-Time Operating System.
SCRUB	Configuration Scrubbing.
SEB	Single Event Burnout.
SEE	Single Event Effect.
SEFI	Single Event Functional Interrupt.
SEGR	Single Event Gate Rupture.
SEL	Single Event Latch-Up.
SERDES	Serializer/Deserializer.
SET	Single Event Transient.
SEU	Single Event Upset.
SFP	Small Form-factor Pluggable.
SFP+	Enhanced Small Form-factor Pluggable.
SI570	Programmable Low-Jitter Oscillator.
SoC	System-on-Chip.
SPENVIS	Space Environment Information System.
SRAM	Static Random-Access Memory.

SYSMON System Monitor for On-Chip Voltage and Temperature.

TC Telecommand.

TID Total Ionizing Dose.

TM Telemetry.

TMR Triple Modular Redundancy.

UltraScale+ Xilinx UltraScale+ FPGA Architecture.

VHDL VHSIC Hardware Description Language.

Chapter 1

Introduction

The rapid development of digital electronics over the past decades has radically transformed the concept of data processing, making microprocessors and embedded systems the beating heart of almost every modern device. At the same time, the progressive miniaturization of transistors and the development of increasingly efficient architectures made it possible to achieve processing power once reserved to supercomputers only. This revolution has also reached the space domain, where the growing demand for onboard autonomy and resilience in satellites has led to the concept of High Performance Spaceflight Computing (HPSC).

Next-generation space missions—such as satellites, CubeSats, and planetary-exploration payloads—require high computing capabilities to perform onboard operations such as computer vision, data compression, or neural-network-based decision making. These tasks, traditionally executed by ground systems, must now be performed directly in orbit to reduce communication latency and increase spacecraft autonomy.

However, the space environment imposes severe constraints: limited power availability, extreme thermal conditions, and, above all, the presence of ionizing radiation. These conditions make the direct use of high-performance commercial processors impractical, as they are too vulnerable to radiation-induced effects such as Single Event Effect (SEE), including Single Event Upset (SEU) and Single Event Latch-Up (SEL).

To overcome these limitations, the adoption of Commercial Off-The-Shelf (COTS) devices—particularly Field-Programmable Gate Arrays (FPGAs)—has become increasingly widespread, providing a flexible and efficient alternative. Modern commercial FPGAs, such as those belonging to the Xilinx 7-Series, UltraScale, and UltraScale+ families, offer significantly higher logic density and clock frequencies, combined with mature design environments and a large library of reusable Intellectual Property Cores (IPs). These features enable rapid prototyping and shorter development cycles, which are essential in the context of the *New Space* paradigm, characterized by cost-effective missions and reduced development times.

Compared with general-purpose processors, FPGAs allow designers to more easily mitigate reliability challenges posed by radiation-induced phenomena. While radiation-hardened (Radiation-Hardened (RH)) devices (e.g., the Xilinx Virtex-5QV) are protected at the silicon level, commercial devices rely on architectural and system-level strategies. Among the most common mitigation techniques are Triple Modular Redundancy (TMR), N-Modular Redundancy (NMR), periodic configuration-memory Configuration Scrubbing (SCRUB), and the use of Error Correction Code (ECC) for Block Random Access Memorys (BRAMs). These approaches enable COTS devices to achieve reliability levels comparable to those of space-grade components even in long-duration missions [1, 2].

A notable example of this philosophy is the European FPGA Radiation-Hardened Architecture for Telecommunications (EuFRATE) project, funded by the European Space Agency (ESA), which aims to develop fault-tolerant processing architectures based on clusters of COTS FPGAs for geostationary telecommunication payloads. By combining redundant modules, distributed watchdogs, and dynamic reconfiguration, EuFRATE demonstrates that systems built with commercial devices can ensure operational continuity and resilience even in harsh radiation environments [1].

Alongside reconfigurable architectures, the introduction of the Reduced Instruction Set Computer - Version Five (RISC-V) architecture represents one of the most significant revolutions in embedded and high-performance computing. Originally developed at UC Berkeley and publicly released in 2015, RISC-V was designed as an open and extensible Instruction Set Architecture (ISA) aimed at fostering innovation and hardware independence [3]. Unlike proprietary ISAs such as ARM or x86, RISC-V is entirely open-source, allowing developers and industries to implement and adapt it without licensing restrictions.

The modular structure of the ISA enables designers to tailor implementations by removing unnecessary hardware units for minimal power consumption or integrating custom co-processors for hardware acceleration. Fully synthesizable VHDL implementations of RISC-V are particularly suitable for radiation-prone environments: hardening techniques such as TMR, parity-protected flip-flops, and duplication of control logic can be applied directly at the Register-Transfer Level (RTL) level.

Among existing soft-core processors, the Open-Source RISC-V Soft-Core Microcontroller (NEORV32) stands out as an open-source, configurable microcontroller-class SoC optimized for FPGA integration. Thanks to its lightweight architecture and modular design, NEORV32 can be instantiated multiple times on the same FPGA, enabling redundant control subsystems—an essential capability in fault-tolerant computing nodes for satellite payloads.

1.1 Main Contribution

Some of the main limitations of space-oriented, FPGA-based HPSC systems stem from the need to exchange data among computational units not only quickly and efficiently, but also reliably. This thesis focuses on that challenge and proposes a custom architecture providing fast, reliable, and easy-to-use communication among multiple FPGA nodes.

To this end, a hybrid architecture is adopted: a lightweight RISC-V soft core—the NEORV32—serves as the main processing and control unit, while high-speed communication is achieved through the Aurora 64b/66b High-Speed Serial Protocol (Aurora) protocol, enabling robust optical-fiber links between FPGA devices. Multiple analyses validate the efficiency and reliability of the proposed system, including characterization of the physical channel in terms of Bit Error Rate (BER), throughput, and link stability.

Furthermore, fault-emulation campaigns are performed by injecting SEU-like faults into the FPGA configuration memory to evaluate system robustness against radiation-induced disturbances.

1.2 Thesis Structure

The thesis is organized as follows. Section 2 reviews the state of the art. Section 3 provides background on radiation effects, RISC-V, and the Aurora protocol. Section 4 details the proposed architecture. Section 5 presents the experimental results. Section 6 concludes the thesis and outlines future developments.

Chapter 2

State of the Art

In recent years, the growing interest in space exploration has created a need for reliable and efficient computing architectures. In particular, satellite payloads increasingly rely on on-board data processing to handle the high data rates produced by modern sensors such as hyperspectral images, high-resolution cameras, and Artificial Intelligence Artificial Intelligence (AI)–oriented instruments. For this reason, whereas traditional radiation-hardened (RH) microprocessors can guarantee high reliability, they offer limited performance and energy efficiency. This limitation has led space agencies and industry to adopt heterogeneous platforms that combine CPUs, GPUs, and, in particular, FPGAs. Among these, recently developed architectures, such as the Snapdragon-based co-processor at Jet Propulsion Laboratory (JPL) [4] and the HPSC initiative at National Aeronautics and Space Administration (NASA) [5], show how modern System-on-Chips (SoCs) can be adapted to space applications through redundancy, monitoring, and robust system-level design.

FPGAs have become central to this evolution thanks to their high degree of parallelism, runtime reconfigurability, and ability to host efficient, mission-specific accelerators. As noted in [6], clusters of FPGA nodes enable high-throughput pipelines tailored to demanding workloads such as hyperspectral imaging or radar processing. However, FPGA fabrics, especially those based on Static Random-Access Memory (SRAM) technology, are highly sensitive to radiation. Whereas space-grade devices (e.g., Microchip RT series) offer non-volatile configuration and extensive characterization [7, 8], COTS FPGAs can provide much higher performance but require specific mitigation strategies to be implemented. Comparative studies such as [2] show that COTS devices can reach the reliability levels of rad-hard components only when supported by fault-tolerant design techniques including TMR/NMR, configuration SCRUB, and SEL-protection circuitry.

To meet the performance demands of next-generation missions, several projects explore fault-tolerant and dynamically reconfigurable FPGA clusters. The Dynamically Reconfigurable Processing Module (DRPM) platform [9] demonstrated that Dynamic Partial Reconfiguration (DPR) can restore functionality after SEUs, a result later confirmed by neutron-beam tests [10]. Building on these concepts, the EuFRATE project [1, 11] proposes a scalable architecture where clusters of COTS FPGAs are interconnected through high-speed serial links, with node-level TMR, configuration scrubbing, and cluster-level fault isolation. These efforts collectively show how multi-FPGA clusters are viable for telecom and payload-processing applications. However, they also highlight a major bottleneck: the communication infrastructure between FPGA nodes.

High-speed links are now essential for on-board processing chains. Inside payloads, serial transceivers such as GigaBit Transceiver (GTX)/Gigabit Transceiver High-performance (GTH) and lightweight link protocols such as Aurora 64b/66b are commonly used to interconnect FPGAs, Analog-to-Digital Converters (ADCs)/Digital-to-Analog Converters (DACs), and external transceivers. Implementations such as [12] show that Aurora combined with Enhanced Small Form-factor Pluggable (SFP+) optics supports multi-Gb/s transfers with low latency, making it appropriate for cluster interconnects. Architectures like EuFRATE rely heavily on such links to distribute workloads and maintain throughput under faults. However, these communication paths are themselves exposed to SEUs affecting control logic, First-In First Outs (FIFOs), and link-initialization sequences. As highlighted by application-driven studies [13, 14], achieving end-to-end reliability requires not only robust compute nodes but also streaming interfaces such as AXI4 Streaming Interface (AXI4-Stream), buffering schemes, and link-level protections capable of sustaining sensor-rate data flows without interruption.

Alongside efficient communication strategies, cluster-level coordination typically relies on embedded soft-core processors, particularly when system availability is compromised by radiation-induced effects. Open architectures such as RISC-V have gained traction due to their extensibility, transparency and versatility. Among the various alternatives, the NEORV32, a lightweight RISC-V soft-core [15], offers a customizable set of peripherals and on-chip memories suitable for control, monitoring, and link-management tasks. Radiation-response experiments [16] indicate that, when implemented on flash-based or protected SRAM-based FPGAs, NEORV32 can reliably support supervisory tasks such as watchdog functions, error handling, scrubbing control, and configuration of high-speed interfaces. Its integration therefore provides a flexible software layer to orchestrate data movement across FPGA clusters while maintaining resilience under radiation.

This thesis focuses on addressing link-layer robustness and efficient streaming architectures, proposing an heterogeneous system that integrates an Aurora-based high-speed links with a RISC-V soft core to implement a fast, resilient communication platform for space-oriented FPGA clusters.

Chapter 3

Technical background

3.1 Radiation Effects in Space Electronics

Electronic systems operating in space are exposed to a radiation environment fundamentally different from that encountered on Earth. Depending on the distance from the earth we are considering, the space environment is mainly characterized by trapped electrons and protons in the Van Allen belts, solar particle events, and high-energy galactic cosmic rays. These sources collectively give rise to three major classes of radiation effects: *Total Ionizing Dose* (TID), *Displacement Damage* (DD), and *Single-Event Effects* (SEE). While TID and DD are cumulative, SEEs are stochastic phenomena caused by individual particle strikes and are particularly critical for advanced Complementary Metal-Oxide-Semiconductor (CMOS) and for SRAM-based FPGAs.

3.1.1 Total Ionizing Dose (TID) and Displacement Damage (DD)

Radiation effects in semiconductor devices can be broadly mapped into three fundamental categories: *dose effects*, *dose–rate effects*, and *single–event effects*. To clarify the relationship between these mechanisms, Figure 3.1 illustrates a conceptual Venn diagram showing how cumulative and stochastic phenomena contribute to reliability degradation in space electronics. This figure is an original schematic created for this thesis.

Total Ionizing Dose (TID). TID results from the long–term accumulation of ionizing energy in insulating regions, primarily SiO_2. Electrons generated by ionizing particles quickly escape, whereas holes become trapped either in the oxide or at the $Si–SiO_2$ interface [17]. The progressive buildup of positive charge induces:

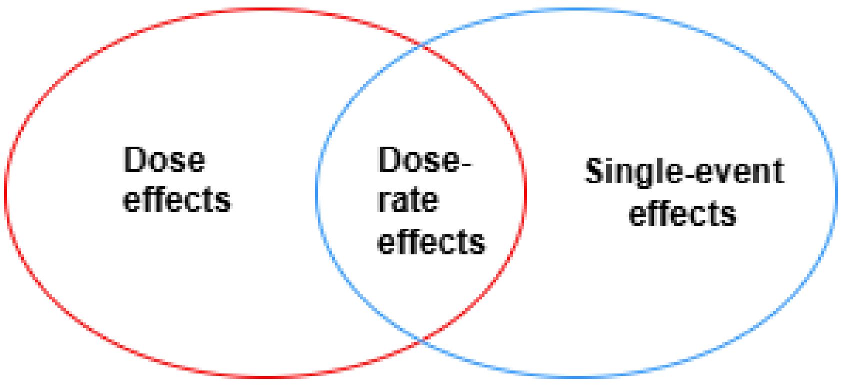


Figure 3.1: Conceptual diagram illustrating the relation among dose effects, dose–rate effects, and single–event effects.

