DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Design and Development of a CubeSat Hardware Architecture with COTS MPSoC using Radiation Mitigation Techniques

KTH Thesis Report

SIDDARTH VASUDEVAN

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Authors

Siddarth Vasudevan <sidvas@kth.se>
Information and Communication Technology
KTH Royal Institute of Technology

Place for Project

Thales Alenia Space GmBH
Ditzingen, Germany

Examiner

Johnny Öberg
Stockholm, Sweden
KTH Royal Institute of Technology

Supervisor

Kalle Ngo

Stockholm, Sweden

KTH Royal Institute of Technology

Abstract

CubeSat missions needs components that are tolerant against the radiation in space. The hardware components must be reliable, and it must not compromise the functionality on-board during the mission. At the same time, the cost of hardware and its development should not be high. Hence, this thesis discusses the design and development of a CubeSat architecture using a Commercial Off-The-Shelf (COTS) Multi-Processor System on Chip (MPSoC). The architecture employs an affordable Rad-Hard Micro-Controller Unit as a Supervisor for the MPSoC. Also, it uses several radiation mitigation techniques such as the Latch-up protection circuit to protect it against Single-Event Latch-ups (SELs), Readback scrubbing for Non-Volatile Memories (NVMs) such as NOR Flash and Configuration scrubbing for the FPGA present in the MPSoC to protect it against Single-Event Upset (SEU)s, reliable communication using Cyclic Redundancy Check (CRC) and Space packet protocol. Apart from such functionalities, the Supervisor executes tasks such as Watchdog that monitors the liveliness of the applications running in the MPSoC, data logging, performing Over-The-Air Software/Firmware update. The thesis work implements functionalities such as Communication, Readback memory scrubbing, Configuration scrubbing using SEM-IP, Watchdog, and Software/Firmware update. The execution times of the functionalities are presented for the application done in the Supervisor. As for the Configuration scrubbing that was implemented in Programmable Logic (PL)/FPGA, results of area and latency are reported.

Keywords

Commercial Off-The-Shelf Multi-Processor System on Chip, Single-Event Upset, Single-Event Latch-up, Cyclic Redundancy Check, Scrubbing, Radiation hardened Micro-Controller Unit, Soft Error Mitigation-Intellectual Property controller, Error Injection, Space packet protocol, NOR Flash, Magnetoresistive RAM

Sammanfattning

CubeSat-uppdrag behöver komponenter som är toleranta mot strålningen i rymden. Maskinvarukomponenterna måste vara pålitliga och funktionaliteten ombord får inte äventyras under uppdraget. Samtidigt bör kostnaden för hårdvara och dess utveckling inte vara hög. Därför diskuterar denna avhandling design och utveckling av en CubeSatarkitektur med hjälp av COTS (eng. Custom-off-The-Shelf) MPSoC (eng. Multi Processor System-on-Chip). Arkitekturen använder en prisvärd strålningshärdad (eng. Rad-Hard) Micro-Controller Unit(MCU) som Övervakare för MPSoC:en och använder också flera tekniker för att begränsa strålningens effekter såsom kretser för att skydda kretsen från s.k. Single Event Latch-Ups (SELs), återläsningsskrubbning för icke-volatila minnen (eng. Non-Volatile Memories) NVMs som NOR Flash och skrubbning av konfigurationsminnet skrubbning för FPGA:er i MPSoC:en för att skydda dem mot Single-Event Upsets (SEUs), och tillhandahålla pålitlig kommunikation mha CRC och Space Packet Protocol. Bortsett från sådana funktioner utför Övervakaren uppgifter som Watchdog för att övervaka att applikationerna som körs i MPSoC:en fortfarande är vid liv, dataloggning, och Over-the-Air-uppdateringar av programvaran/Firmware. Examensarbetet implementerar funktioner såsom kommunikation, återläsningsskrubbning av minnet, konfigurationsminnesskrubbning mha SEM-IP, Watchdog och uppdatering av programvara/firmware. Exekveringstiderna för utförandet av funktionerna presenteras för den applikationen som körs i Övervakaren. När det gäller konfigurationsminnesskrubbningen som implementerats i den programmerbara logiken i FPGA:n, rapporteras area och latens.

Nyckelord

Commercial Off-The-Shelf Multi-Processor System on Chip, Single-Event Upset, Single-Event Latch-up, Cyclic Redundancy Check, Scrubbing, Radiation hardened

Micro-Controller Unit, Soft Error Mitigation-Intellectual Property controller, Error Injection, Space packet protocol, NOR Flash, Magnetoresistive RAM

Acknowledgements

First and foremost, I would like to thank my family and friends for believing in me, encouraging and supporting me in the most difficult of times.
I would like to extend my gratitude to my examiner Johnny Öberg and thesis adviser Kalle Ngo for allowing me to pursue a wonderful thesis topic. Also, I would like to thank Gustavo Ambrosio, Fabian Steinmetz, Satheesh Konduru, Sarthak Kelapure and the whole SDR team from Thales Alenia Space in helping me successfully complete my thesis project.
Finally, I would like to thank KTH and Erasmus team for providing me with scholarship to pursue my master thesis project away from the university.

Acronyms

COTS Commercial Off-The-Shelf
FPGA Field Programmable Gate Array
MCU Micro-Controller Unit
EIVE Exploratory In-Orbit Verification of an E/W-Band
MPSoC Multi-Processor System on Chip
SEM-IP Soft Error Mitigation-Intellectual Property
SEE Single-Event Effect
SEL Single-Event Latch-up
SEU Single-Event Upset
SET Single-Event Transient
SEFI Single-Event Functional Interrupt
CRAM Configuration Random Access Memory
PL Programmable Logic
CMOS Complimentary Metal Oxide Semiconductor
BJT Bi-polar Junction Transistor
SCR Silicon-Controlled Rectifier
SRAM Static Random Access Memory
DRAM Dynamic Random Access Memory
NVM Non-Volatile Memory
OBC On-Board Computer
I2C Inter-Integrated Circuit
EEPROM Electrically Erasable Programmable Read-Only Memory
ICAP Internal Configuration Access Port
DSP Digital Signal Processor
TMR Triple Modular Redundancy
FinFET Fin Field-Effect Transistor

PS Processor System
ECC Error-Correcting Codes
CRC Cyclic Redundancy Check
PCAP Processor Configuration Access Port
APU Application Processing Unit
RPU Real-time Processing Unit
GPU Graphic Processing Unit
UART Universal Asynchronous Receiver Transmitter
SPI Serial Peripheral Interface
GPIO General Purpose Input Output
AXI Advanced eXtensible Interface
PCIe Peripheral Component Interconnect express
DDR Double Data Rate
LUT Look-Up Tables
BRAM Block RAM
FF Flip-Flop
JTAG Joint Test Action Group
MIO Multiplexed Input/Output
EMIO Extended Multiplexed Input/Output
ASCII American Standard Code for Information Interchange
SECDED Single-Error Correction and Double-Error Detection
LET Linear Energy Transfer
EDAC Error Detection And Correction
CAN Controller Area Network
ADC Analog to Digital Converter
DAC Digital to Analog Converter
DMA Direct Memory Access
FRAM Ferroelectric RAM
MRAM Magnetoresistive RAM
MBU Multi-Bit Upset
ISR Interrupt Service Routine

List of Figures

2.1.1 Different types of Single-Event Effects 4
2.2.1 Cubesat 5
2.3.1 Zynq Ultrascale+ MPSoC architecture 9
2.5.1 SEM-IP Controller states 15
2.5.2 11-bit hexadecimal value for error injection 16
2.6.1 Space Packet 17
2.8.1 Rad-Hard Vorago MCU 20

3.1.1 Cubesat hardware architecture 22
3.3.1 Configuration scrubbing block diagram 29

4.1.1 Communication flow chart 31
4.2.1 Watchdog flow chart 33
4.3.1 Memory scrubbing flow chart 35
4.4.1 Software/Firmware update flow chart 37
4.5.1 Configuration scrubbing flow chart 39

5.1.1 Excerpt of output of the Communication implementation 42
5.1.2 Excerpt of output of the Communication implementation 42
5.2.1 Excerpt of output of the watchdog implementation 43
5.3.1 Excerpt of output of the memory scrubbing implementation 44
5.3.2 Excerpt of output of the memory scrubbing implementation 44
5.4.1 Excerpt of output of the image update (24 KB) 46
5.4.2 Excerpt of output of the image update (24 KB) 47
5.4.3 Excerpt of output of the image verification (24 KB) 47
5.4.4 Excerpt of output of the image verification (24 KB) 48
5.5.1 Excerpt of output of the configuration scrubbing implementation . . . 50
5.5.2 Excerpt of output of the configuration scrubbing implementation . . . 50

5.5.3 Excerpt of output of the configuration scrubbing implementation . . . 50

List of Tables

2.3.1 Number of components present in the Ultrascale+ FPGA 9
2.5.1 11-digit hexadecimal value for error injection 16
2.6.1 Space packet primary header 17
2.7.1 CRC-32 description 19
2.7.2 CRC-16 description 19
2.9.1 NOR Flash memory Specifications 21
2.9.2 MRAM Specifications 21

4.1.1 Hardware/Software requirements for communication 32
4.2.1 Hardware/Software requirements for watchdog 34
4.3.1 Hardware/Software requirements for Memory scrubbing 35
4.4.1 Hardware/Software requirements for update and verification 38
4.5.1 ASCII commands for SEM-IP state transitions 39
4.5.2 Hardware/Software requirements for FPGA configuration scrubbing . 39

5.1.1 Execution time for Communication implementation 41
5.2.1 Execution time for watchdog implementation 43
5.3.1 Execution time for memory scrubbing implementation 44
5.4.1 Execution time for software/firmware update/verification implementation 45
5.4.2 Update function execution time for various images 45
5.4.3 Verification function execution time for various images 46
5.5.1 SEM-IP execution time 48
5.5.2 SEM-IP area analysis 48
5.5.3 SEM-IP report description 49

Contents

1 Introduction **1**

1.1 Goals . . . 2
1.2 Delimitation . . . 2
1.3 Structure of the thesis . . . 2

2 Related Work **3**

2.1 Single Event Effects . . . 3
2.1.1 Single-Event Latch-ups . . . 3
2.1.2 Single-Event Upsets . . . 4
2.2 CubeSats . . . 5
2.3 Zynq Ultrascale+ MPSoC Architecture . . . 8
2.3.1 Radiation Tests for Zynq Ultrascale+ MPSoC . . . 10
2.3.2 Built-in Fault-Tolerant features of Zynq Ultrascale+ . . . 11
2.4 FPGA Configuration Scrubbing . . . 12
2.4.1 FPGA Configuration . . . 12
2.4.2 Scrubbing . . . 12
2.5 Soft-error Mitigation Intellectual property . . . 13
2.5.1 Different States of the SEM-IP Controller . . . 14
2.6 Space Packet Protocol . . . 16
2.7 Cyclic Redundancy Check . . . 18
2.8 Radiation Hardened MCU . . . 19
2.9 Memories . . . 21

3 Cubesat Hardware Architecture **22**

3.1 Overview . . . 22
3.2 Supervisor . . . 23
3.2.1 Interfaces . . . 23

3.2.2 Functions 25
3.3 Processing Board 28
3.3.1 Interfaces 28
3.3.2 Functions 28

4 Implementations **30**
4.1 Communication 30
4.1.1 Hardware/Software Requirements 32
4.1.2 Hardware/Software Limitations 32
4.2 Watchdog 33
4.2.1 Hardware/Software Requirements 34
4.3 Memory scrubbing 34
4.3.1 Hardware/Software requirements 35
4.3.2 Hardware/Software Limitations 36
4.4 Software/Firmware Update 36
4.4.1 Hardware/Software requirements 37
4.5 FPGA Configuration Scrubbing 38
4.5.1 Hardware/Software requirements 39

5 Results **41**
5.1 Communication 41
5.2 Watchdog 43
5.3 Memory scrubbing 44
5.4 Software/Firmware update 45
5.5 FPGA configuration study 48

6 Conclusion and Future works **52**
6.1 Conclusion 52
6.2 Limitations 53
6.3 Future Work 53

Bibliography **54**

Chapter 1

Introduction

Space Missions are generally carried out using traditional satellites made by government agencies or other big organizations. With the advent of CubeSats[6], small companies and educational institutions made their way into the space domain by using it as a tool for educational purposes by carrying out small-scaled missions[26]. The change is attributed to the low-cost design and development of the payload and also the CubeSat itself. Nowadays, Commercial Off-The-Shelf (COTS) products such as Field Programmable Gate Array (FPGA)s, Micro-Controller Unit (MCU)s are being used to develop the payloads for the CubeSats as they are affordable, accessible, and require a short time to develop them as a product. However, they are not reliable when it comes to space missions because of their vulnerability towards radiation. Exploratory In-Orbit Verification of an E/W-Band (EIVE) is one such project initiated by the University of Stuttgart that aims to develop an E-Band communication using a CubeSat equipped with COTS hardware[20].