- shifts of MOSFET threshold voltage,
- increase in subthreshold and junction leakage,
- reduced transconductance and drive capability,
- instability in bias networks and analog front-ends,
- timing degradation and possible loss of digital functionality.

Scaling trends partially reduce TID sensitivity thanks to thinner gate oxides, but deep–submicron nodes rely heavily on shallow trench isolation (STI). STI sidewalls accumulate trapped charge that forms parasitic leakage paths, which can compromise logic-cell stability and routing resources—an especially critical issue in dense SRAM–based FPGAs.

Displacement Damage (DD). DD originates from non–ionizing energy loss (Non-Ionizing Energy Loss (NIEL)), mainly due to neutrons and protons displacing silicon atoms from their lattice sites [17]. The resulting defects act as recombination centers and degrade minority–carrier lifetime. This mechanism severely affects:

- bipolar devices (gain degradation),
- optical sensors and photodiodes (dark current increase),
- transimpedance amplifiers and high–speed ADCs front-ends,
- mixed–signal interfaces used in high–throughput payloads.

Although DD plays a minor role in purely digital CMOS logic, it becomes significant for optical transceivers and analog components that support high–bandwidth communication subsystems.

Dose–rate effects. At very high ionizing dose rates (e.g., solar flares or trapped–belt spikes), oxide–trapped charge builds up faster than it can anneal. This imbalance leads to transient bias shifts and performance loss even before the total accumulated dose becomes critical. Such short–term effects are relevant in missions crossing regions of intense proton flux or during solar-particle events.

Together, TID, DD, and dose–rate phenomena define the cumulative component of radiation degradation. Their interplay sets the baseline for device selection and system–level hardening strategies adopted in this thesis.

3.1.2 Single Event Effects in CMOS and SRAM Based FPGAs

Single Event Effects (SEE) are radiation induced phenomena that occur when a heavy ion, proton, or high energy neutron deposits charge within a sensitive region of a semiconductor device. When the collected charge exceeds the critical charge Q_{crit}, the perturbation may alter the state of a memory node, corrupt logic behaviour, or even trigger destructive mechanisms [17].

SEE are commonly grouped into two categories: *non destructive effects*, which temporarily alter circuit behaviour, and *destructive effects*, which may permanently damage the device unless power is removed. An overview of this classification is shown in Figure 3.2.

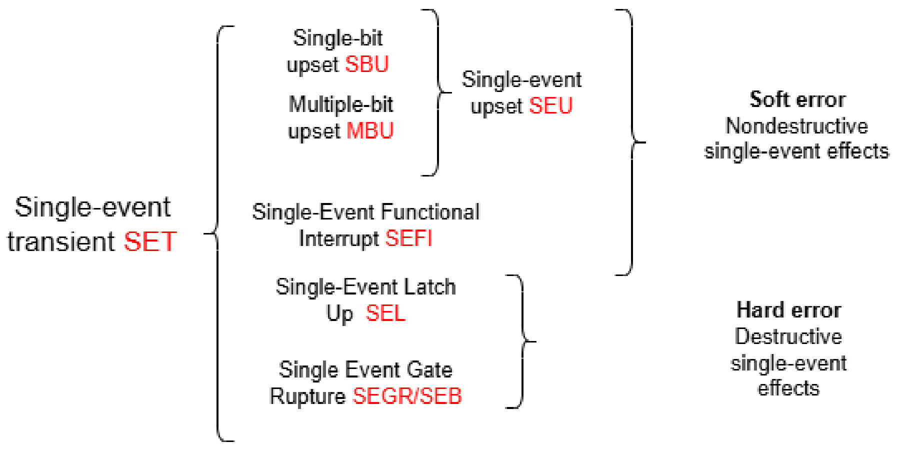

Single-event transient SET
Single-bit upset SBU
Multiple-bit upset MBU
Single-event upset SEU
Single-Event Functional Interrupt SEFI
Single-Event Latch Up SEL
Single Event Gate Rupture SEGR/SEB
Soft error Nondestructive single-event effects
Hard error Destructive single-event effects

Figure 3.2: Classification of non destructive and destructive Single Event Effects.

Non destructive SEE include:

- **Single Event Upset (SEU)**: a bit flip in a latch, flip flop, or memory cell;
- **Single Event Transient (SET)**: a voltage pulse generated along a combinational path, potentially latched at the next clock edge;

- **Single Event Functional Interrupt (SEFI)**: malfunction of an entire functional block such as a PLL, Serializer/Deserializer (SERDES), or configuration controller.

Destructive SEE include:

- **Single Event Latch Up (SEL)**: activation of a parasitic SCR formed by p-well, n-well and substrate structures. Once triggered, it causes a persistent high current state requiring immediate power cycling;
- **Single Event Gate Rupture (SEGR)** and **Single Event Burnout (SEB)**: typically observed in power FETs and RF devices due to oxide rupture or thermal runaway.

SEE are especially critical in SRAM based FPGAs, which contain millions of configuration bits. SEU in configuration frames may alter lookup table contents, routing resources or finite state machine transitions, causing persistent functional corruption until scrubbing rewrites the affected frame. Upsets in user memories (BRAM or distributed RAM) may corrupt data, while SETs can propagate along deep combinational paths and be latched.

Experimental campaigns such as those reported in [2] highlight several trends introduced by deep submicron CMOS scaling:

- reduced sensitive volume decreases SEU cross section in FinFET devices;
- spatial charge sharing increases multi bit upsets in adjacent cells;
- susceptibility of mixed signal blocks to SEFI may increase in advanced nodes;
- configuration memory remains the dominant vulnerability point.

These results confirm that advanced FPGAs still require strong architectural and protocol level mitigation to ensure long term reliability in space missions.

3.1.3 Mitigation Strategies and Architectural Implications

Radiation tolerant FPGA systems rely on multiple complementary mitigation techniques rather than a single universal solution [17]. The most widely used approaches include:

- **Device level selection**: antifuse and flash based FPGAs inherently avoid configuration upsets, whereas commercial SRAM based devices offer higher performance at the cost of additional mitigation;

- **Spatial redundancy**: techniques such as TMR or NMR replicate logic and vote outputs to mask SEU;
- **Configuration scrubbing**: periodic readback and correction of configuration frames prevents long term accumulation of upsets;
- **Temporal filtering**: reduces the impact of SET through resampling or delay based mitigation;
- **Fault detection and recovery**: continuous monitoring of supply current to detect SEL, watchdog timers for SEFI recovery, and reinitialisation of sensitive components such as high-speed transceivers.

In multi FPGA clusters, these techniques must be coordinated by a supervisory controller capable of monitoring error counters, checking link integrity, commanding scrubbing engines, and recovering from protocol or fabric faults. In the architecture proposed in this thesis, this supervisory role is performed by a compact NEORV32 RISC V soft core tightly integrated with the high-speed Aurora serial interconnect.

3.2 RISC-V Architecture

RISC-V is an open, modular, and extensible instruction set architecture designed to support implementations ranging from ultra-low-power microcontrollers to high-performance computing platforms [3]. The ISA is based on a small base instruction set (such as RV32I for 32-bit integer processors), which can be extended using optional standard or custom extensions. The unprivileged specification defines:

- a load store architecture featuring 32 general-purpose registers;
- a fixed-length 32-bit instruction encoding, optionally complemented by a 16-bit compressed format (C extension);
- modular extensions for multiplication/division (M), atomic operations (A), floating-point arithmetic (F/D), bit manipulation, and other application-specific features.

The privileged architecture introduces machine, supervisor, and user privilege modes, as well as a comprehensive system of Control and Status Registers (CSRs) for interrupt handling, exception management, and memory protection. This separation between ISA and microarchitecture is particularly useful in radiation-prone applications: fault tolerance techniques such as hardened flip-flops, logic duplication, or full TMR can be applied to the implementation without requiring changes to the software-visible ISA [3].

3.2.1 The NEORV32 Soft-Core

The RISC-V architecture is an open and modular instruction set designed to provide a clean and extensible foundation for embedded and high-performance systems. Its base ISA (such as RV32I) consists of a compact load–store design with 32 general-purpose registers, a fixed 32-bit instruction format, and a well-defined set of optional extensions for integer multiplication (M), atomics (A), floating point (F/D), compressed instructions (C), and other domain-specific accelerators [3]. A key advantage of the architecture is the strict separation between ISA and microarchitecture: radiation-hardening techniques such as TMR on pipeline registers, hardened flip-flops, and duplicated control logic can be applied at RTL level without altering the ISA itself. As a consequence, RISC-V is particularly attractive for space applications, where predictable behavior, portability, and long-term maintainability are critical.

Among the available FPGA-oriented implementations, a representative example is the NEORV32 soft-core processor, an open-source microcontroller-class system specifically designed for resource-constrained FPGAs [15, 18]. The NEORV32 integrates a compact RV32I-compatible CPU, optional ISA extensions, on-chip instruction and data memories (IMEM and DMEM), configurable caches, and a rich set of peripherals including UARTs, SPI, timers, GPIO, a watchdog timer, and a streamlined Direct Memory Access (DMA) engine. All components are connected through a unified, memory-mapped interconnect (XBUS), allowing software to configure and monitor peripherals using standard load/store instructions. Because every peripheral and CPU extension can be individually enabled or disabled via VHDL generics, the synthesized system can be tailored to the available FPGA resources—an important consideration in radiation-tolerant design, where minimizing area helps reduce both power consumption and the likelihood of radiation-induced faults.

A simplified overview of the processor setup typically used for system bring-up and testing is shown in Figure 3.3. The diagram illustrates the essential components (CPU, IMEM/DMEM, reset and clocking network, UART for console interaction, and basic GPIO), providing a high-level representation of the NEORV32 environment relevant to supervisory control tasks.

One of the most relevant aspects of the NEORV32 for this thesis is its interrupt organization. The processor implements the standard machine-level interrupt model defined by the RISC-V privileged specification [3], complemented by sixteen dedicated *fast interrupt request* channels (FIRQ0–FIRQ15), each associated with a specific on-chip peripheral [18]. This organization allows fine-grained and low-latency interrupt handling, which is essential when supervising high-speed communication interfaces such as Aurora links or when reacting quickly to protocol anomalies, buffer transitions, and event-driven peripheral activity.

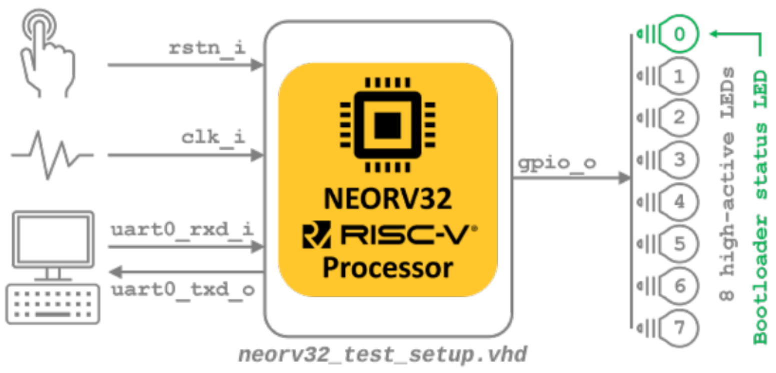


Figure 3.3: NEORV32 test configuration used for basic "Hello World" bring-up, highlighting CPU, memory blocks, UART interface, and reset/clocking infrastructure. Adapted from the NEORV32 User Guide [15].

The NEORV32 Runtime Environment (RTE) provides default handlers for traps, interrupts, and exceptions, which can be selectively overridden by the application layer [15]. This simplifies the development of fault-handling routines capable of logging errors, resetting peripherals, performing link reinitialization, or recovering from Single Event Functional Interrupts (SEFIs). Optional debugging features, such as an on-chip debugger compliant with the RISC-V debug specification, facilitate development and testing, while the integrated watchdog timer provides continuous health monitoring during operation.

Experimental neutron-irradiation campaigns confirm that NEORV32 implementations on flash-based FPGA fabrics maintain stable behavior under particle fluxes representative of space environments when supported by architectural protection mechanisms such as configuration scrubbing, selective redundancy, and parity or ECC-protected memories [16]. These observations reinforce the suitability of the core as a supervisory controller in radiation-exposed platforms.

Within the architecture developed in this thesis, the NEORV32 plays the role of a centralized supervisor responsible for configuring the high-speed Aurora links, initializing data-path components, monitoring link integrity and error counters, and orchestrating recovery procedures when protocol or fabric anomalies occur. Its lightweight nature, deterministic behavior, and flexible peripheral set make it well aligned with the requirements of multi-FPGA spaceborne computing nodes.

3.3 Aurora 64b/66b High-Speed Serial Protocol

The Aurora 64b/66b protocol is a lightweight, point-to-point communication standard developed by AMD/Xilinx to exploit the high-speed transceivers (GTX, GTH, GTY) available in modern FPGAs. Unlike complete networking stacks such as Ethernet or PCIe, Aurora is designed to provide minimal overhead, extremely low latency, and deterministic behavior in high-throughput serial links. The official specification SP011 [19] and the product guide PG074 [20] define the architecture, initialization sequence, and user interface of the protocol.

An Aurora channel consists of one or more bonded lanes that together form a logical serial link. User data are injected through an AXI4-Stream interface, encoded into 66-bit blocks, scrambled, then serialized and transmitted by the high-speed transceivers. On reception, the transceiver performs clock and data recovery while the Aurora core reconstructs the 66-bit blocks, performs lane deskewing, verifies block alignment, decodes the 64b/66b format, and outputs AXI4-Stream words to the user logic. This streaming interface enables seamless integration with DMA engines, packet routers, and high-rate processing pipelines.

A simplified, original schematic of the Aurora transmit–receive datapath is shown in Figure 3.4, highlighting the encoder, lane logic, Physical Coding Sublayer (PCS) components, and AXI-Stream endpoints.

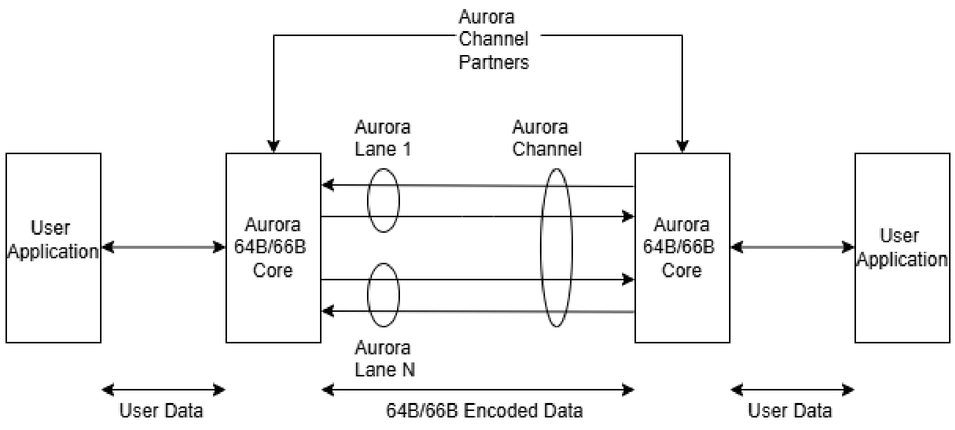


Figure 3.4: Schematic of a simplified Aurora TX/RX datapath.

Aurora uses the same 64b/66b line encoding adopted in high-speed Ethernet. Each block consists of a 2-bit *sync header* identifying the type of block (`01` for data, `10` for control) and a 64-bit payload field [19]. The resulting overhead is only about 3%, significantly lower than the 25% overhead of 8b/10b coding, making Aurora well suited for inter-FPGA streaming at multi-gigabit rates. In multi-lane configurations, data blocks are striped across lanes and reassembled at the receiver.

The logic in the Aurora controller automatically handles channel bonding, deskew, and word alignment to maintain a continuous and correctly ordered data stream [20]. Figure 3.5 illustrates the structure of a 66-bit encoded block.

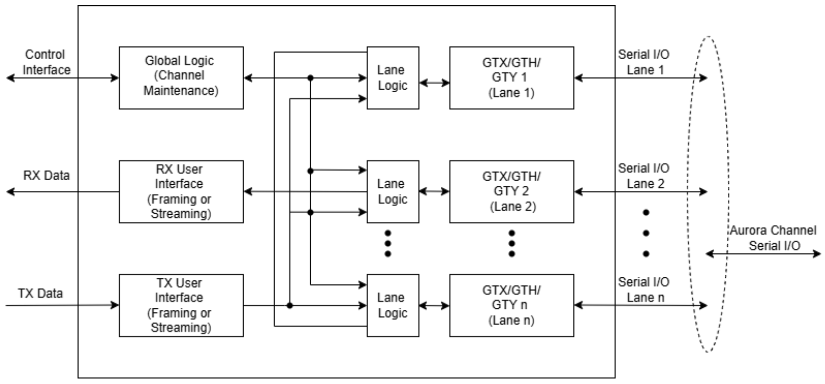


Figure 3.5: Structure of a 66-bit Aurora block (custom schematic).

Before data transmission, the Aurora controller executes an initialization sequence to configure transceivers, synchronize lanes, and validate remote-link readiness. The procedure includes transceiver reset and Phase-Locked Loop (PLL) locking, block synchronization within each lane, channel bonding, remote-end verification, and finally assertion of `lane_up` and `channel_up` signals when the link becomes operational [20]. The high-speed serial path and the user clock domain may drift relative to one another; therefore, Aurora employs *clock compensation* by inserting or removing special control blocks, preventing FIFO overflows or underflows within the PCS and improving resilience under transient disturbances.

Error handling is a central element of the protocol. Aurora detects invalid sync headers, alignment violations, and lane deskew errors, and it continuously monitors transceiver health indicators such as loss-of-lock or CDR faults. Optionally, a 32-bit Cyclic Redundancy Check (CRC) can be applied to user payloads. Minor faults generally trigger local resynchronization, whereas persistent anomalies assert `hard_err`, forcing the link to reset and reinitialize [20]. The protocol also provides several flow-control mechanisms, including Native Flow Control (NFC), User Flow Control (UFC), and USER-K blocks, which allow throttling, synchronization, or transmission of metadata over the same serial link.

In radiation-tolerant FPGA clusters such as those presented in [1, 11], Aurora forms the primary high-throughput interconnect used to share data among computing nodes. However, its control logic, PCS buffers, and clock recovery units are sensitive to radiation-induced effects such as SEUs, Single Event Transients (SETs), and SEFIs. For this reason, the supervisory processor—in this work, the NEORV32 soft-core—continuously monitors `lane_up` and `channel_up` signals, error counters, transceiver lock indicators, and CRC results. Persistent faults trigger link reinitialization, lane reconfiguration, or higher-level recovery procedures. Thus, Aurora serves a dual role: it provides the essential high-rate data path between FPGA nodes, while simultaneously requiring active supervision and fault-management strategies to guarantee resilience in radiation-exposed operational environments.

Chapter 4

Proposed Architecture

Modern payload and on-board processing systems rely extensively on high-throughput serial communication channels implemented on commercial FPGAs, where high-speed transceivers and lightweight link-layer protocols such as Aurora enable multi-gigabit data exchange between computational nodes. However, operating these devices in radiation environments introduces the possibility of single-event–induced corruptions that affect transceiver logic, protocol controllers, and supervisory processors.

The architecture presented in this chapter has been designed to provide resilience against such effects by combining:

- a deterministic low-latency data path based on Aurora 64b/66b;
- a supervisory control plane based on triplicated NEORV32 soft cores;
- a robust reset and monitoring subsystem;
- an AXI4-Stream infrastructure enabling scalable and verifiable data movement.

The entire system has been implemented on the **ZCU102** evaluation board, whose transceiver resources, differential clocking network, and FPGA fabric characteristics make it particularly suitable for high-speed serial experiments and fault-tolerant prototyping [21].

Figure 4.1 illustrates the final block design synthesized in Vivado. The design includes a complete Aurora 64b/66b transceiver channel, three NEORV32 soft cores operating in a triplicated configuration, a TMR voter operating directly on the AXI4-Stream datapath produced by the processors, clock and reset controllers, and the AXI4-Stream multiplexing blocks necessary to interface the supervisory logic with the Aurora subsystem [20, 22, 23, 24].

The architecture is organized around three complementary planes:

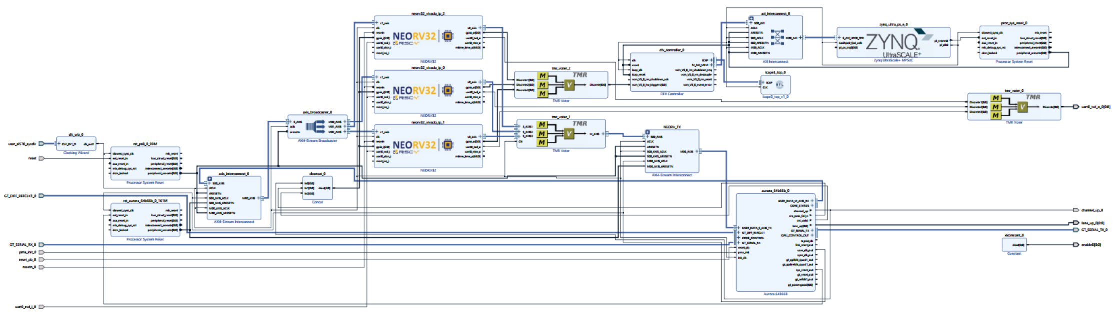


Figure 4.1: Final block design.

- a **high-speed datapath** implemented through the Aurora PCS/PMD and AXI4-Stream user interface, responsible for transporting 64-bit payloads at multi-gigabit rate;
- a **supervisory control plane** based on triplicated NEORV32 processors, which configure the Aurora core, read status registers, verify link health, manage resets, generate test traffic and request error correction through partial reconfiguration via DFX Controller.
- an **FPGA-based error correction logic** based on the DFX Controller, which enables fast and efficient partial reconfiguration of the Aurora core in case its availability and operativity are compromised due to upsets in the configuration memory.

4.1 Clocking and Reset Infrastructure

The ZCU102 provides a suitable clocking scheme using the SI570 programmable oscillator and the GTH reference pins [21]. The Aurora core uses its own recovered clock for the user interface, while the NEORV32 processors and AXI4-Stream components operate on a Programmable Logic (PL) clock generated using the Clocking Wizard IP [25].

The reset distribution follows the structure of the PG164 `proc_sys_reset` [23], ensuring deterministic start and controlled resynchronisation. The Aurora subsystem asserts a dedicated `sys_reset_out` signal when its internal Finite-State Machine (FSM) detects a fault, which in turn resets all NEORV32 instances without disrupting transceiver stability.

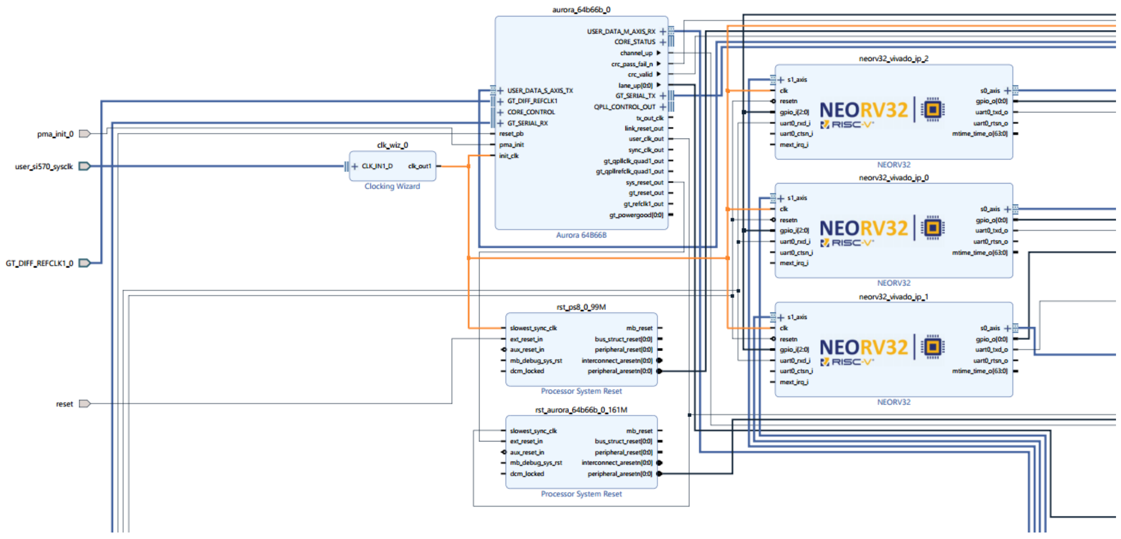


Figure 4.2: Clocking and reset

4.2 Aurora IP

The high–speed communication backbone of the proposed architecture is based on the Aurora 64b/66b core, instantiated and configured according to the options available in the official Product Guide PG074 [20] and the protocol rules defined in the *Aurora 64B/66B Protocol Specification* (SP011) [19]. In this work, the Aurora core is implemented as a single–lane GTH transceiver operating at a line rate of 10.312,5 Gbps, matching the configuration capabilities of the ZCU102 evaluation platform.