This thesis work aims to propose a reliable CubeSat hardware architecture, which employs a powerful COTS MPSoC, the Zynq Ultrascale+, which will be used in the EIVE project in developing their application. Several radiation mitigating techniques are incorporated into the design to increase the resilience of the COTS MPSoC. This project considers the Zynq Ultrascale+ MPSoC as a black box and concentrates on building a haven around it to recover it from the adverse radiation effects.

1.1 Goals

The primary goal of this thesis work is to prototype a mission independent architecture for the CubeSat using a COTS MPSoC and other components that support it to withstand the radiation effects without compromising the functionality. This can be subdivided into two categories:

- Monitoring and rectifying the MPSoC, where the application for the space mission is built, through an external Supervisor. This Supervisor is also responsible for the communication between On-Board Computer and the MPSoC.
- Monitoring and rectifying the FPGA present in the MPSoC from within by incorporating Configuration scrubbing.

1.2 Delimitation

The Thesis work presented in this paper proposes and prototypes proof of concept architecture for the CubeSat. It is not tested in the radiation environment because of the huge expense to validate the design in radiation facilities. Also, this experimental thesis will involve certain techniques to mitigate latch-ups. However, testing this is practically difficult. Nevertheless, the Configuration Scrubbing can be tested by injecting errors using the Soft Error Mitigation-Intellectual Property (SEM-IP) controller.

1.3 Structure of the thesis

Chapter 2 discusses the background study and literature survey related to the work of this thesis project. Chapter 3 discusses the architecture of the CubeSat and explains the design decisions. Chapter 4 talks about the experimental designs implemented in this thesis work regarding the architecture. Chapter 5 talk about the results and analysis of the implemented designs along with the testing of the configuration scrubbing. Chapter 6 projects the results obtained from the implementations. Chapter 7 concludes the thesis work by summarizing the project and reveals the limitations of the project. Finally, possible future works are listed.

Chapter 2

Related Work

2.1 Single Event Effects

Single-Event Effect (SEE) is caused when high energy-charged particles strike the device and lose energy by ionizing the substrate atoms. It expels the electrons out of the substrate during this process[14]. This causes an electron-hole pair to be created, giving rise to a potential difference. When the charge collection exceeds the threshold, a SEE is generated. The SEE can be classified into two categories, Destructive and Non-destructive. Destructive SEEs will cause a permanent failure of the device while Non-destructive SEEs will lead to a loss of state or data but does not physically harm the device. The SEEs focused within this thesis work are Single-Event Latch-up (SEL)s, that fall under the destructive subdivision of SEEs and Single-Event Upset (SEU)s, that comes under the non-destructive subdivision of SEEs. This is because Xilinx claims their Ultrascale architecture devices to be immune to gate ruptures and burnout conditions[29]. However, there are some cases where SELs were being reported, which will be covered in this chapter. Similarly, SETs and SEFIs in Ultrascale architectures are very low[29], making them negligible, hence SEUs become a primary concern as they affect Configuration RAM (CRAM) of the Programmable Logic (PL), i.e., the FPGA. Figure 2.1.1 outlines the different types of SEEs that can affect a device.

2.1.1 Single-Event Latch-ups

Most of the Integrated Circuits currently manufactured make use of Complimentary Metal Oxide Semiconductor (CMOS) transistors in them. A parasitic Bi-polar Junction

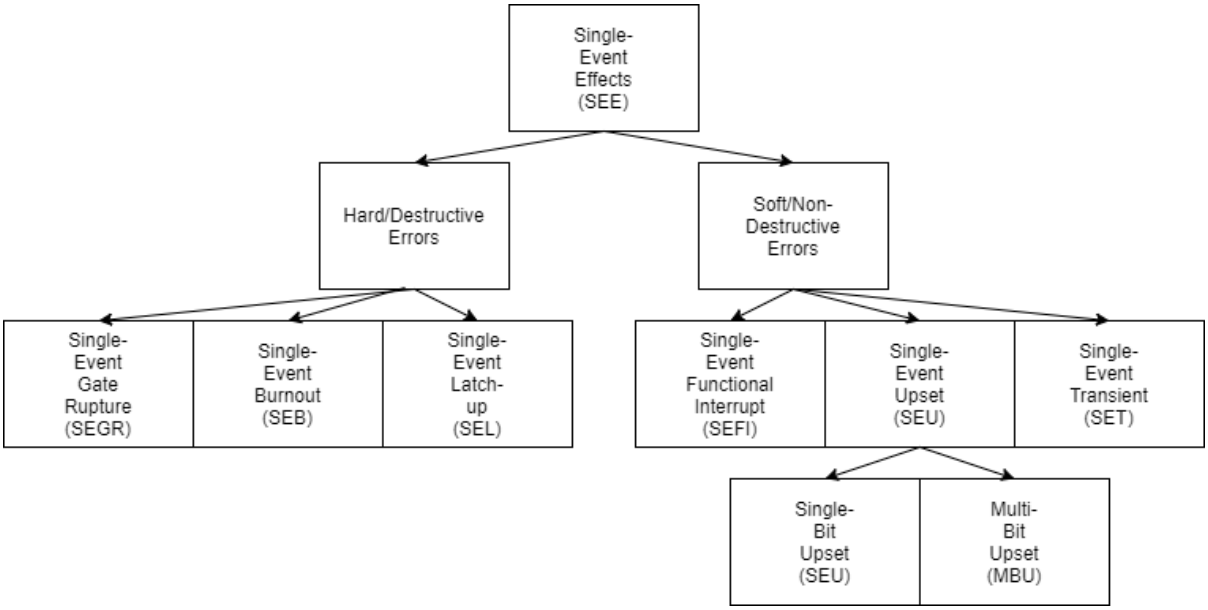

Figure 2.1.1: Different types of Single-Event Effects

Transistor (BJT) structure is formed when the CMOS transistors are closely located, and these behave like a Silicon-Controlled Rectifier (SCR). In a positive feedback loop (SCR), these parasitic structures are dormant under normal operating conditions. However, the slightest of current in them will lead to a high-current output because of the positive feedback loop. This will remain until a power-cycling is done. This high-current has the potential to damage the device permanently. This scenario is referred to as a SEL[14].

2.1.2 Single-Event Upsets

Single-event Upsets[14], which falls under the category of non-destructive SEEs, is commonly referred to as bit-flips. When a heavy-ion or proton strikes a memory cell such as a flip-flop, it causes a change of state in that memory cell. This change of state is known as a SEU. The SEUs typically occur to memory elements like SRAMs, DRAMs, and Non-Volatile Memory (NVM)s like NOR Flash memory. SRAM based FPGAs are sensitive to radiation for the same reason. Their configuration Memories have their bits flipped, leading to a change in the functionality of the FPGA[25]. The SEUs are corrected by rewriting the original content to the memory elements.

2.2 CubeSats

Cubesats, short for Cube Satellites that come under the nanosatellites class, are used for research and educational purposes. Generally, they come in a standard measurement unit of 10 cm X 10 cm X 10 cm[6]. This cubic unit size is also termed as 1 U. CubeSats are scalable and can also be made in 2 U, 3 U, or 6 U measurements, and they cannot weigh more than 1.33 kg/1U [27]. Figure 2.2.1 portrays the way a 1 U CubeSats look like in general.

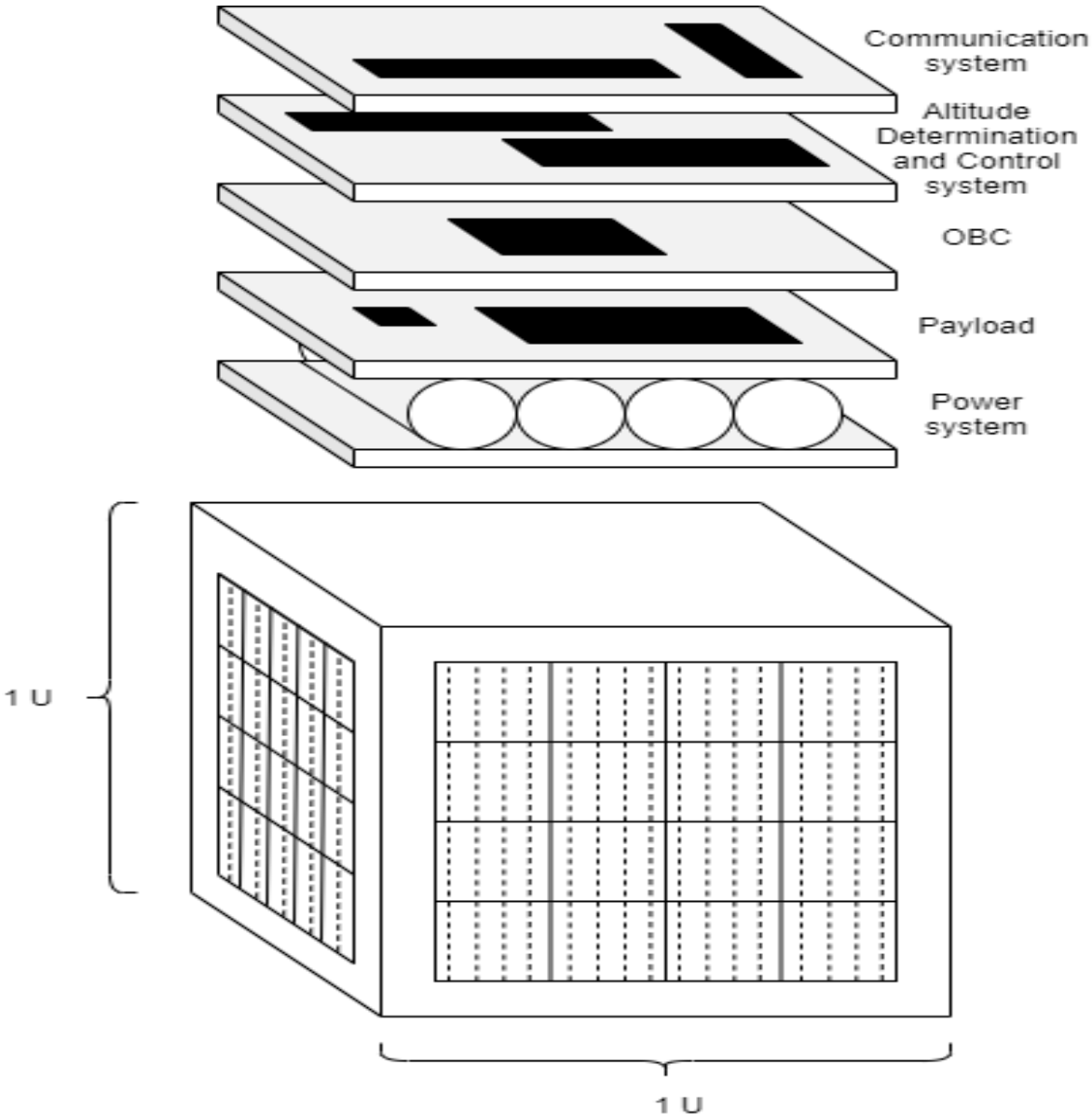

Figure 2.2.1: Cubesat

As aforementioned in the introduction, these CubeSats paved the way for a niche market that allowed institutions and companies to develop small-scaled space missions

or auxiliary missions for big satellites. Some of the CubeSat missions and their hardware architectures described below are conducted in the previous years:

- Chandrasekhar Nagarajan et al. [15] have designed a CubeSat hardware architecture as a part of a space mission that carries out terrestrial thermal imaging. The architecture employs two On-Board Computer (OBC) using micro-controllers. The primary OBC (TI MSP430) enforces operations related to sub-modules communication, sensor data acquisitions, processing, and storing. It also performs health monitoring and housekeeping of components. This OBC downlinks a telemetry packet to the ground station. The secondary OBC (STM32 Cortex-M3) manages communication from the ground station to the CubeSat. Also, it is responsible for the storage of thermal images captured by a 14-bit thermal imaging camera and sending the image data to the ground station. The two MCUs interact through I2C serial communication protocol.