The core uses the **GTH** transceiver type, selected through the Physical Layer options of PG074, and is clocked through a 156.25 MHz reference clock generated from the programmable oscillator SI570 onboard. The Quad transceiver is configured as X1Y1, lane X1Y4, and internally uses the MGTFREFCLK0 reference source. A 100 MHz `init_clk` is supplied to drive the core initialization and reset sequences.

The Aurora link is configured in **Duplex** mode, with the **Framing** interface enabled and no link–level flow control, as shown in the configuration summaries of Fig. 4.3. Since the design implements a dedicated single FPGA data path and does not involve multicore Aurora clusters, the option "Include Shared Logic in the" core" has been selected. According to PG074, placing the Shared Logic inside the core allows encapsulating:

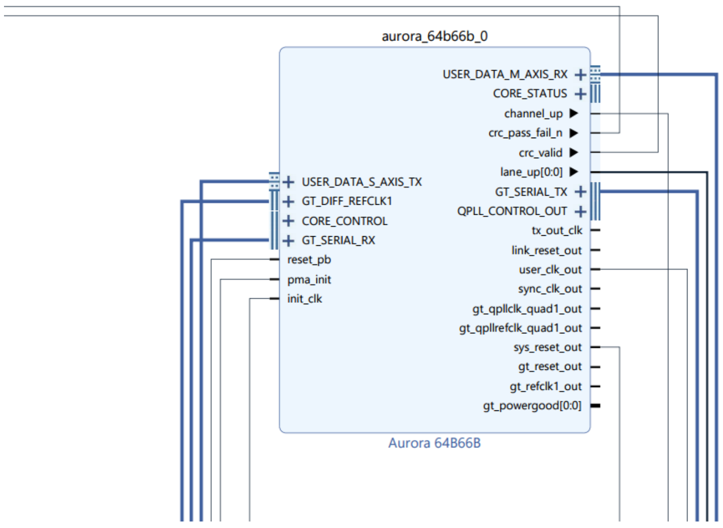


Figure 4.3: Aurora 64B/66B core instantiated in the design.

- the Quad PLL and reference–clock buffers,
- the reset FSM handling `init_clk` and transceiver bring–up,
- the differential GT reference–clock network,
- lane alignment and clock compensation logic.

This option produces a self–contained instance that does not depend on additional external Aurora masters, reducing integration complexity and avoiding the multi–IP Shared Logic partitioning that is typical in multi–FPGA Aurora clusters or master–slave Aurora configurations. This choice also simplifies future scalability: should a multi–FPGA Aurora fabric be implemented, Shared Logic may be externalized to allow multiple Aurora endpoints to share the same GT clocking domain; in the present design, this is unnecessary and would only increase system complexity.

4.2.1 AXI4-Stream Integration

A key feature of the Aurora core is the AXI4-Stream-based streaming user interface. The instantiated Aurora module exposes the following:

- an `S_AXIS_TX` slave interface for transmitting 64-bit payloads toward the GT,

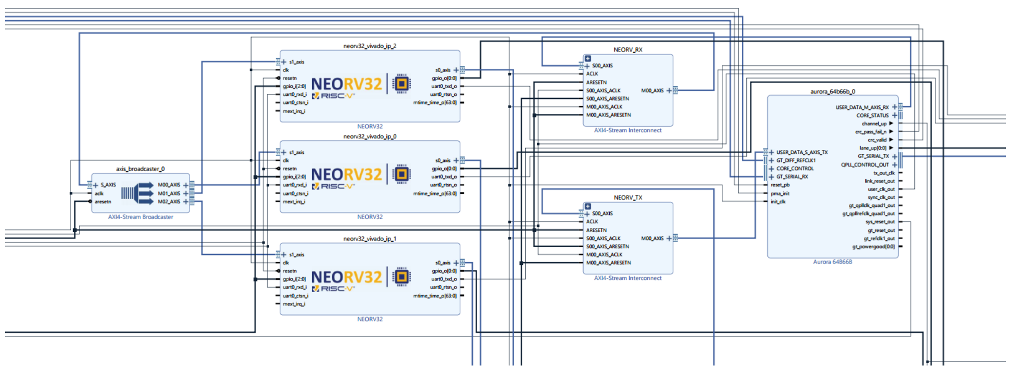


Figure 4.4: AXI4-Stream Interconnect and Broadcaster connected to the Aurora RX/TX interfaces.

- an M_AXIS_RX master interface delivering the received words to the upstream logic.

Two AXI4-Stream infrastructure blocks (PG085 [22]) are instantiated:

1. **AXIS Interconnect (RX path).** It adapts the 64-bit Aurora RX stream to the 32-bit internal width required by the supervisory processors. This block performs the clock domain crossing between the Aurora user_clk_out and the internal PL clock, guaranteeing the correct alignment of tdata, tkeep and tlast under all backpressure conditions.

2. **AXIS Broadcaster (RX fan-out).** Since the NEORV32 subsystem is in triplicate, the single Aurora RX stream must be consistently delivered to all three supervisory nodes. A simple fan-out cannot be used, because AXI4-Stream requires strict handshake semantics; therefore, the **Broadcaster** replicates the stream while ensuring identical word ordering and synchronized valid/ready behavior toward all sinks.

On the transmit side, the voted AXI4-Stream produced by the TMR voter is routed through an additional AXIS Interconnect that converts the 32-bit supervisory width into the 64-bit payload required by the Aurora IP, completing the round-trip datapath.

4.2.2 Clocking and Reset Integration

The Aurora core generates its own recovered user_clk_out for all the datapath logic of the transmitter and receiver. The reset distribution follows the guidelines of PG164 (proc_sys_reset) [23]. The core asserts `sys_reset_out` when its internal FSM detects a fatal GT or protocol fault, which automatically resets the downstream AXI logic and the supervisory control subsystem. Additionally, control signals such as `lane_up` and `channel_up` propagate to the NEORV32 domain through GPIO and concatenation logic to enable system-wide fault recognition.

4.2.3 CRC Mechanism in Aurora 64B/66B

Aurora performs CRC generation and verification at the end of each transmitted frame, as specified in the Aurora 64B/66B Product Guide (PG074, Sec. 3.2.4) [20]. During the final cycle of a frame, the core asserts the `crc_valid` signal and evaluates the correctness of the received payload using the `crc_pass_fail_n` output. CRC is computed per lane and integrated into the internal PCS logic, ensuring deterministic error detection in multi-lane configurations [20].

Polynomial CRC. According to the *Aurora 64B/66B Protocol Specification* (SP011) [19], the protocol employs the standard CRC-32 polynomial used in 64b/66b Ethernet encoding:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1,$$

which corresponds to the hexadecimal representation `0x04C11DB7`. To ensure complete compatibility with Aurora's internal PCS pipeline, the system adopts a bitwise LFSR-style update rule that mirrors the hardware computation, guaranteeing CRC values that match between the PCS logic and the external supervisory domain.

Supervision and Fault Handling. During operation, the architecture continuously monitors the health of the Aurora link through status signals such as `lane_up`, `channel_up`, `soft_err`, and `hard_err`, made available by the core as described in the status interface PG074 [20]. Persistent CRC mismatches, repeated soft errors, or loss-of-sync conditions can trigger a controlled reinitialization of the link or a partial reconfiguration request issued to the fault–tolerant control subsystem, ensuring robust operation in radiation-prone environments.

The integration of this deterministic CRC pipeline with triplicated supervisory logic, AXI4-Stream infrastructure, and TMR votes ensures that the high-speed link remains resilient even in the presence of transient radiation-induced faults.

4.3 NEORV32 Instance

The NEORV32 is in charge of managing the entire architecture, managing data exchange through Aurora, validating sent and received data, and requesting partial

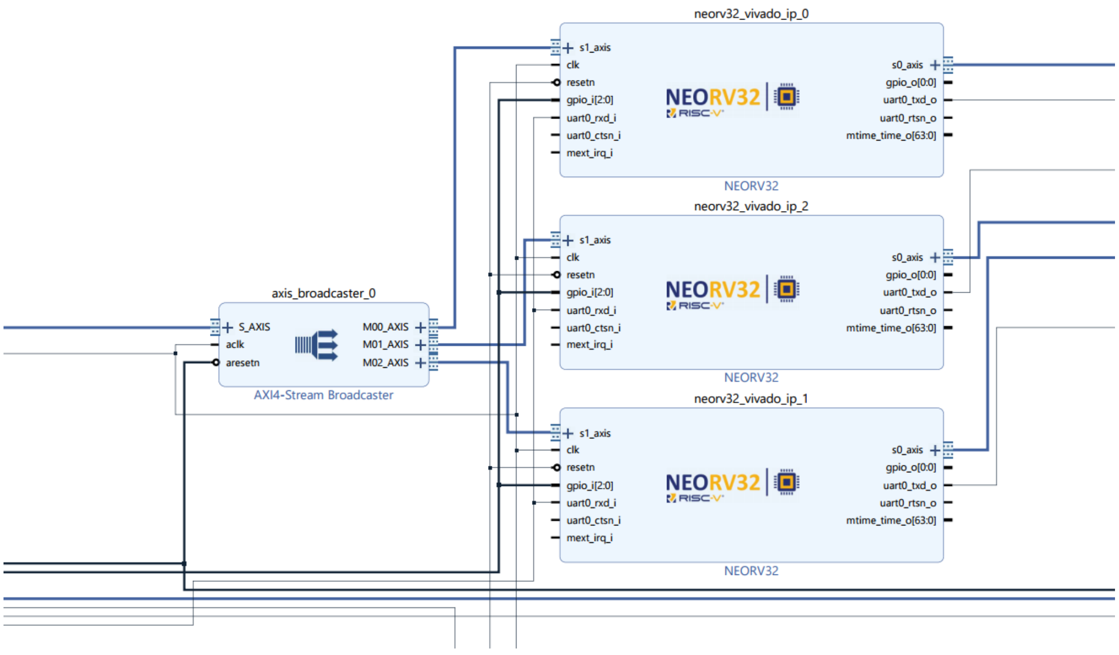


Figure 4.5: Triplicated NEORV32 processors connected through an AXI4-Stream Broadcaster.

reconfiguration in case of errors. To perform these operations, the base RISC-V ISA has been adopted. UART communication, to perform debugging and communication with a computer Host, was enabled and configured to 19200 baudrate, 8 data bits, no parity and one stop bit. Two GPIOs are used to communicate with the DFX Controller and receive information regarding the status of the physical transceiver resource. Additionally, the AXI4-Stream interface was enabled to support fast and efficient data exchange with the Aurora module. To reduced the probability of failure due to SEE affecting the configuration memory of the ZCU102, TripleModular Redundancy has been applied to the NEORV32. This architectural choice required adapting the AXI4-Stream data flow: the Aurora 64b/66b receiver exposes a single `M_AXIS` interface, which cannot directly feed multiple processors due to the AXI4-Stream protocol limitations. To address this, the **AXI4-Stream Broadcaster** [22] was inserted to replicate the incoming stream toward all three NEORV32 nodes. Unlike a simple fan-out, the broadcaster preserves the full AXI4-Stream handshake semantics, ensuring that every processor receives an identical and coherently ordered sequence of payload words, even in the presence of backpressure or temporary stalls on any of the parallel paths.

On the transmit side, each NEORV32 produces its own independent `S_AXIS` stream. However, in this stage of development, the three processors may diverge in timing due to interrupt handling, local stalls, or variable AXI4-Stream backpressure. This natural desynchronization motivated the introduction of majority voting stages in the subsequent architecture, which ensure cross-checking and consistency of the transmitted data across the triplicated supervisory domain.

4.3.1 TMR Voters on Control and Datapath

The implementation of TMR requires majority voters to be adopted among the outputs of all the replicated instances of the processor. For this purpose, two TMRs mechanisms based on the PG268 guidelines [24] were used:

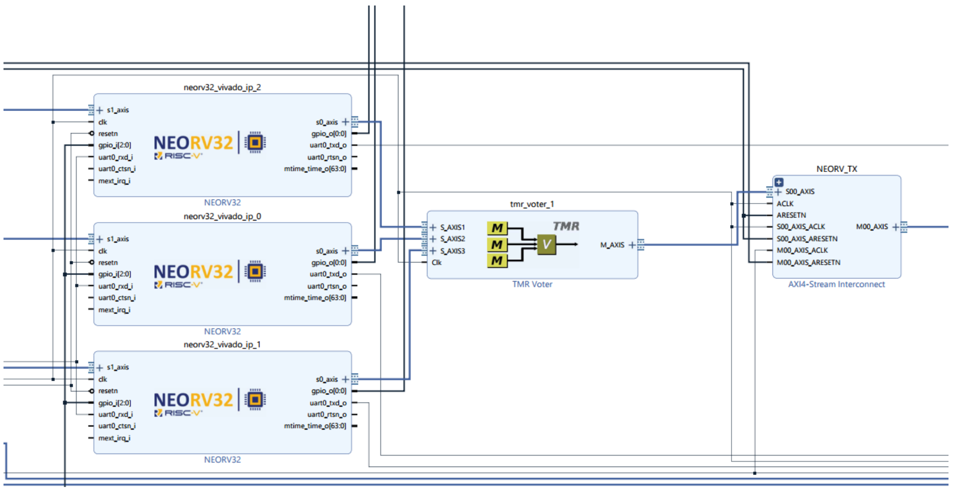


Figure 4.6: TMR Voter

1. an **AXI4-Stream TMR voter**, performing cycle-by-cycle majority voting on `tdata`, `tlast`, and control signals produced by the three processors before injecting them into the Aurora transmitter;
2. a **UART TMR voter**, ensuring coherent debug output even in the presence of faults.

Because AXI4-Stream interfaces may pause depending on `tready`, synchronizing the three processors became essential. The hardware voter ensures that a transient SEU corrupting only one processor does not propagate to the Aurora datapath. Together, these measures produce a supervisory subsystem capable of masking single faults.

4.4 DFX Based Error Correction

To further enhance system reliability, we combine TMR on the NEORV32 with an error-correction strategy for the Aurora IP based on Dynamic Partial Reconfiguration (DPR). When the Aurora core shows erroneous behavior, we reconfigure

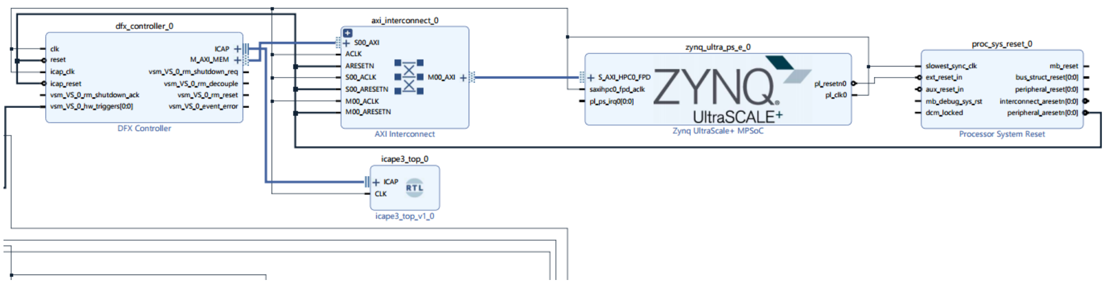


Figure 4.7: DFX Controller

only its region instead of reloading the entire FPGA. Errors are detected either in software, through CRC checks performed by the NEORV32 on Aurora data streams, or by monitoring the channel and lane signals, which should remain asserted during normal operation; any unexpected de-assertion is treated as a fault and triggers recovery. We implement DPR using AMD's Dynamic Function eXchange (DFX) flow, set up with the DFX Wizard [26] and managed at runtime by the DFX Controller [27]. A key element of this flow is the Pblock. A Pblock is a physical floorplanning constraint defined in a Vivado constraint file (XDC)

```
create_pblock aurora
add_cells_to_pblock [get_pblocks aurora] [get_cells -quiet [list design_1_i/aurora_64b66b_0]]
resize_pblock [get_pblocks aurora] -add {CLOCKREGION_X3Y6:CLOCKREGION_X3Y6}
```

that carves out a specific, named region of the FPGA fabric—i.e., a fixed set of device resources (logic, BRAM, DSP, clocking, and routing capacity) at defined locations—and reserves it for part of the design. By constraining the Aurora core to its own Pblock (as shown in Figure 5.5), we create a clean reconfigurable partition with clear resource and placement boundaries. This partitioning causes implementation to produce both a full bitstream for the static design and a partial bitstream that programs only the Aurora Pblock. Although DFX is often used to switch between alternative implementations of a module to match workload needs, here we use the same mechanism for error correction. Reloading the Aurora partial bitstream refreshes only the configuration memory associated with that Pblock, clearing non-destructive single-event effects that may have corrupted the Aurora region while leaving the rest of the system undisturbed. To keep the system independent of external intervention, the NEORV32 can request partial reconfiguration directly from the DFX Controller. The partial bitstream is stored in the ZCU102 DDR memory; when DPR is triggered, the controller fetches the partial image from DDR and programs the configuration memory for the Aurora Pblock. This arrangement enables a fast and localized repair path: the static logic, including the TMR-protected NEORV32 and other unaffected modules, continues operating while the Aurora region is refreshed. In practice, this reduces downtime, avoids a full reboot, and restores link functionality quickly once CRC checks or link-status monitoring indicate a fault. In summary, coupling TMR on the processor with DPR-based repair of the Aurora IP yields an efficient and dependable error-correction technique that improves availability without adding significant resource or complexity overhead.

Chapter 5

Experimental results

5.1 Experimental setup

The experimental validation of the proposed architecture has been performed using the **ZCU102 Evaluation Board**, a reference platform based on Zynq UltraScale+ Multiprocessor System-on-Chip (MPSoC). Its combination of high-speed GTH transceivers, programmable clock resources, and a rich PL fabric makes it particularly suited for prototyping radiation tolerant communication architectures and high throughput serial links such as Aurora 64b/66b.

A functional overview of the board is shown in Figure 5.1, extracted from the official user guide [21]. Even at a glance, the layout highlights the integration of several subsystems that are fundamental for the work presented in this thesis, including the XCZU9EG MPSoC, multiple FPGA Mezzanine Card (FMC) expansion connectors, programmable reference oscillators and a wide set of high-speed interfaces.

The ZCU102 is built around the XCZU9EG-2FFVB1156 device, which combines a quad core ARM Cortex A53 processing system with a large and flexible PL region. According to the official documentation [21], the device integrates more than 5000 logic cells, more than 500,000 flip flops, 32 Mb of block RAM, and more than two,000 DSP slices. In addition, the device provides 24 GTHs transceivers on the PL side, supporting line rates up to 16.3 Gb/s. This amount of resources offers ample headroom for implementing triplicated soft cores, TMR voting logic, buffering structures, and the complete Aurora based datapath.

5.1.1 High-Speed Transceiver Infrastructure

A crucial reason for selecting the ZCU102 is its extensive GTH transceiver availability. The board exposes high performance serial channels on several connectors, including:

Figure 5.1: ZCU102 evaluation board source [21].

- two FMC high-pin-count connectors (HPC0 and HPC1), each providing access to multiple GTH lanes;
- four dedicated SFP+ channels driven by a complete GTH quad;
- three GTH lanes routed to HDMI TX and RX ports;
- a GTH lane available on SMA connectors for direct differential testing.

The distribution of these transceivers across banks 128, 129, 130, 228, 229, and 230 is documented in the user guide [21]. This diversity of high-speed interfaces enables both electrical and optical experimentation, including loopback setups, inter board communication, and link robustness evaluation.

In addition to high-speed transceivers, the ZCU102 offers a broad set of interfaces suitable for on-board processing and communication experiments. These include PS and PL UARTs, USB 3.0, SD, SATA, DisplayPort, Ethernet, GPIO headers, PMOD connectors, and a complete System Monitor for On-Chip Voltage and Temperature (SYSMON) subsystem to monitor supply voltages and temperature. This rich connectivity makes the board not only a transceiver evaluation platform but also a complete environment for mixed hardware–software prototypes.

5.1.2 Board-Level Constraints and Clocking Configuration

The constraint file used in this work mirrors the physical organization of the ZCU102 and ensures strict coherence with the board layout. All clock and transceiver signals are mapped according to the assignments reported in the user guide [21], guaranteeing correct electrical behavior and proper operation of the Aurora core.

GTH Reference Clock. The Aurora channel relies on the differential reference clock GT _DIFF_REFCLK1_0, driven by the SI570 MGT oscillator. The following constraints define its position and timing:

```
create_clock -period 6.400 -name GT_DIFF_REFCLK1_0_clk_p \
    -waveform {0.000 3.200} \
    [get_ports GT_DIFF_REFCLK1_0_clk_p]

set_property PACKAGE_PIN C7 [get_ports GT_DIFF_REFCLK1_0_clk_n]
set_property PACKAGE_PIN C8 [get_ports GT_DIFF_REFCLK1_0_clk_p]
```

This clock is isolated from the SI570 user clock using an asynchronous clock-group declaration, since the two oscillators are independent:

```
set_clock_groups -asynchronous \
    -group [get_clocks -of_objects [get_ports
  ↪ GT_DIFF_REFCLK1_0_clk_p]] \
    -group [get_clocks -of_objects [get_ports
  ↪ user_si570_sysclk_clk_p]]
```

Transceiver Serial I/O. The constraint file maps the Aurora serial lanes to the corresponding GTH pins:

```
set_property PACKAGE_PIN D2 [get_ports {GT_SERIAL_RX_0_rxp[0]}]
set_property PACKAGE_PIN D1 [get_ports {GT_SERIAL_RX_0_rxn[0]}]
set_property PACKAGE_PIN E4 [get_ports {GT_SERIAL_TX_0_txp[0]}]
set_property PACKAGE_PIN E3 [get_ports {GT_SERIAL_TX_0_txn[0]}]
```

which correspond to the differential TX/RX pairs of the selected GTH quad described in the board documentation.

Isolation of the Aurora User Clock. Since the Aurora user clock is generated internally by the transceiver and does not share a fixed phase relationship with any external source, it is also isolated:

```
set_clock_groups -asynchronous \
    -group [get_clocks -of_objects [get_pins design_1_i/
  ↪ aurora_64b66b_0/user_clk_out]]
```

This prevents Vivado from trying to complete timing in unrelated domains.

The constraint file reflects the physical and logical organization of the ZCU102 and ensures that all transceiver, clocking, and supervisory interfaces operate within the electrical and timing limits dictated by the board.

To validate the complete architecture under realistic operating conditions, the ZCU102 evaluation board was integrated into a multi-board test stack, shown in Figure 5.2. The experimental setup consists of **four interconnected boards** arranged in a vertical stack through dedicated mechanical spacers and wired via high-speed serial links and USB connections. Although the platform supports experiments involving all four boards, **only the first two were used in this work**, as the implemented architecture requires a single Aurora 64B/66B endpoint and one supervisory subsystem for correct validation.