- Similarly, Dua A.M. Osman, et al. [17] have proposed a CubeSat hardware architecture for the application of geographical imaging of Khartoum, Sudan. This proposed architecture, unlike the architecture mentioned in the previous paragraph, uses a single OBC and it makes use of COTS MCU. This OBC is interfaced with several peripherals such as a Temperature sensor, a Real-Time clock, an EEPROM, a camera, and a control system. It manages the entire communication of the CubeSat and regulates the camera to capture the image and store it in the EEPROM.

- Christian Fuchs et al. [8], discusses a CubeSat architecture that aided a 4 year ESA project. This design uses COTS FPGA which houses a tiled MPSoC architecture. ARM Cortex-A53 is used as the processor in each tile, and they have dedicated peripheral IP-cores, caches, and local interconnects. Tiles are separated from each other, meaning; they have their re-configurable partition and a clock domain to enable partial reconfiguration. Tiles have shared access to all the memories. The tiled MPSoC and whole of the FPGA are monitored using an external Supervisor, which happens to be a MCU. This paper proposes radiation mitigation techniques with this said architecture. This is done in 3 stages:

  - Stage 1: Each tile performs thread replication, that is, running the same application in different threads. This is done along with lockstep. This

enables thread synchronization and, at the same time, checks for issues within the tile. A voting system between the replicated threads notifies the occurrence of an error if any. This information is sent to the Supervisor from every tile.

- Stage 2: The Supervisor is responsible for keeping track of error counts and solving any aroused discrepancies from any of the tiles. If the voting decision from any tile indicates an error, then the Supervisor issues a partial reconfiguration of the affected tile. This is done through Internal Configuration Access Port (ICAP) of the FPGA using the SEM-IP controller without affecting the functionality of the whole system since they have their re-configurable partition. The Supervisor can also issue a full-reconfiguration to reset all the tiles. The Supervisor also keeps track of the number of times a tile is getting affected. If this number crosses a defined threshold, then the Supervisor deems this tile as permanently damaged and replaces it with a new tile.

- Stage 3: This stage is only when the hardware is aged and has fewer resources to maintain the same functionality and performance. In this case, the highly critical application is given the utmost priority and made to run with the remaining available resources.

- The paper by Zhen Dong, et al. [7] discusses about a reliable CubeSat OBC architecture using re-configurable FPGA. The proposed scheme uses 2 FPGAs and 4 Digital Signal Processor (DSP)s out of which 1 FPGA and 3 DSPs work at any given point of time, and others serve as backup hardware in case a fault occurs in the working hardware. The FPGA is responsible for state information, commands, communication between interfaces and DSPs, data gathering along with the necessary information for the spacecraft. Whereas the DSP is where the processing of data collected by FPGA takes place. Triple Modular Redundancy (TMR) is implemented using the 3 DSPs, while the fourth DSP serves as a backup for the other three. The FPGA gathers the voted results and checks for faults in the DSP. In case a fault is reported, then it replaces the affected DSP with a backup DSP. Similarly, in the case of the FPGA, if it stops functioning, or if the switching time runs out, the backup FPGA comes into play replacing the existing one.

In this thesis work, CubeSat hardware architecture is proposed using a COTS MPSoC and an external Rad-Hard MCU that acts as a Supervisor. A few functionalities of the proposed architecture were implemented and tested.

2.3 Zynq Ultrascale+ MPSoC Architecture

Zynq Ultrascale+ EG MPSoC[3] is one of the most powerful and versatile chips currently present in the market. Because of its unique heterogeneous architecture, this MPSoC opens up various opportunities and finds its application in various fields such as IoT, 5G, and industries, such as automotive and aerospace. This 16 nm FinFET technology houses the following features:

- Quad-core ARM Cortex-A53 Application Processing Unit
- Dual-core ARM Cortex-R5 Real-time Processing Unit
- Programmable Logic (FPGA)
- Platform Management unit
- Graphics Processor Unit
- Gigabit transceivers
- High-speed serial interface

Figure 2.3.1 shows the architecture of Zynq Ultrascale+ MPSoC. The architecture is divided into Processor System (PS) and Programmable Logic(PL), connected together by Advanced eXtensible Interface (AXI) bus. The PS side can be used to perform high-end as well as low-end applications and these applications can be run on APU, GPU and RPU. The PS also provides interfaces such as UARTs, SPIs, GPIOs to interact with external devices such as memories, MCUs, etc. The PL of the Zynq Ultrascale+ MPSoC can be programmed to produce digital circuits or Intellectual Properties that matches the application requirements.

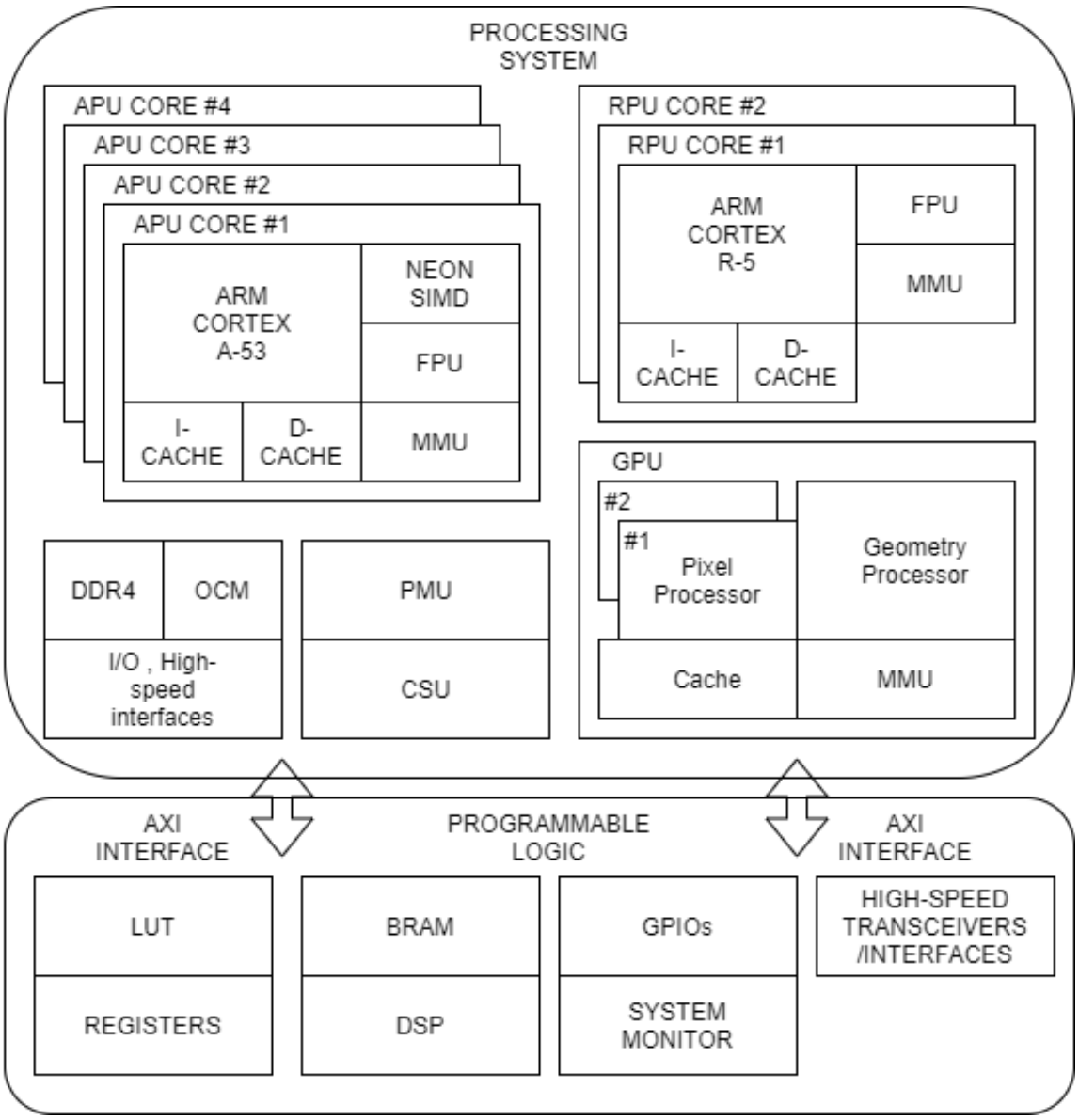

Figure 2.3.1: Zynq Ultrascale+ MPSoC architecture

Component	Quantity
LUTs	274 K
BRAMs	32.1 Mbit
FFs	548 K
Distributed RAMs	8.8 Mbit
DSPs	2520 slices

Table 2.3.1: Number of components present in the Ultrascale+ FPGA

The table 2.3.1 shows the resources available for utilization in the FPGA[31]. These can be used by PS using AXI. Additionally, the FPGA can be programmed to use High-speed interfaces such as PCIe that can be interfaced with external DDR memories, etc. All these features make this an extremely versatile yet powerful board. This thesis

work utilizes the Zynq Ultrascale+ MPSoC in which any CubeSat mission related to communications and data processing can be deployed. This work treats the MPSoC as a black box around which a reliable architecture is designed that ensures proper functioning of the MPSoC and resilience against SEUs and SELs.

2.3.1 Radiation Tests for Zynq Ultrascale+ MPSoC

The Zynq Ultrascale+ MPSoC gained more popularity in the aerospace domain because of its versatility and better radiation tolerance than its predecessors[29]. To validate the radiation tolerance of this device, different radiation tests were conducted as part of research by several research institutes. The following points discuss the performance of Zynq Ultrascale+ MPSoC in different radiation environment:

- Heimstra et al. [10] conducted a proton irradiation test with a 105 MeV proton beam at the TRIUMF proton irradiation facility, Vancouver, British Columbia, Canada. The results of this test showed no occurrence of destructive latch-ups in the Ultrascale+ MPSoC. However, the test recorded various SEUs for different implementations varying from 1 to 35 upsets of SRAM configuration per fluence.
- Similarly, NORSAT3, et al.[23] conducted a proton irradiation test also at the TRIUMF. This test aimed at observing the destructive latch-up events and the upset rates. Two beams of different energy were used. The first test was conducted with a 480 MeV proton beam, which was targeted at individual components of the MPSoC. The second test used a 105 MeV beam to irradiate the whole device to represent the in-orbit scenario. 5 SELs and 4 SELs were observed for 480 MeV beam and 105 MeV beam respectively. All the latch-ups were cleared by power cycling the MPSoC board.
- Heavy ion irradiation tests were performed on the Zynq Ultrascale+ MPSoC by Maximilien Glorieux, et al.[9]. Two different tests, namely standard ion test and ultra high energy test, were carried out at UCL Heavy ion irradiation facility, Belgium and at CERN, respectively. The standard ion test used various heavy-ion beams like Xenon, Neon, Argon, Nickel, etc, with 995 MeV, 238 MeV, 379 MeV, 582 MeV energy, respectively. The Ultra high energy test at CERN employed a 30 GeV/n Xenon beam to test the device. SELs occurred at V_{ccaux}, $V_{ccpsaux}$ and V_{auxio} power rails of the MPSoC when a beam of energy greater than 5.7 MeV cm^2 mg^{-1} was shot at the device during the standard heavy-ion

test. However, no SELs were noted during the Ultra high energy test at normal incidence. The test also showed a very low SEU effect in SRAM.

- Jordan et al. [2] performed a Neutron beam irradiation test at the LANSCE neutron beam facility. The test aimed at observing the number of SEUs occurring at PS and the Programmable Logic (PL) of the MPSoC. The PS remained unaffected during the test; however, 168 and 1302 SEUs were recorded on the OCM and the cache. The PL was affected by SEUs, and they were rectified using scrubbing mechanisms. It was also noted that SEM-IP, which was used to implement the configuration scrubbing in the FPGA, was affected by SEUs as well.

2.3.2 Built-in Fault-Tolerant features of Zynq Ultrascale+

This part of the section discusses the modules that are already available in the Zynq Ultrascale+ as a part of reliable fabrics to tackle SEEs [18]. They are the following:

- Real-time Processing Unit: This unit has a dual-core ARM cortex R5 processor with a single memory that is protected by the Error-Correcting Codes (ECC) mechanism. This ARM-v7R architecture enables us to use redundant application processing in lockstep. This means that the same application can be run in both the processors with a short time offset. This avoids the same error occurring on both processor's application. The outputs coming from both the processors are compared at every clock cycle, and if a mismatch is observed in the outputs, an interrupt is triggered to rectify the error.

- Platform Management Unit: This unit consists of TMR Microblaze processors for the Fault-Tolerant power management system. It is responsible for the initial boot process or pre-configuration stage in the booting process. This unit also allows user custom code to be loaded in the TMR Microblaze processors in order to take advantage of the Fault-Tolerant design given by Xilinx. The RAM memory connected to this design is also protected by the ECC.

- Redundant Coding techniques: The PL's configuration frames, have ECC at every frame to identify and correct up to 4-bit errors. Additionally, CRC codes are generated for the whole configuration bitstream, which accounts for the errors

that are not detected by frame level ECC. The only downside is that CRC codes do not reveal the location of the errors in the configuration bitstream. Hence, the programming logic needs to be completely re-configured again.

- Apart
  from the Fault-Tolerant designs mentioned above, Xilinx also provides access to configuration memory, CRAM, of PL to allow partial/full reconfiguration by the user through Processor Configuration Access Port (PCAP) using RPU/PMU or through ICAP by making use of the SEM-IP core.

2.4 FPGA Configuration Scrubbing

2.4.1 FPGA Configuration

FPGAs are made of umpteen number of configurable logic blocks, interconnect switches, block RAMs, and clock trees. Programming a digital circuit would mean to interconnection these blocks together to bring about a functionality. These programmed configurations are stored in the CRAM of the FPGA in the form of configuration bit-streams. Conceptually, a change in the said configuration bits present in the CRAM would bring about a change in the functionality of the FPGA. However, this is not always the case as only a part of the FPGA resources are used most of the time, and an upset in those regions will not affect the functionality of the FPGA. Xilinx coins the term “Essential bits” for such a category of bits. The critical bits form a subset of the Essential bits, and any change to the critical bit will disrupt the FPGA functionality[13]. The configuration bit-streams are separated and packed into entities called Frames. The configuration memory is arranged in frames as well. Each frame contains the configuration data and ECC syndrome. The ultrascale+ architecture contains 71,260 frames, and each frame consists of 93 words[30].

2.4.2 Scrubbing

To rectify the bit flips in the configuration bits, a mitigation method known as scrubbing is implemented. Configuration scrubbing is a periodic process of replacing the affected configuration bits with the unaffected ones. The general scrubbing circuit contains an interface to the configuration memory, processing logic, and a reference configuration bit-stream. Scrubbing can be sub-divided into two types: Blind and

Readback scrubbing[24].

- Blind Scrubbing: This scrubbing technique periodically replaces the whole Configuration bit-stream without checking if the configuration bits are affected. This method is fast and easy to implement. However, the downside to this method is that there is no upset detection scheme employed in the design, and the bandwidth is wasted due to unwanted data transfers.
- Readback Scrubbing: In this scrubbing technique, the configuration bit-stream is read from the CRAM and compared against a golden copy of the configuration bit-stream. If the comparison results in presence of an error in the frame, only the affected part is overwritten with the contents of the golden copy. This method overcomes the disadvantages of blind scrubbing.

Scrubbing of a configuration memory is not possible without a configuration access port. Xilinx provides two configuration ports, namely, ICAP and PCAP (Available only in the SoC devices). Only one Configuration Access Port can obtain access to the CRAM at any time. The ICAP can be used with the SEM-IP controller or with the user-design in the PL area, whereas the PCAP can be used by the PS in the SoC devices. The user-design can gain access to ICAP using an ICAP primitive provided in the Xilinx design suite. However, this feature is supported only till 7-series FPGAs and not for Ultrascale architectures[18]. Hence scrubbing in Zynq Ultrascale+ MPSoC can be performed using three different methods, they are: JTAG, PCAP and SEM-IP. In this thesis, the focus is mainly on using theSEM-IP.

2.5 Soft-error Mitigation Intellectual property

The SEM-IP controller is an IP core that is offered by Xilinx to handle soft-errors occurring in the system in an efficient manner[28]. The six different modes offered by the SEM-IP are:

- Mitigation and Testing
- Mitigation
- Detect and Testing
- Detect only

- Emulation
- Monitoring only

The Mitigation and testing modes enable SEM-IP error detection, error correction, error classification(if enabled), and error injection. As this thesis work focuses only on mitigation and injecting errors into the design to test, the other modes are not explained. The SEM-IP controller provides two interfaces to interact with it. They are:

- Command Interface: This is a simple interface containing three signals: *command_strobe*, *command_code* and *command_busy*. The *command_code* signal can be used to send commands to the SEM-IP controller to perform tasks like error detection, correction, etc. The *command_strobe* signal is used to deliver the command to the controller by pulsing the signal for one clock cycle. The *command_busy* signal indicates if the controller is ready to accept commands. The command interface can be used with a user-design in the PL part of the FPGA or with the PS through EMIO signals to the FPGA. Conventionally, the status interface is used along with the the command interface to know the controller's active state.
- Monitor Interface: This interface is user-friendly as it uses ASCII characters to send commands to the controller. The controller acknowledges the command with a status report of the controller's action and state, thereby avoiding the use of status interface. The monitor interface can be used directly with the FPGA or the processor system similar to the command interface or using a UART helper block provided along with the SEM-IP example design. In this thesis, the SEM-IP is implemented using the monitor interface.

2.5.1 Different States of the SEM-IP Controller

The SEM-IP controller has seven valid states in mitigation and testing mode as shown in figure 2.5.1

- Initialization: When the FPGA boots up, the controller enters the initialization state where it tries to gain access to the ICAP. After obtaining the grant of the ICAP access, the SEM-IP controller performs two readbacks and completes initialization by shifting to the observation state.

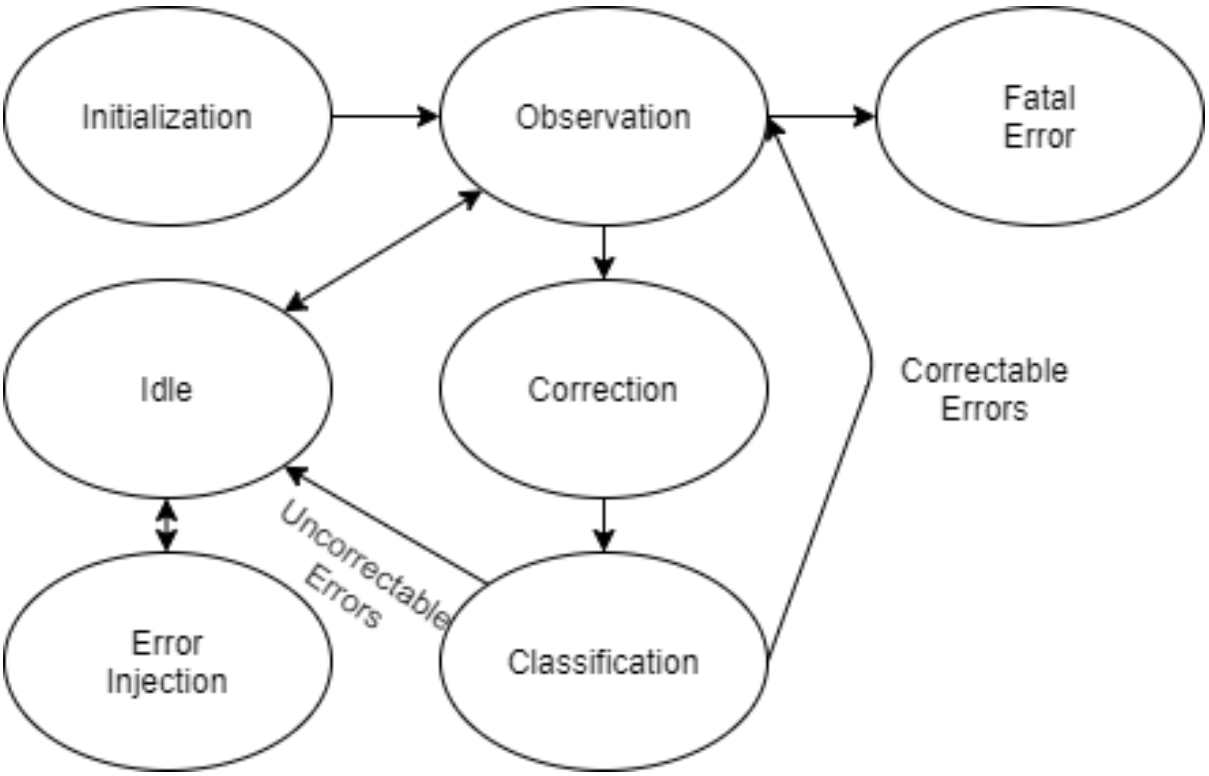

Figure 2.5.1: SEM-IP Controller states

- Observation: This state acts as a watchdog by continuously scanning for errors in the CRAM. If an error is detected, then the controller gathers the necessary data for rectification before transitioning into the correction state. Additionally, the observation state accepts two commands, either to move the controller to the idle state in order to perform other actions of the controller or to provide a status report of the controller.

- Correction: The controller tries to correct the error detected in the CRAM configuration bits in this state. The controller attempts to fix the error with the collected information. Once this attempt is executed, the controller reports either the error that was identified and corrected or the uncorrectable errors.

- Classification: The controller goes through the classification state even if this state is not enabled. If the error classification is disabled, then all errors are classified as essential errors unconditionally. Else, the uncorrectable errors are classified as essential, and the rest are classified as non-essential. The FPGA must be reconfigured in case of an uncorrectable error. The SEM-IP then generates the request of essential bits through the fetch interface connected to an external memory. The controller either changes to the idle state if it is uncorrectable or to

the observation to continue error detection.

- Idle: This state retains the controller in the idle mode and waits for the command to perform error correction, error detection, error injection, query a status report from the controller, or perform a software reset.
- Injection: As the name indicates, the controller injects error into the frame specified by the linear frame address given along with the command. In order to inject error, the following command is used:
  N {11-digit hex value} <carriage return>
  For Ultrascale+ devices, the 11-digit hex value is split into the following, as shown in figure 2.5.2:

11	10	9	8	7	6	5	4	3	2	1
1100	0000	0000	ssLL	LLLL	LLLL	LLLL	LLLL	wwww	wwwb	bbbb

Figure 2.5.2: 11-bit hexadecimal value for error injection

The table 2.5.1 describes the hexadecimal value entered in order to inject errors.

Notation	Name	No. of bits
s	Hardware SLR number	2
L	Linear frame address	16
w	Word in the frame	7
b	Bit location in the word	5

Table 2.5.1: 11-digit hexadecimal value for error injection

- Fatal: The controller runs into fatal state when the functionality of the SEM-IP controller is compromised in which case the FPGA needs to reconfigured. Until then, the controller retains this state.