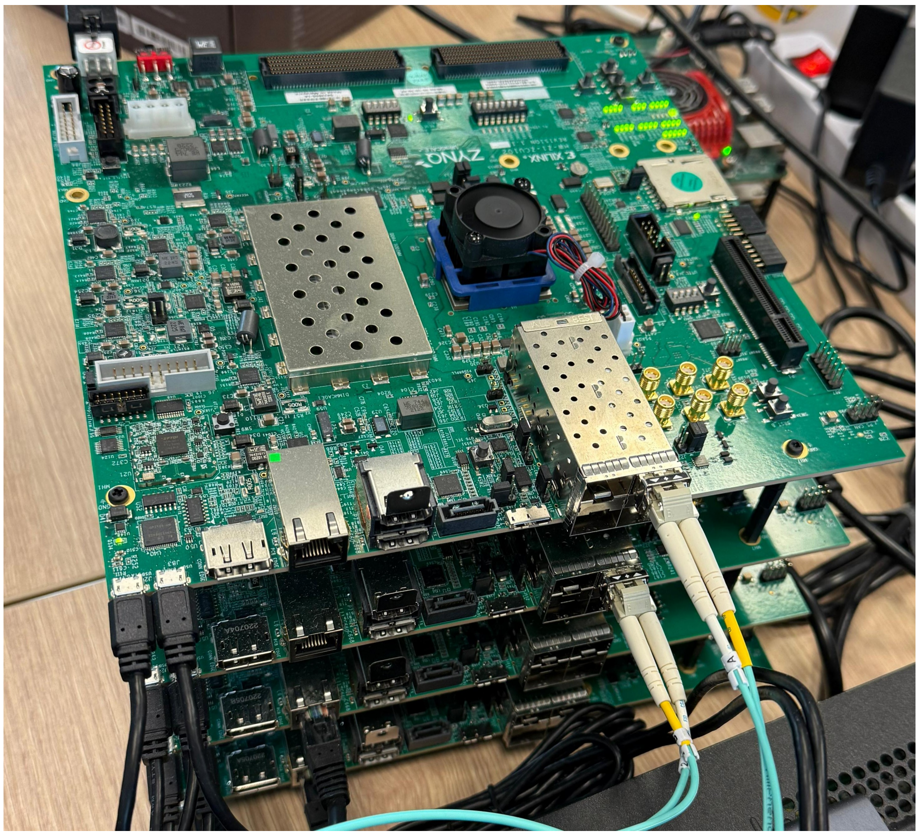


Figure 5.2: Side view of the experimental setup, showing the interconnections and SFP+ cabling.

The ZCU102 board provides the high-performance GTH transceiver used to implement the Aurora 64B/66B link, while the secondary board hosted the remote endpoint of the communication channel. The two remaining boards, although powered, did not participate in the transmission chain or supervisory logic and were not required for the experiments conducted in this thesis.

The stack is powered through individual USB connections and controlled using a dedicated USB hub, ensuring stable and isolated power delivery during long-duration experiments, including fault-injection campaigns and dynamic configuration tests. The optical and electrical links shown in the figures were used to validate clock recovery, link recovery, and continuous data streaming under the Aurora protocol.

The target operating frequency for the design was set to **160 MHz**, following a structured *trial-and-error* methodology that progressively evaluated the maximum stable frequency supported by the system. The initial reference value of **100 MHz** was chosen based on common practice for early Aurora prototypes and on the default configuration of the internal clocking resources of the ZCU102.

At each iteration, the target frequency was increased, the design was synthesized, placed, and routed, and the timing stability was validated through static timing analysis. The objective was twofold:

- maximize the operating performance of the datapath, reducing latency, and increasing AXI4-Stream throughput;
- reach the practical upper limit imposed by the Aurora 64B/66B core [20] and the GTH transceiver configuration of the XCZU9EG device [21].

The selected transceiver settings and the 16.3 Gb/s GTH capability allow user-clock frequencies well above 150 MHz. During implementation, Vivado consistently met the timing at **160 MHz**, achieving a positive slack. Attempts to raise the restriction beyond 160 MHz produced negligible performance gains while increasing routing pressure near the GTH columns.

For these reasons, **160 MHz was selected as the optimal operating point**:

- safely within the frequency envelope supported by the Aurora configuration;
- maximizing throughput without compromising timing stability;
- maintaining low dynamic power consumption;
- offering wide safety margins for radiation-tolerant and fault-injection experiments.

This choice ensured a robust and deterministic operating point that remained stable across all experiments performed on the dual-board setup.

The utilization of the entire design resource, implemented at a target frequency of 160 MHz, is summarized in Table 5.1. The NEORV32 subsystem, including the TMR majority voters, is the dominant contributor to logic usage, as expected from the triplication required by the TMR architecture. The Aurora subsystem accounts for approximately one quarter of the total LUT count, including the GTH transceiver interface and the AXI4-Stream infrastructure required for the 64b/66b datapath. The remaining blocks, which implement clocking, reset handling, interconnection, and DFX support logic, represent roughly 15% of the total logic footprint. In general, the design comfortably fits within the ZCU102 device, consuming less than 3% of LUTs and less than 10% of BRAM resources.

Table 5.1: Resource utilization summary at 160 MHz

Module Group	**LUT**	**FF**	**BRAM**	**GTH Ch.**
NEORV32 (TMR + voters)	4107	3981	51	0
Aurora subsystem	1964	3345	1	1
Other logic	1170	1453	0	0
Total	**7241**	**8779**	**52**	**1**

5.2 Power and Timing Characterisation

A detailed power and timing analysis was performed on the implemented design to assess its performance margin, thermal behavior, and suitability for deployment within resource-constrained on-board processing platforms. The power figures were extracted from the final implemented netlist using the Vivado vector-less power engine, while the timing analysis was carried out during sign-off static timing analysis (STA) after place-and-route.

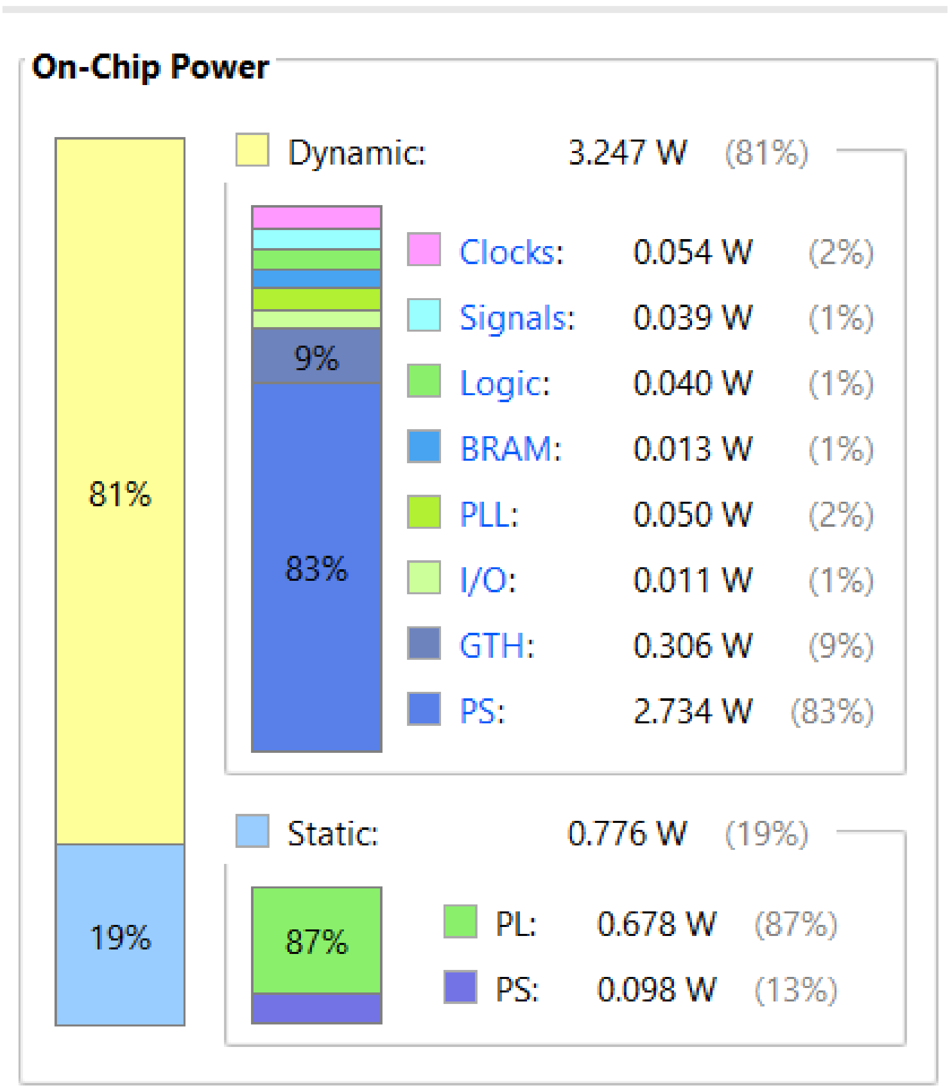
The power breakdown on the chip is summarized in Table 5.2. The total power consumption on the chip amounts to **4.023 W**, with dynamic power representing approximately **81%** of the total power and static power accounting for the remaining **19%**. The dominant contribution comes from the Processing System (PS), which alone dissipates **2.734 W** dynamically due to the activity of the ARM cores, interconnect, and memory subsystem.

Despite operating a high-speed Aurora 64B/66B serial link, the GTH transceiver contributes only **0.306 W** to dynamic power (about **9%**), confirming the efficiency of the transceiver configuration at the tested line rate. Programmable Logic (PL), including triplicated NEORV32 processors, TMR voters, AXI4-Stream data path and supervisory logic, contributes **0.040 W** of dynamic power and **0.678 W** of static power, corresponding to roughly **3%** of the power envelope of the device.

The resulting junction temperature is **28.9 °C**, with a thermal margin of more than **71 °C** relative to the device limit. This confirms stable operation even without active cooling and demonstrates full compatibility with typical spacecraft thermal and power constraints.

Table 5.2: On-chip Power Breakdown at 160 MHz

Category	**Dynamic Power [W]**	**Static Power [W]**
Programmable Logic (PL)	0.040	0.678
Processing System (PS)	2.734	0.098
GTH Transceiver	0.306	–
Clocks / PLL / I/O / BRAM	0.167	–
Total	**3.247**	**0.776**



🟨	Dynamic:	3.247 W	(81%)
🟪	Clocks:	0.054 W	(2%)
🟦	Signals:	0.039 W	(1%)
🟩	Logic:	0.040 W	(1%)
🟦	BRAM:	0.013 W	(1%)
🟩	PLL:	0.050 W	(2%)
🟩	I/O:	0.011 W	(1%)
🟦	GTH:	0.306 W	(9%)
🟦	PS:	2.734 W	(83%)
🟦	Static:	0.776 W	(19%)
🟩	PL:	0.678 W	(87%)
🟪	PS:	0.098 W	(13%)

Figure 5.3: Dynamic and static power contributions of the main subsystems of the Zynq UltraScale+ device at 160 MHz.

The power distribution in Fig. 5.3 highlights a fundamental aspect of the architecture: the bulk of power consumption originates from Processing System (PS), which dissipates more than 2.7 W of dynamic power due to ARM cores, cache hierarchy, and interconnect fabric. In contrast, the PL which hosts the triplicated NEORV32 subsystem, the TMR voters, and the AXI4-Stream data path –consumes less than 0.8 W in total. This behaviour confirms the lightweight nature of the proposed hardware design, whose logic footprint and switching activity remain extremely limited.

The GTH transceiver contributes approximately 0.3 W of dynamic power, in line with the expected consumption of a single Aurora lane operating at multi-gigabit rates. Finally, the clocking network (MMCMs, PLLs, and buffer trees) adds a negligible overhead, remaining below 0.2 W. In general, the power profile is fully compatible with the constraints of embedded and space-grade platforms, where thermal stability and low dissipation are essential.

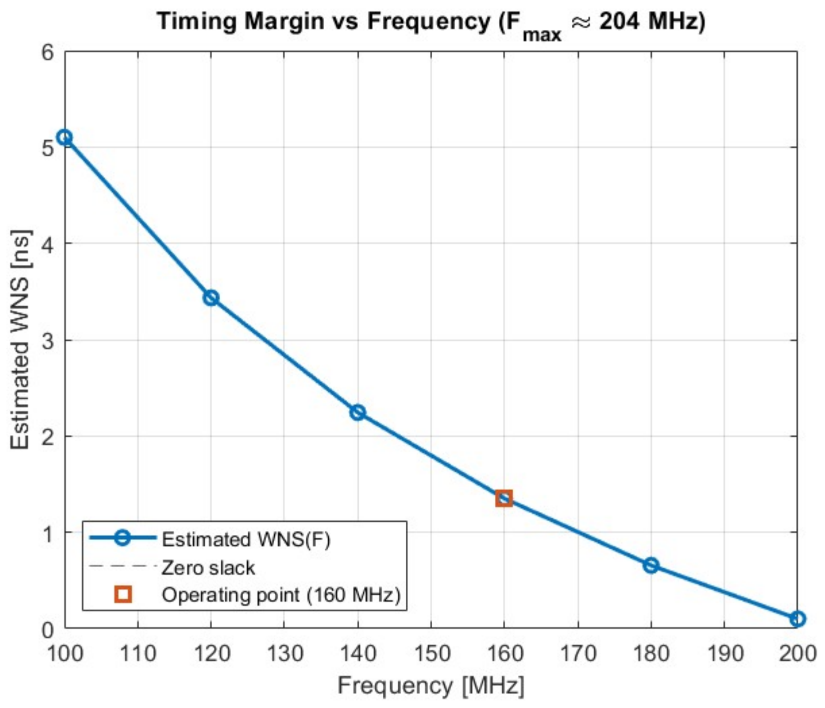


Figure 5.4: Estimated timing margin (WNS) as a function of the operating frequency.