2.6 Space Packet Protocol

Space packet protocol [5] is used for transmission of data in ground-space communication or space-space communication. This protocol is based on the OSI reference model. The sender is responsible for generating the space packet with the task required to be performed by the receiver along with the essential data. The receiver decodes the packet and performs the task accordingly. The receiver should also send an appropriate acknowledgment. The Space packet is a variable frame that can be

extended up to 64 kB. The protocol can be tailored according to the user application. The standard space packet frame is shown in figure 2.6.1:

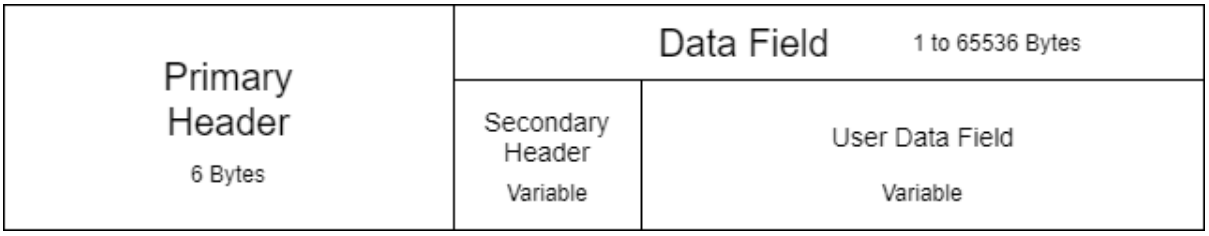

Figure 2.6.1: Space Packet

The space packet frame can be viewed as a collection of a header field and a data field.

Primary Header Field The primary header field can be further split into four categories, as shown in Table 2.6.1:

Category	Sub-division	No. of bits
Packet version number	NA	3
Packet identification	Packet type Secondary header flag Application ID	1 1 11
Packet sequence control	Sequence flags Sequence count	2 14
Packet data length	NA	16

Table 2.6.1: Space packet primary header

- Packet version number: It comprises 3-bits, and its value is 000 by default, signifying that the current packet is a space packet protocol. The version number is reserved for future references to integrate other packet structures.
- Packet identification: Bits 3-15 of the primary frame header makes up the packet identification field. As the name suggests, it is used to identify the type of space packet and other necessary information such as application ID and secondary header presence. Bit 3 identifies the packet as either a telecommand(request) or a telemetry(report/acknowledgement). The next bit indicates the presence of a secondary header. Followed by the secondary header flag, lies the 11-bit application ID field that specifies the different tasks carried out in the space mission. The user can design the application ID field according to their respective applications. However, the protocols demand that the user refrains from using certain reserved application IDs.

- Packet sequence control: This field is again divided into packet sequence flags and packet sequence count, which is useful in segmenting large data. The packet sequence flag indicates that the current packet is either the first part of the segment, the last part of the segment, a continuous segment or the whole packet is unsegmented.
- Packet data length: This 16-bits field starting from 32^{nd} bit of primary header field specifies the length of the data field. The maximum length of the data can be 64 kB.

Data Field The data field is divided into the secondary header field and the user data field. If the secondary header flags indicate the secondary header field's presence, only then can this field be used/defined by the user, else they will belong to the user data field. Depending on the data length assigned in the data length field of the primary header, the user can calculate the number of data to be read from the user data field.

2.7 Cyclic Redundancy Check

Cyclic Redundancy Checks[21] are error-detecting codes that are useful for detecting and correcting single and multi-bit errors. They overcome the disadvantages of Single-Error Correction and Double-Error Detection (SECDED) codes. In order to calculate CRC checksum, a generator polynomial is required. For an n-bit CRC, the length of a generator polynomial is n+1 bits. The CRC checksum is obtained by dividing the data with a generator polynomial in a long division. The quotient of the long division is discarded, and the remainder is used as the checksum[22].
The CRC checksum is calculated for data of different sizes depending on the number of CRC bits used in the application. For example, a 16-bit CRC can hold checksum for 64 kB of data and a 32-bit CRC can easily hold checksum for 4 GB of data. In this project 2 standard CRCs are used as depicted in tables 2.7.1 and 2.7.2.

Category	Value
CRC name	CRC-32
Width	32 bits
Polynomial	`0x04C11DB7`
Initial remainder	`0xFFFFFFFF`
Final XOR value	`0xFFFFFFFF`
Parity	Odd

Table 2.7.1: CRC-32 description

Category	Value
CRC name	CRC-16
Width	16 bits
Polynomial	`0x1021`
Initial remainder	`0xFFFF`
Final XOR value	`0x0000`
Parity	Even

Table 2.7.2: CRC-16 description

2.8 Radiation Hardened MCU

The Vorago PEB1-VA41630 is a radiation-hardened MCU[19] that houses a 32-bit ARM Cortex-M4 single precision floating point unit core. The Total Ionization Dose of this MCU is over 200 krad. It provides latch-up immunity with Linear Energy Transfer (LET) greater than 110 MeV cm^2 mg^{-1} and a SEE of less than 1×10^{-15} errors/bit-day with EDAC enabled. This radiation performance makes this device worthy to be used in aerospace applications. The device can functions over the temperature range of -55 to 125 °C.

The following are the peripherals available with the MCU:

- 3 UART Serial Interface
- 3 SPI
- 3 I2C
- 104 GPIOs
- Ethernet
- 2 CAN bus
- Spacewire

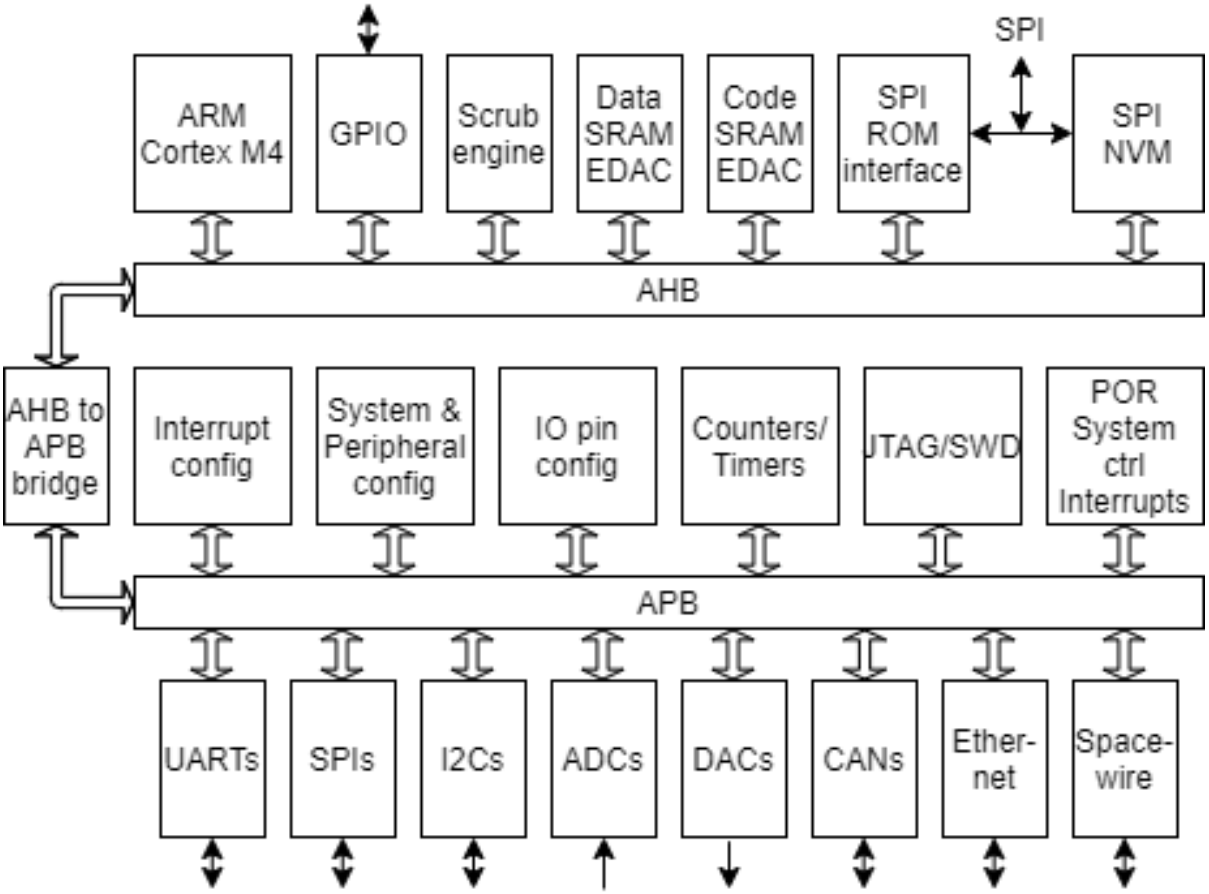

Figure 2.8.1: Rad-Hard Vorago MCU

- 8 12-bit ADC
- 2 12-bit DAC
- 4-channel DMA
- Temperature sensor.
- 24 32-bit timers/counters

Figure 2.8.1 depicts the architecture of the radiation-hardened MCU. The MCU works at a 100 MHz frequency and has a set of memory devices, for instance, 256 kB FRAM, 256 kB SRAM for code and 64 kB for data memory. Both the code and data memory have EDAC and memory scrubbing support. The wide range of features and peripherals offered by this MCU make it a very good candidate to be a Supervisor for the Zynq Ultrascale+ system.

2.9 Memories

Memories are a rudimentary and an essential component in a CubeSat hardware architecture. They are responsible for data logging of important mission data, storage of executable binaries of MCUs, FPGAs, or the MPSoCs that carries out the application necessary to run the CubeSat mission. However, despite being fundamental entities of such crucial tasks, they are susceptible to radiations[16]. Since memories are cell blocks that store binary values, 1 and 0, bit-flips are a regular occurrence. The probability of occurrence of SEE is even more in the case of high-density memory units. This work uses NVMs such as NOR Flash and MRAM. For more information regarding NOR Flash, MRAM, and how they work, refer to [4] and [11]. Research paper [12], suggests that NOR Flash is vulnerable to SEUs. However, the radiation tests conducted by [1], shows that MRAM is immune to SEUs, Multi-Bit Upset (MBU)s and SELs.

Category	Value
Memory Type	NOR Flash
Memory Size	32 MB
Interface	PMOD
Connection	SPI
Voltage	3.3 V
Current	20 mA
Max. Frequency	100 MHz

Table 2.9.1: NOR Flash memory Specifications

Category	Value
Memory Type	MRAM Flash
Memory Size	0.5 MB
Connection	SPI
Voltage	3.3 V
Max. Current (at Max. Frequency)	10 mA (read), 27 mA (write)
Max. Frequency	40 MHz

Table 2.9.2: MRAM Specifications

The Tables 2.9.1, 2.9.2 describes the specifications of the memories used in this project.

Chapter 3

Cubesat Hardware Architecture

This section discusses the architectural design of CubeSat in detail. First, the overview of the architecture is presented. Then, the individual components are discussed along with their respective interfaces and functionalities.

3.1 Overview

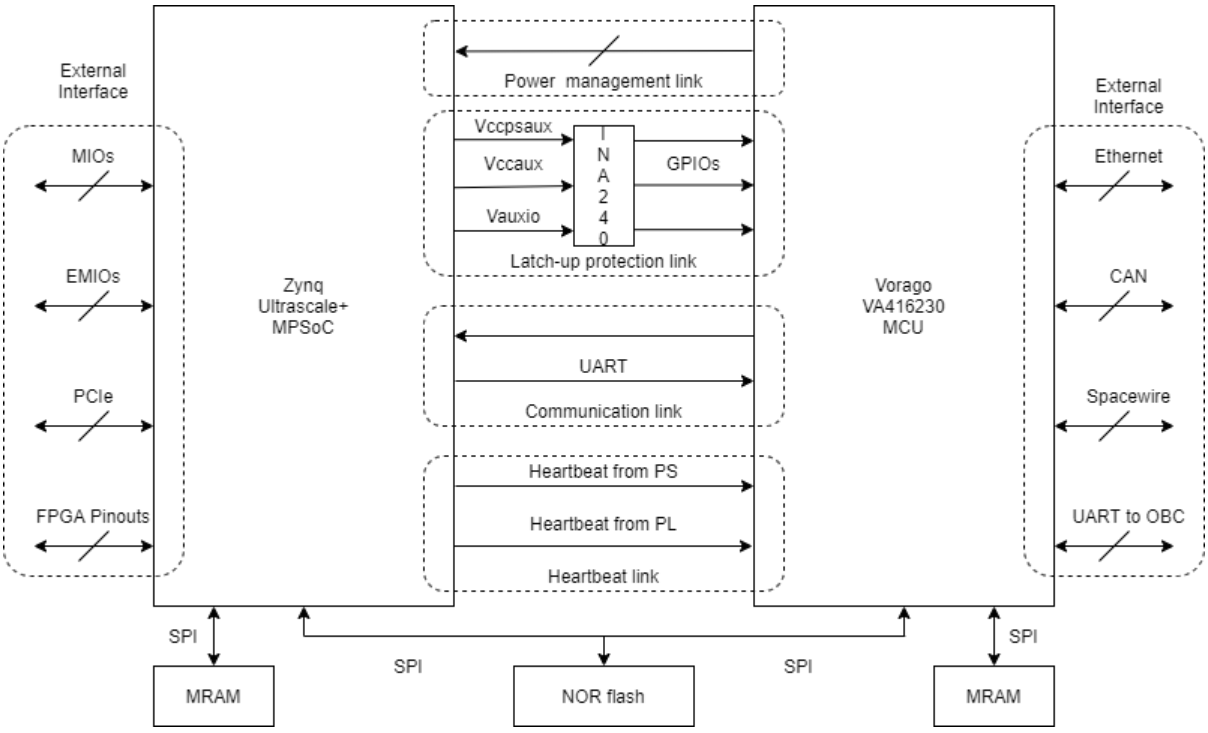

Figure 3.1.1: Cubesat hardware architecture

As shown in figure 3.1.1, the COTS MPSoC is the data processing center, where the user-developed application is executed in the processor system and/or the programmable logic section. This architecture is not dependent on any application making it adaptable. Since, the Zynq Ultrascale+ is a very versatile device, its peripherals such as UART, SPI, PCIe, etc can be accessed using MIO or EMIO pins by the user according to the demands of their respective application. The Supervisor is responsible for the safe operation of the COTS MPSoC. It also acts as a medium between the rest of the hardware present in the satellite and the MPSoC. The architecture is such that the processing board and the Supervisor system can be incorporated with any OBC with a UART serial communication. The working of OBC is out of the scope of this paper and is considered a black box.

3.2 Supervisor

The Supervisor is a radiation-hardened MCU, Vorago VA41630, based on an ARM Cortex-M4 core. Being radiation-hardened, the Supervisor is not affected by radiation, making it a safe island for the COTS MPSoC, which is susceptible to radiation. Therefore, the Supervisor treats the COTS MPSoC as a black box and tries to recover it in case of disruption during its operation. The Supervisor also assumes responsibility for its communications in, out, and within the CubeSat. The interfaces and the functionalities of the Supervisor are discussed in detail in the following sections.

3.2.1 Interfaces

The Supervisor is connected to the COTS MPSoC, NOR Flash memory, MRAM memory and the OBC of the CubeSat. The following describes in detail about each interface that is connected to the Supervisor:

- OBC:
  The OBC is connected to the Supervisor via UART serial communication. The OBC is responsible for sending telecommands with the help of space packets to the Supervisor in order to perform several tasks that will be discussed later in detail. The Supervisor sends a Telemetry space packet back to the OBC as an acknowledgment, a status report, or a data transfer.

- Zynq Ultrascale+ MPSoC:
  The interface between the ultrascale+ and the Supervisor is divided into four links, such as:

  – Communication link:
    Similar to the OBC-Supervisor communication link, the communication between the Supervisor and the Zynq ultrascale+ is through a UART serial communication. The Supervisor forwards the space packets to the MPSoC if the packet is a telecommand meant to be sent to the processing board or from the MPSoC to OBC if the packet is telemetry.

  – Latch-up Protection link:
    The Supervisor monitors the Zynq Ultrascale+ MPSoC for SEL by reading the current values of the power rails using current amplifying sensor INA240 that is connected to the ADCs of the Supervisor. The latch-up detection circuit operation is explained in detail in the functions section 3.2.2.

  – Power Management link:
    The GPIOs of the Supervisor are connected to the essential power rails that are necessary for the power-on/off sequencing of the MPSoC. This is an essential link when the processing board is said to be power-cycled or reset.

  – Watchdog link:
    Two of the GPIOs of the Supervisor are connected to the PS and the PL of the MPSoC, respectively. This is done in order to receive a heartbeat signal from either of them as part of the health monitoring system. The working will be explained in detail in the functions section 3.2.2.

- NOR Flash memory:
  The Quad-SPI NOR Flash boot memory of the Zynq Ultrascale+ MPSoC is also accessed by Supervisor through SPI in order to perform functions such as memory scrubbing, software, and firmware update for the MPSoC. A detailed insight on this topic is given in the functions section 3.2.2 of the Supervisor.

- MRAM memory:
  The MRAM memory is connected to one of the 3 available SPI ports of the Vorago VA41630 and is used to store state information of the Supervisor. For instance,

information of the most recent successful image update, previous successful transfer of data, health status of the processing board, image information in the memory.

3.2.2 Functions

The following section discusses about the functionalities of the Supervisor system. As aforementioned, the Supervisor is responsible for a guaranteed operation of the COTS MPSoC for which it must shoulder the following essential tasks:

- Communication
- Latch-up protection circuit
- Watchdog
- Memory scrubbing
- Power management
- Software/Firmware update
- Data Logging

Communication

The radiation-hardened MCU establishes a safe link between the OBC and the processing board, that is, Zynq Ultrascale+ MPSoC. The communication link is made of UART serial communication with a speed of 115200 Bauds and a default 8N1 configuration setting. The messages are sent or received in form of packets conforming to the space protocol, and they can be distinguished as telecommands and telemetries. The messages originating from the OBC are telecommands, while the ones starting from the processing board or the Supervisor are telemetries. Only the OBC can send telecommands to both Supervisor and the processing board, and it has to wait for an acknowledgment to be received before sending another telecommand. In other words, only one command can be processed at a given time. The Supervisor acts as a router of the space packets since it decodes the space packets received. Depending on the application ID, it either forwards the packets or retains them and executes the requested task. In order to make the communication reliable, a 16-bit CRC is attached at the end of every data field in the space packet. This is useful to identify errors during

transmission and reception of data. The loss of packets is identified by the sequence count number in the space packet frame. Before every transaction, the Supervisor should get the information regarding the number of packets it might be receiving. If the sequence number does not match with the initial number sent after the transaction, a loss of a packet is identified, and a proper acknowledgment is sent. If any anomaly is detected by CRC or sequence count check, the Supervisor initiates a request to resend the data.

Latch-up protection

One of the Supervisor's critical and rudimentary tasks is to constantly monitor the processing board for the occurrence of SEL. According to the radiations tests, the power rails V_{ccaux}, $V_{ccpsaux}$ and V_{ccio} are recorded with latch-up occurrences. Hence the aforementioned in 3.2.1 power rails are connected indirectly to the ADCs of the Supervisor through a current sensing amplifier, `INA240`. The output of this current sensing amplifier is an analog current value which is fed to the ADC in the Supervisor. The converted digital values of the current from the power rails are compared against a threshold value. If the current value exceeds the threshold, then it signifies that a latch-up has occurred, and a power cycle for MPSoC is initiated in order to clear the latch-up.

Watchdog

The Supervisor acts as a watchdog for the processing board by constantly waiting for a heartbeat signal from the PS and the PL of the Zynq Ultrascale+ Multi-Processor System on Chip (MPSoC). A heartbeat is a pulse signal generated by a system in order to notify that it is still executing its application without crashing. In this architecture, the application designer holds the responsibility of generating a heartbeat signal out of the PS and PL along with their application development. This will make the heartbeat generation dependent on the application execution, and in which case an application stops executing, then a heartbeat will not be generated. On the other hand, the Supervisor waits for the 2 heartbeat signals from the MPSoC for a fixed amount of time. In case the heartbeat signals are not sensed by the Supervisor before this time, then the Supervisor triggers a system reset for the MPSoC.

Memory Scrubbing

The Supervisor is also connected to a Non-Volatile NOR Flash memory using SPI, as shown in the overview of the architecture3.1.1. This NVM is a primary boot memory for the MPSoC, which houses the first stage boot-loader, second stage boot-loader, bitstream for the FPGA, and some other necessary binary files. Single event upset inside this memory will lead to an incorrect application being loaded and executed during the boot sequence. To avoid this, the Supervisor performs readback memory scrubbing on this memory. A CRC check is done to ensure no errors are present in the memory. A request for data restoration is made if an error is detected.

Power management

The power management task is responsible for conducting power cycles and reset for the MPSoC. The reset and power-on/off line of the processing board are controlled by the GPIO lines of the Supervisor. Additionally, certain power rails that are necessary for power-on and power-off sequence of the zynq ultrascale+ MPSoC are connected to the GPIOs of the Supervisor as well. The processing board follows a power-on and power-off sequence to safely operate different power domains.

Software/Firmware update

This is an advantageous task provided by the Supervisor wherein a software or a firmware update can be made during the mission. In this task, the Supervisor obtains the binary files provided by the OBC in sequence and stores them in the primary NVM. One other sub-task that the Supervisor carries out is the verification of the updated image in response to a telecommand by the OBC. This is done by making use of the CRC code. A CRC check is done after a readback of the updated image, and an acknowledgment is provided to OBC regarding the status of the update.

Data Logging

With many functionalities being processed in the Supervisor, it is necessary to save the state of the progress of different functions from time to time. This would help the Supervisor to keep track of all the functions or track-back to the safe state when there is a discrepancy in the execution of any functionality. This is done by storing the state data in the MRAM. The MRAM does not face the issue of SEUs and hence can be used

to store such important data. Additionally, the Supervisor also delivers a status report Telemetry to the OBC of the state of the system.

3.3 Processing Board

The Zynq Ultrascale+ MPSoC board is the heart of the data processing center since the application developed for the CubeSat mission resides in this COTS MPSoC. The following discusses the interfaces and the functionalities of the processing board in detail.

3.3.1 Interfaces

The processing board is connected to the Supervisor. Other than being connected to Supervisor, the processing board is connected to 2 NVMs:

- NOR Flash:
  As aforementioned in the Supervisor section 3.2, the NOR Flash memory is a shared memory interface between the Supervisor and the processing board. The Supervisor performs memory readback scrubbing to mitigate SEUs. In contrast, the Zynq Ultrascale+ utilizes this NOR Flash as its boot memory. This memory houses the boot image and linux kernel image, which are in charge of starting up the processing board.

- MRAM:
  Similar to the MRAM present in the Supervisor, the MRAM interfaced with the processing board hold valuable mission data values.

3.3.2 Functions

The processing board is meant for payload application in which different types of applications can be deployed with the resources available from the Zynq Ultrascale+ MPSoC. Apart from carrying out mission-related applications and generating heartbeat signals along with the application, the processing board is responsible for two main operations. They are communication and configuration scrubbing. The following paragraph describes in detail about the tasks:

Communication

The processing board receives commands from OBC either to perform tasks or to request a data or a status update from the MPSoC. Similarly, the MPSoC should reply to the OBC with an acknowledgment for the completed task. This acknowledgment can be in the form of data or status. As mentioned before in 3.2, the Supervisor acts as a middle ground for the OBC and the processing board, where the packets are routed to each other reliably. Hence, out of four APU cores, one of the ARM Cortex-A53 cores of the processing board is connected to the Supervisor using to UART communication. The UART is configured at 115 200 Bauds and 8N1 setting similar to UART communication setting between Supervisor and OBC. The APU is responsible for receiving space packets and parsing the data to extract the necessary information such as Application ID, Sequence flags/counts, and a 16-bit CRC. The Application ID is used to identify the operation to be performed. The sequence flag/count and CRC are used to maintain robust communication by detecting packet loss and errors.