The timing–frequency curve in Fig. 5.4 provides a visual interpretation of the relationship between operating frequency and available setup slack. Starting from the measured WNS at 160 MHz, an extrapolated model of the critical-path delay is used to estimate the slack at nearby frequencies. As expected, increasing the clock frequency reduces the available timing margin, with the WNS approaching zero near 204 MHz. This value represents the approximate limit at which the design would cease to meet timing constraints. The chosen operating point at 160 MHz therefore guaranties a substantial margin of more than 1.3 ns, ensuring stable operation under voltage, temperature and radiation-induced variations.

All timing constraints were met with positive margins in the setup, hold, and pulse-width checks, as summarized in Table 5.3. The worst negative slack (WNS) is **+1.351 ns**, indicating that the critical path is well within the timing budget of the 160 MHz target frequency. Similarly, the worst hold slack (WHS) of **+0.010 ns** and the worst pulse-width slack (WPWS) of **+0.392 ns** confirm that there are no hazards or timing instabilities in the implemented datapath.

Based on the available setup slack, the estimated maximum operating frequency can be approximated as

$$f_{\max} = \frac{1}{T_{\text{target}} - \text{WNS}} = \frac{1}{6.25\ \text{ns} - 1.351\ \text{ns}} \approx 204\ \text{MHz},$$

providing approximately **25% timing headroom** beyond the nominal operating frequency.

Table 5.3: Timing Summary at 160 MHz

Metric	**Value**	**Failing Endpoints**
Worst Negative Slack (Setup)	+1.351 ns	0
Worst Hold Slack	+0.010 ns	0
Worst Pulse Width Slack	+0.392 ns	0
Total Failing Endpoints		**0**

The combined power and timing results indicate that the proposed architecture is computationally efficient, thermally stable, and exhibits a substantial performance margin. The PL logic operates with extremely low dynamic power consumption, reflecting the moderate utilization of the FPGA fabric despite the presence of TMR logic and triplicated processors.

The Aurora 64B/66B link maintains low-power operation while meeting all timing constraints with wide margins, demonstrating robust implementation with minimal routing congestion and optimal placement near the GTH transceiver column.

Overall, the system fits comfortably within the constraints of the ZCU102 device, offering ample margin in terms of power, timing, and thermal behavior. The results obtained confirm that the architecture is suitable for deployment in mission-critical and radiation-sensitive contexts, where reliability, determinism, and resource efficiency are essential.

5.3 Result Analysis

Validation of the proposed architecture was carried out on the Zynq UltraScale+ ZCU102 evaluation board, utilizing the integrated GTH transceivers and the deterministic clocking network that the platform provides. The board-level infrastructure, described in detail in the official user guide [21], ensures stable high-speed serial operation and predictable behavior of the transceiver resources used by the Aurora 64B/66B channel. The clocking and reset scheme, generated, respectively, through the Clocking Wizard [25] and the Processor System Reset IP [23], was verified to release all domains synchronously and without metastability risks, providing a clean and deterministic start for the entire communication subsystem.

The fully implemented design is shown in Fig. 5.5. The floorplan highlights the automatic placement of the Aurora core adjacent to the GTH transceiver column, as recommended by the Aurora 64B/66B Product Guide [20]. This placement ensures minimal routing delays, optimal signal integrity, and the correct interaction between the Aurora lane logic and the physical transceiver primitives. The rest of the datapath, including the AXI4-Stream infrastructure, register slices, interconnect elements, and width converters, is placed consistently across the device, without congestion or timing violations.

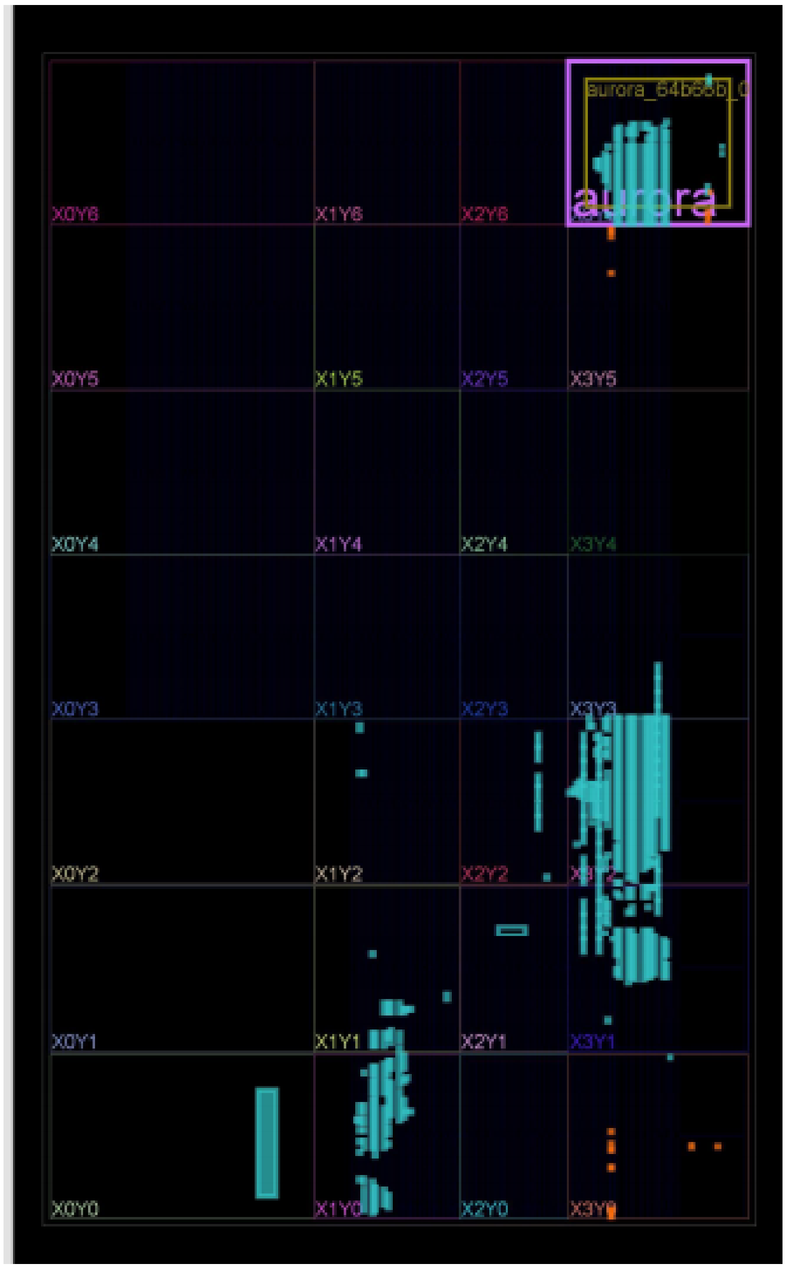


Figure 5.5: Implemented design floorplan on the ZCU102.

To verify the correct behavior of the streaming datapath, the Integrated Logic Analyzer ILA was inserted into the design and connected to both Aurora channel interfaces and the AXI4-Stream interconnect. Fig. 5.6 shows the first capture, where the 32-bit data produced by the triplicated NEORV32 processors are injected into the Aurora TX path. The TVALID/TREADY handshake remains continuously asserted, demonstrating that no back-pressure is generated anywhere along the datapath. The payload increases monotonically, confirming that AXI4-Stream the broadcaster and other infrastructure modules follow the timing, routing, and behavior rules documented in the AXI4-Stream Infrastructure Suite [22].

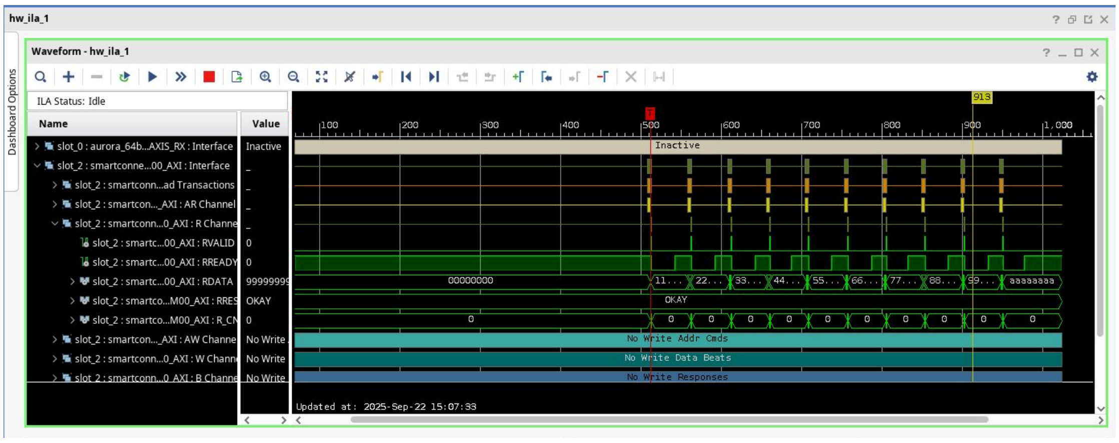

hw_ila_1
Waveform - hw_ila_1
ILA Status: Idle
Name
Value
slot_0 : aurora_64b...AXIS_RX : Interface
Inactive
slot_2 : smartconne...00_AXI : Interface
slot_2 : smartconn...ad Transactions
slot_2 : smartcon..._AXI : AR Channel
slot_2 : smartconn...0_AXI : R Channel
slot_2 : smartc...00_AXI : RVALID
slot_2 : smartc...00_AXI : RREADY
slot_2 : smartc...00_AXI : RDATA
99999999
slot_2 : smartco...M00_AXI : RRESP
OKAY
slot_2 : smartco...M00_AXI : R_CN
slot_2 : smartcon..._AXI : AW Channel
No Write
slot_2 : smartconn...0_AXI : W Channel
No Write
slot_2 : smartconn...0_AXI : B Channel
No Write
No Write Addr Cmds
No Write Data Beats
No Write Responses
Updated at: 2025-Sep-22 15:07:33
Dashboard Options

Figure 5.6: ILA capture of the AXI4-Stream datapath

A second ILA capture, shown in Fig. 5.7, focuses on the receiving side of the channel. The Aurora core outputs 64-bit words that must be resized to 32 bits before being delivered to the supervisory processors. The AXI4-Stream Interconnect internally performs this operation exactly according to the behavior described in [22], splitting each Aurora word into two aligned 32-bit beats while preserving TLAST, TKEEP and the order of the data. The figure clearly shows correctly aligned 64-bit Aurora frames (top) and the reconstructed 32-bit stream (bottom). No gaps, misalignments, or duplicate cycles were observed, and the CRC logic embedded in the Aurora core [20] did not report any errors during the entire test window.