Programmable Logic Configuration Scrubbing

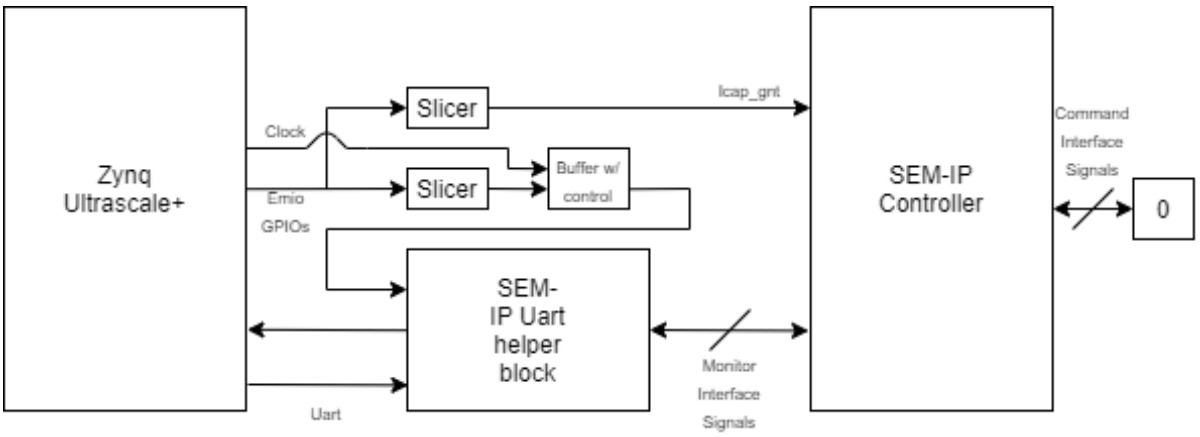

Figure 3.3.1: Configuration scrubbing block diagram

The processing board internally performs the configuration scrubbing on the Programmable Logic with the help of the Processor System and the SEM-IP controller. The SEM-IP is instantiated inside the FPGA and connected to the PS through a UART helper block, as shown in the figure 3.3.1. The PS can send commands to the SEM-IP periodically to get the status of the configuration bits. The SEM-IP stays vigilant and looks for the SEUs and MBUs inside the configuration bit-stream. In case it encounters SEUs or an MBU, it corrects them immediately and sends a report to the user. If it is uncorrectable, the FPGA need to be reprogrammed.

Chapter 4

Implementations

In this chapter, the implementation of the CubeSat is discussed. The detailed description of the software architecture of the proposed design is outside the scope of this project. Firstly, a methodology of a specific implementation is shown with the help of a flow chart, and then it is described. Secondly, the hardware and software requirements are mentioned for each implemented design. Hardware and software limitations are discussed wherever applicable for the implementations.

4.1 Communication

The Supervisor handles two separate communication links, the communication with OBC and the communication with the processing board. As mentioned before in 3.2.2, communication of commands and data takes place in the form of the space packet protocol via UART serial communication. In this implementation, the communication link is set between OBC, Supervisor, and the processing board. The flow chart will give details on the approach followed to establish reliable communication.

The flow chart shown in 4.1.1 depicts the process of management of communication between OBC and the Supervisor. The management of communications from the Processing board is very similar to that of the OBC. It which can be obtained by swapping the roles of OBC and processing board in the flow chart 4.1.1. The UART peripheral receives data in bytes, and a total of 1024 B are sent or received during communication at any instant. The reason for this is because of the choice of size of the space packet adopted for the design. Once the data is received, the Supervisor

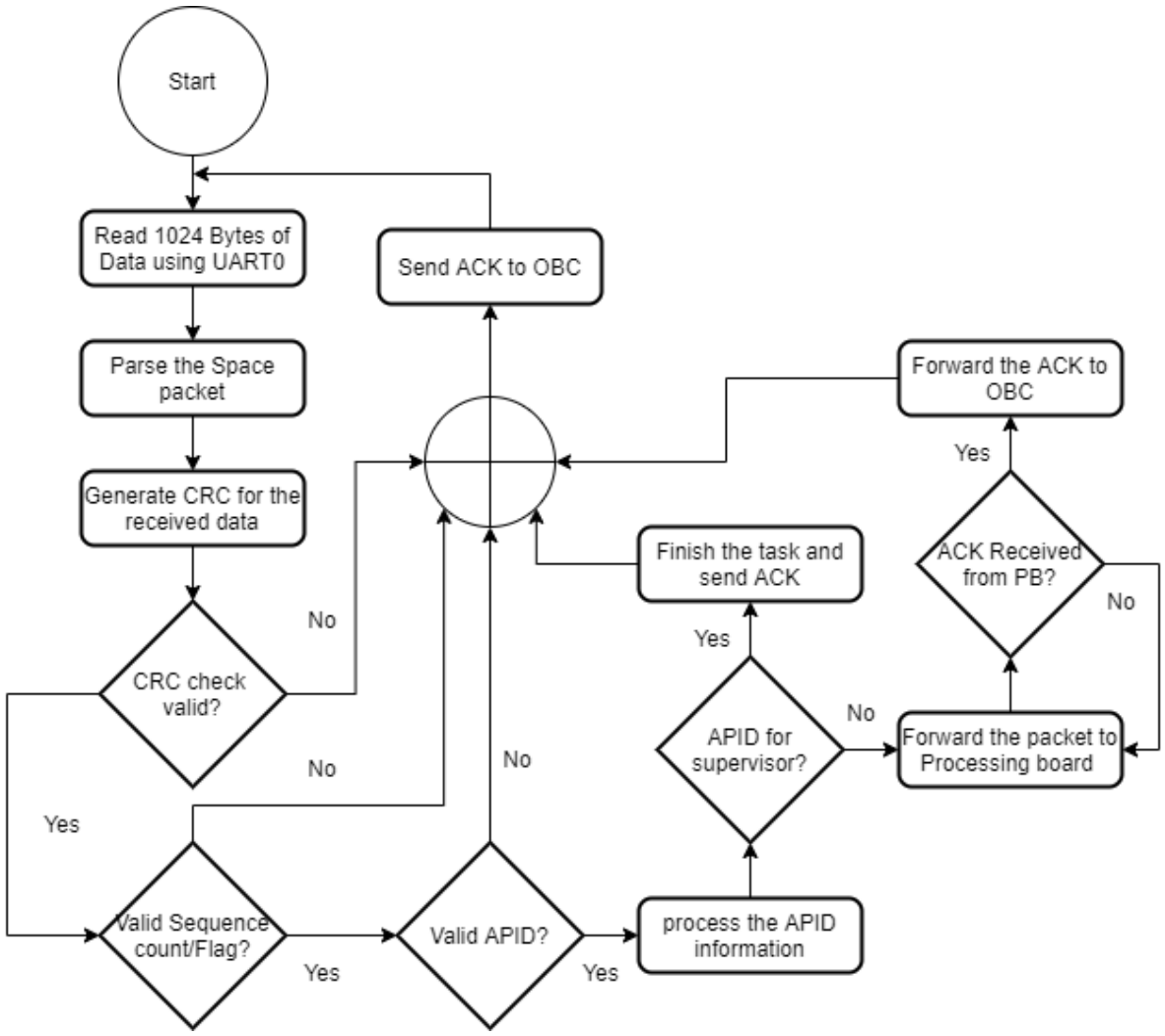

Figure 4.1.1: Communication flow chart

arranges the data in the form of the space packet protocol to extract information such as packet identification, packet sequence control, data length, and CRC. With the packet identification information, the Supervisor will identify the sender and the operation to be performed by the Supervisor. The Supervisor generates a CRC of the received data and compares it with the extracted CRC. In case the CRC is mismatched, an acknowledgment is sent to the source to resend the packet again. Then the sequence count and sequence flags are checked for every transmission to identify the packet loss. In case a mismatch occurs, an acknowledgment is sent to the source to resend the lost packet. Finally, the application ID is checked for valid tasks that can be performed by the Supervisor or processing board. If the task was meant for the processing board, the Supervisor forwards the packet promptly to the processing board through a separate UART channel. The Supervisor waits for the acknowledgment from the processing board and forwards the acknowledgment back to OBC after receiving it.

4.1.1 Hardware/Software Requirements

The table 4.1.1 describes the necessary hardware/software requirements for the communication implementation. In order to make the communication possible, 2 UARTs serial communications are required for the Supervisor to interact with the OBC and the processing board. The UART settings used in this scenario are standard 115k Baud with 8 data bits, 1 stop bit, and no parity bits. The UARTs are used in an interrupt mode. A library for space packet was created and used apart from the standard c libraries and board support package of the Micro-Controller Unit.

Category	Description
Hardware	Vorago VA41630 Zynq Ultrascale+ MPSoC Linux PC emaulating OBC
Connection	2 UARTS USB to UART convertor
Software	Keil uVision MDK v5 Vivado 2018.3 Cent OS 7
Terminal	Putty Minicom
Custom Libraries	Space packet protocol
CRC	16-bit

Table 4.1.1: Hardware/Software requirements for communication

4.1.2 Hardware/Software Limitations

The primary bottleneck of this communication is memory usage. The data memory available in the Vorago VA41630 is 64 kB. The maximum size of the space packet that can be used is 64 kB. Considering the various tasks stated in this report that are deployed in the Supervisor, only a low percentage of memory is available for space packet protocol. Hence, a ballpark of roughly 1 kB is allocated for space packet protocol. Another limitation is the speed of communication. The UARTs are implemented at 115K Bauds for prototyping purposes. However, this limitation can be overcome by increasing the baud rate of the UARTs until its supported hardware limit, which is 1 Mega Baud for VA41630 MCU.

4.2 Watchdog

The watchdog utilizes GPIOs from the VA41630 board to monitor the heartbeat signals from the processing board. The flow chart in figure 4.2.1 provides a better understanding of the implementation.

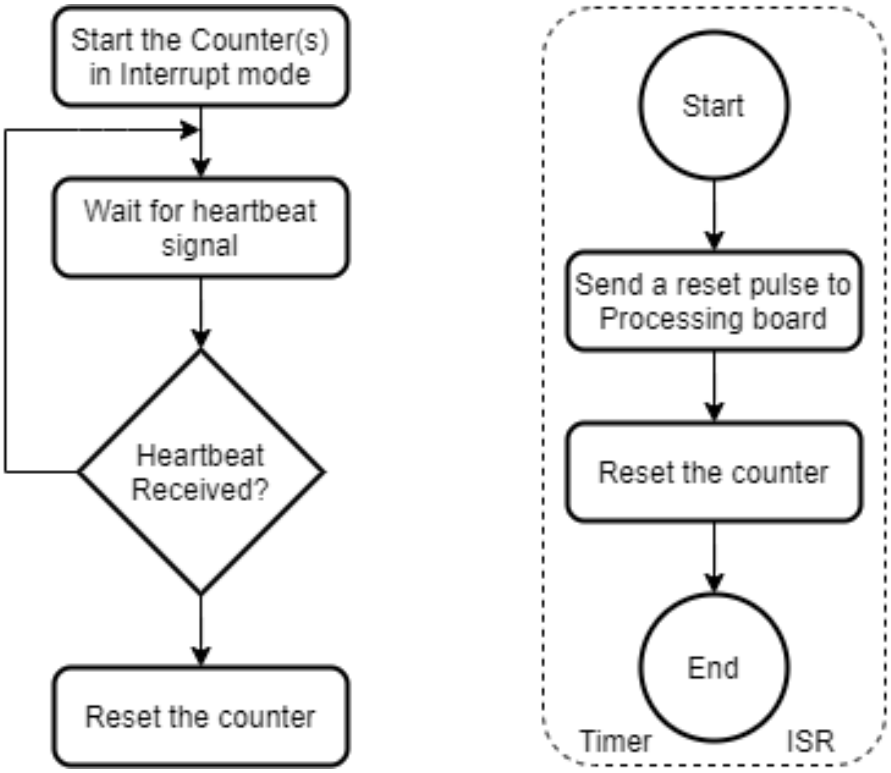

Figure 4.2.1: Watchdog flow chart

The depicted flow chart is used to explain the watchdog monitoring for both Processor System and Programmable Logic. Two 32-bit timers are invoked in interrupt mode to count down from 60 seconds. Two of the GPIOs from the Supervisor waits for pulsed signal (heartbeat) from both PS and PL, respectively. If the GPIO(s) receives any heartbeat pulse from their respective system, then an interrupt is initiated. This Interrupt Service Routine (ISR) resets the counter, and the counter again starts the down counting from 60 seconds. This process repeats itself until there are no heartbeat signals available to the GPIOs. In this case, the counter/timer expires after down counting from 60 seconds. The whole task is repeated time and again to ensure that the applications in the processing board are always under execution.