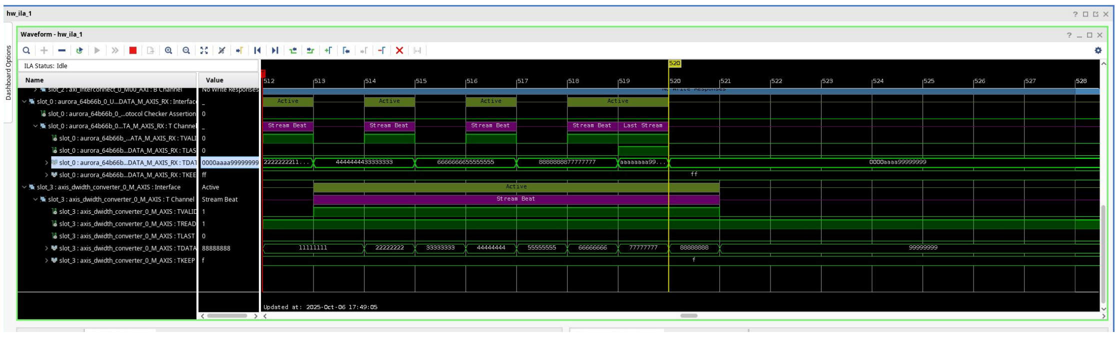

hw_ila_1
Waveform - hw_ila_1
ILA Status: Idle
Name
Value
slot_0 : aurora_64b66b_0_U...DATA_M_AXIS_RX : Interface
slot_0 : aurora_64b66b_0_...otocol Checker Assertion
slot_0 : aurora_64b66b_0...TA_M_AXIS_RX : T Channel
slot_0 : aurora_64b66b...DATA_M_AXIS_RX : TVALID
slot_0 : aurora_64b66b...DATA_M_AXIS_RX : TLAST
slot_0 : aurora_64b66b...DATA_M_AXIS_RX : TDATA
0000aaaa99999999
slot_0 : aurora_64b66b...DATA_M_AXIS_RX : TKEEP
slot_3 : axis_dwidth_converter_0_M_AXIS : Interface
Active
slot_3 : axis_dwidth_converter_0_M_AXIS : T Channel
Stream Beat
slot_3 : axis_dwidth_converter_0_M_AXIS : TVALID
slot_3 : axis_dwidth_converter_0_M_AXIS : TREADY
slot_3 : axis_dwidth_converter_0_M_AXIS : TLAST
slot_3 : axis_dwidth_converter_0_M_AXIS : TDATA
88888888
slot_3 : axis_dwidth_converter_0_M_AXIS : TKEEP
Active
Stream Beat
Last Stream
Updated at: 2025-Oct-06 17:49:05
Dashboard Options

Figure 5.7: ILA capture of the Aurora RX datapath.

The design also integrates DFX Controller, included to enable future experiments involving partial reconfiguration.

5.3.1 Fault Injection Analysis

To evaluate the impact of SEE on an FPGA's configuration memory, two complementary strategies are commonly used: radiation testing and fault emulation. In radiation testing, the device is exposed to beams of accelerated particles so that both the cause and the effect of the fault are reproduced, enabling validation under near real-world conditions. This method provides high-fidelity insight into cross-section, error modes, and system-level behavior, but it is constrained by cost, scheduling, and access: dedicated beam time is expensive, setup and instrumentation are non-trivial, and only a limited number of facilities worldwide can host such campaigns. To obtain valid results related to the reliability of a design, fault emulation is often adopted. The idea is to mimic the effect of particle strikes by introducing controlled bit flips in the configuration memory, thereby corrupting the bitstream in a targeted way. Although this does not reproduce the full physics of radiation transport, it preserves the essential failure mechanism at the configuration level and allows rapid, repeatable experiments over large fault sets. In practice, fault emulation makes it possible to sweep a wide range of configuration frames, and quickly evaluate detection latency, recovery time, and coverage of mitigation techniques at a fraction of the cost and time of beam tests. In our work we use PyXEL [28], a bitstream manipulation tool that enables corruption and injection of faults into selected regions of the configuration memory. Through PyXEL we can target the portion of bitstream that hosts the implemented design, inject single-bit upsetss, and then observe the system's response under the same monitoring and recovery flow used in normal operation. This setup lets us quantify how often injected faults propagate to observable errors, how reliably our detection triggers respond, and how quickly Dynamic Partial Reconfiguration restores correct behavior. The platform has been validated through these injection campaigns, and the results are summarized in Table 5.4.

	TX	**RX**
HALT	0,58	0,59
SDC	0,79	0,72
OTHERS	0,04	0,07
TOT	**1,41%**	**1,38%**

Table 5.4: TX and RX Error Rates

A total of 10'000 random injections were performed simultaneously at the

TX and RX sides, specifically targeting the regions of the configuration memory corresponding to the implemented design. Errors were classified into three main categories: **HALT**, indicating a systematic failure in which the system stopped operating (either the Aurora core or the TMR-protected NEORV32); **SDC**, when the data received differed from the data transmitted, producing a CRC error; and **OTHERS**, which aggregates all remaining anomalies (e.g., incorrect word count, unexpected characters, and similar behaviors). As expected, thanks to the redundancy in the NEORV32 with TMR, the number of HALT events is significantly reduced, while most observed faults fall into the SDC category. Overall, the resulting error rates are sufficiently low to indicate good performance and strong reliability against radiation-induced faults.

Chapter 6

Conclusions

This thesis presents a custom architecture for FPGA-based clusters targeting space applications. The combination of a lightweight RISC-V processor and the Aurora IP grants an efficient platform that enables fast and reliable data exchange among different nodes. Mitigation techniques such as Triple Modular Redundancy (TMR) and cyclic redundancy checks (CRC) provide complementary protection at both hardware and software levels, significantly improving overall system reliability and communication integrity. In addition, the use of Dynamic Partial Reconfiguration (DPR), implemented through AMD's Dynamic Function eXchange (DFX) flow and controller, allows the system to rapidly recover from faults affecting the Aurora IP, minimizing downtime and accelerating error recovery when configuration-memory upsets occur. Operating at 160 MHz, the platform achieves excellent communication performance, supporting high-throughput data transfers across nodes. Finally, to evaluate the platform's robustness, dedicated fault-injection campaigns targeted the design regions hosting the TMR-protected NEORV32 and the Aurora core. The results indicate that the system is sufficiently reliable, with an overall error rate lower than 2%.

6.1 Future Work

The proposed architecture serves as a foundation for more advanced space-oriented systems that require efficient and dependable inter-node communication. The number of communication channels and Aurora modules can be increased to support larger clusters, and more sophisticated mitigation strategies can be introduced to further enhance reliability. For example, advanced Forward Error Correction (FEC)—such as LDPC or turbo codes using the integrated SD-FEC IP—can be incorporated to strengthen link robustness under varying channel conditions. A hardened supervisory plane with adaptive FEC policies could further optimize reliability as the environment changes.

Beyond error correction, the DFX Controller present in the design enables dynamic reconfiguration to meet the different payload requirements. This includes scaling up the number of compute units or modifying specific portions of the design on the fly, rather than limiting DPR to recovery. In particular, the increasing presence of on-board accelerators for compute-intensive tasks demands greater system versatility in the face of failures. If a node in the cluster becomes unavailable, workload redirection strategies combined with partial reconfiguration can allow healthy nodes to assume additional computation by provisioning more resources while the faulty node is being repaired.

Finally, more complex RISC-V microarchitectures may be explored, and AXI4-Stream data movement can be offloaded to DMA so that, once a transmission is requested, the processor remains free to execute other tasks while the transfer proceeds autonomously. These directions would extend the proposed platform toward higher performance, improved availability, and even stronger resilience to radiation-induced faults.

Bibliography

[1] L. Sterpone et al. «EuFRATE: European FPGA Radiation-hardened Architecture for Telecommunications». In: *Proc. IEEE/ESA Conf. Aerospace and Telecommunications Systems*. Paris, France, 2022, pp. 1–8. DOI: 10.1109/EUFRATE.2022.00001 (cit. on pp. 14, 17, 27).

[2] S. Azimi, C. D. Sio, A. Portaluri, D. Rizzieri, and L. Sterpone. «Reliability of Space-Grade vs. COTS SRAM-based FPGA in N-Modular Redundancy». In: *Microelectronics Reliability* 138 (2022), pp. 114–125. DOI: 10.1016/j.microrel.2022.114125 (cit. on pp. 14, 16, 22).

[3] A. Waterman, Y. Lee, D. Patterson, and K. Asanović. *The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA, Version 2.2*. RISC-V Foundation. 2017. URL: https://riscv.org/specifications/isa/ (cit. on pp. 14, 23, 24).

[4] D. Mandl, P. Backes, A. Jain, et al. «The JPL Snapdragon Co-Processor: A Compact High-Performance Computer for Spaceflight Applications». In: *Proc. IEEE Aerospace Conference*. 2023, pp. 1–21 (cit. on p. 16).

[5] D. Powell. *High Performance Spaceflight Computing (HPSC) Overview*. Tech. rep. NASA Jet Propulsion Laboratory, 2025 (cit. on p. 16).

[6] F. Ferrandi, A. Leva, et al. «A Survey on FPGA-Based Heterogeneous Cluster Architectures». In: *ACM Computing Surveys* 52.6 (2019), pp. 1–34 (cit. on p. 16).

[7] *Space Solutions Brochure*. Microchip Technology Inc., 2023 (cit. on p. 16).

[8] *Radiation-Tolerant FPGA Portfolio Overview*. Microchip Technology Inc., 2022 (cit. on p. 16).

[9] L. Sterpone, M. Porrmann, and J. Hagemeyer. «A Fault-Tolerant and Runtime Reconfigurable Platform for Satellite Payload Processing». In: *IEEE Transactions on Computers* 62.8 (2013), pp. 1508–1525. DOI: 10.1109/TC.2012.121 (cit. on p. 17).

[10] R. Sass, S. Buscemi, et al. «Dynamic Neutron Testing of Dynamically Reconfigurable Processing Modules Architecture». In: *Proc. RADECS*. 2012, pp. 1–8 (cit. on p. 17).

[11] L. Bozzoli et al. «EuFRATE: European FPGA Radiation-Hardened Architecture for Telecommunications». In: *Proc. DATE*. 2023, pp. 1–6 (cit. on pp. 17, 27).

[12] S. Kim, J. Kim, Y. Kim, and H. Choi. «Implementation of Aurora Interface using SFP+ Transceiver». In: *Proc. Int. SoC Design Conference (ISOCC)*. 2022, pp. 350–351. DOI: 10.1109/ISOCC55366.2022.9991234 (cit. on p. 17).

[13] A. Lopez, M. Sicard, et al. «On-board Hyperspectral Processing Using Radiation-Tolerant Kintex UltraScale FPGAs». In: *IEEE JSTARS* (2024), pp. 1–17 (cit. on p. 17).

[14] J. Fernandez, A. Rios, et al. «Parametric Pipelined k-Means Implementation for Hyperspectral Processing on Spacecraft Embedded FPGA». In: *Proc. IEEE Aerospace Conference*. 2019, pp. 1–10 (cit. on p. 17).

[15] S. Nolting. *NEORV32 User Guide*. neorv32.org. 2025. URL: https://github.com/stnolting/neorv32 (cit. on pp. 17, 24, 25).

[16] H. Simoes, A. Canelas, J. Alves, and J. Sousa. «Neutron Radiation Tests of the NEORV32 RISC-V SoC on Flash-Based FPGAs». In: *IEEE Transactions on Nuclear Science* (2023), pp. 1–11 (cit. on pp. 17, 25).

[17] *Radiation Handbook for Semiconductor Devices*. TI Radiation Handbook Version A. Texas Instruments. 2017. URL: https://www.ti.com (cit. on pp. 19–22).

[18] S. Nolting. *NEORV32 Processor Datasheet*. Nightly build datasheet. neorv32.org. 2025. URL: https://github.com/stnolting/neorv32 (cit. on p. 24).

[19] *Aurora 64B/66B Protocol Specification (SP011)*. Version 1.8. Xilinx Inc. 2015. URL: https://www.xilinx.com (cit. on pp. 26, 31, 34).

[20] *Aurora 64B/66B v13.0 Product Guide (PG074)*. Accessed 2025. AMD/Xilinx. 2023. URL: https://docs.xilinx.com (cit. on pp. 26, 27, 29, 31, 34, 43, 49, 51).

[21] *ZCU102 Evaluation Board User Guide*. UG1182. 2023 (cit. on pp. 29, 30, 39–41, 43, 49).

[22] *AXI4-Stream Infrastructure IP Suite*. PG085. 2023 (cit. on pp. 29, 33, 35, 51).

[23] *Processor System Reset*. PG164. 2023 (cit. on pp. 29, 30, 34, 49).

[24] *TMR Subsystem*. PG268. 2023 (cit. on pp. 29, 36).

[25] *Clocking Wizard*. PG065. 2023 (cit. on pp. 30, 49).

[26] *Dynamic Function eXchange Guide*. UG909. 2025 (cit. on p. 37).

[27] *Dynamic Function eXchange Controller*. PG374. 2025 (cit. on p. 37).

[28] Corrado De Sio, Sarah Azimi, Luca Sterpone, David Merodio Codinachs, and Filomena Decuzzi. «PyXEL: Exploring Bitstream Analysis to Assess and Enhance the Robustness of Designs on FPGAs». In: *2023 19th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. 2023, pp. 1–4. DOI: 10.1109/SMACD58065.2023.10192116 (cit. on p. 52).