4.2.1 Hardware/Software Requirements

The table 4.2.1 provides details on the necessary things to perform this implementation. Three GPIO pins are required, two to sense heartbeat signals from PS and PL parts of the Zynq Ultrascale+ MPSoC and one to reset it. Also, two timers/counters are required to countdown a period of 60 seconds to measure the period of inactivity.

Category	Description
Hardware	Vorago VA41630 Zynq Ultrascale+ MPSoC
Peripherals	2 Counters/timers 3 GPIOs
Software	Keil uVision MDK v5 Vivado 2018.3
Terminal	Minicom

Table 4.2.1: Hardware/Software requirements for watchdog

4.3 Memory scrubbing

Memory readback scrubbing is performed when the NOR-Flash memory is not accessed by any device. The memory scrubbing flow chart is shown in figure 4.3.1.

The Supervisor performs readback scrubbing on the NOR flash memory by reading 1024 B off the memory. A 32-bit CRC is generated with the 1024 B of data read from the memory with an initial remainder of `0xFFFFFFFF`. The generated CRC is stored to be passed as an initial remainder for generating the CRC of the next set of 1024 B of data. Once the CRC is obtained, the address is incremented to read the next batch of data. This process is repeated until the whole memory is scrubbed. The final CRC value generated by reading the last 1024 B of data is compared against a golden CRC value of the memory. If they are found to be different, a request is sent to the OBC to re-write the whole memory. Otherwise, the Supervisor repeats the readback scrubbing process.

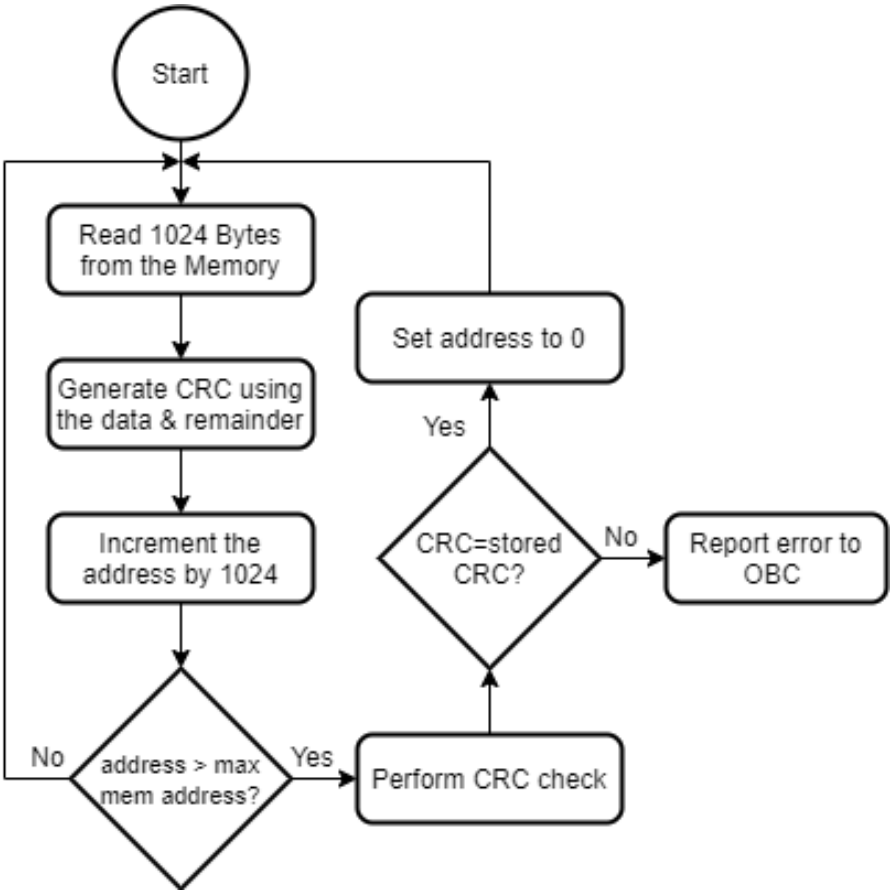

Figure 4.3.1: Memory scrubbing flow chart

4.3.1 Hardware/Software requirements

The design requires a NOR Flash memory and a SPI peripheral from Vorago VA41630 to access the NOR-Flash. A driver has to be designed specifically for NOR Flash in order to access it. A CRC generation function is required to check for errors in the NVM memory. Refer to 4.3.1 for more details.

Category	Description
Hardware	Vorago VA41630 N25Q256A NOR flash
Peripherals	SPI
Software	Keil uVision MDK v5
Terminal	Segger RTT Viewer
CRC	32-bits
Custom library	NOR flash driver CRC generator

Table 4.3.1: Hardware/Software requirements for Memory scrubbing

4.3.2 Hardware/Software Limitations

The speed of the memory scrubbing is limited by the read speed of NOR Flash and also the speed at which SPI operates. Additionally, the read of the whole memory cannot be done at once because of the lack of memory available to store a large amount of data. Hence, only 1024 B are being read at once.

4.4 Software/Firmware Update

The software/firmware update and verification implementation make use of binary images of different sizes to be updated and verified in the NOR flash memory. The Supervisor waits for the appropriate application ID to perform software/firmware update or verification. The flow chart in figure 4.4.1 describes the update routine performed by the Supervisor. The flow chart for the verification task was skipped because the process is very similar to memory scrubbing wherein the scrubbing is done only for the location where the image was updated, instead of scrubbing the whole memory.

The Supervisor enters the update task with data, data length, and end of the update flag as parameters. The data is split into pages using the data length parameter to be stored in the memory since the writing operation is done page by page. The Supervisor is also responsible for address and page handling. The address handling is done in order to keep track of the address location each time an image data is written to the memory. And the page handling is done to erase a new sub-sector before writing any data into it. A page write is performed on the data that was split into pages followed by which the remaining data (that could not form a page) was stored, after address handling and page handling. This process is repeated until the Supervisor receives the final packet of data, which is indicated by ‘end of the update flag’.

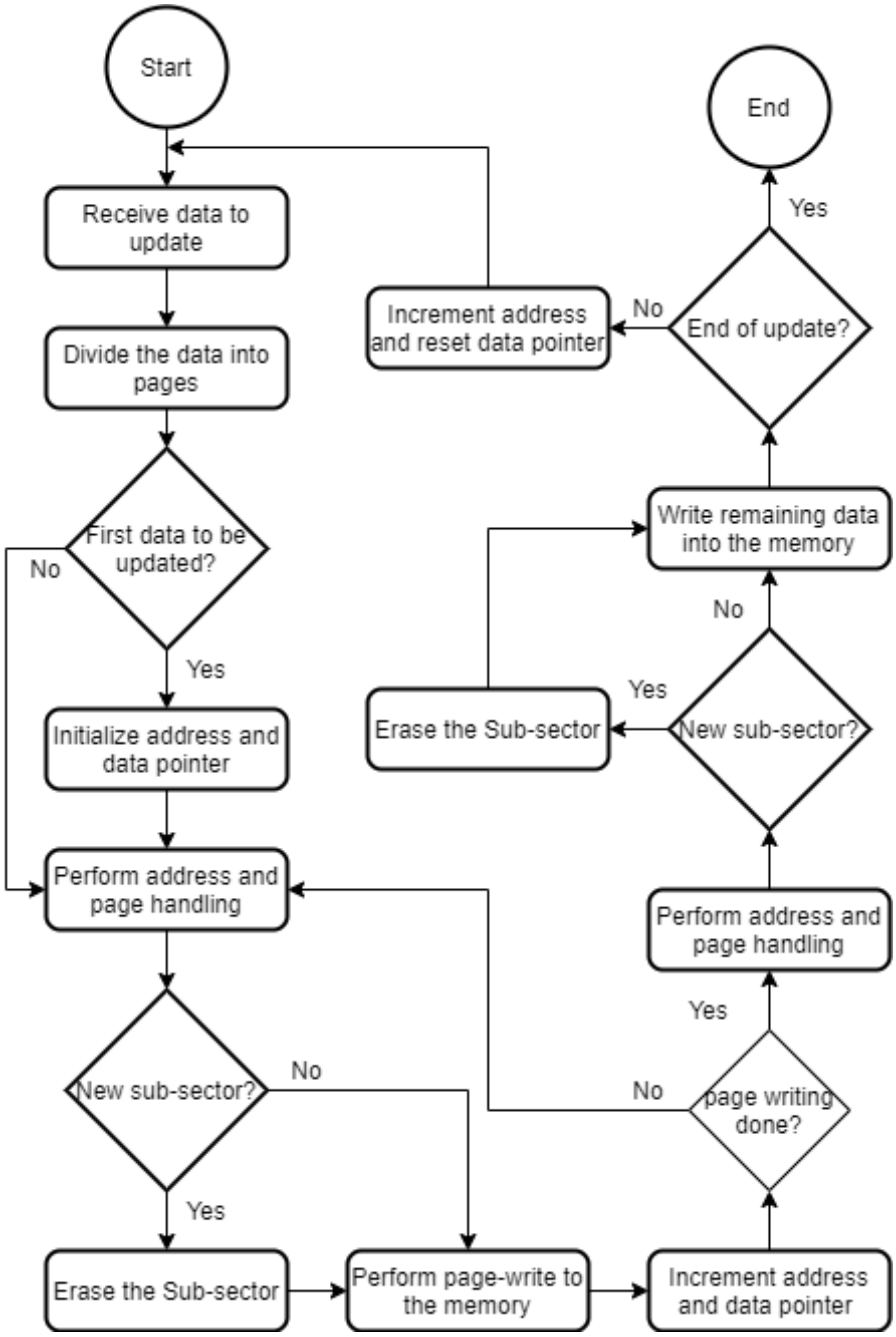

Figure 4.4.1: Software/Firmware update flow chart

4.4.1 Hardware/Software requirements

UART and SPI peripherals are used to communicate with the OBC and NOR flash, respectively. The UART is using 115 200 Bauds and the SPI runs at 25 MHz clock speed. Two custom made drivers, a NOR flash driver and a CRC generator, are used in this implementation. Additionally, Different binary file sizes mentioned in the table 4.4.1 are used to test the update and verification functionality of the Supervisor.

Category	Description
Hardware	Vorago VA41630 Linux PC emulating OBC N25Q256A NOR flash
Peripherals	SPI UART
Software	Keil uVision MDK v5 Cent OS 7
Terminal	Segger RTT Viewer Minicom
CRC	32-bits
Custom library	NOR flash driver CRC generator
Binary file sizes	1KB, 4KB, 13KB, 24KB, 1MB

Table 4.4.1: Hardware/Software requirements for update and verification

4.5 FPGA Configuration Scrubbing

This implementation demonstrates the error detection and correction capabilities of the SEM-IP by performing error injections into the configuration memory. In order to visualize the detection and correction of errors performed by the SEM-IP controller, it is instantiated in mitigation and testing mode. This enables us to inject errors into the CRAM of the FPGA. The following flow chart will brief about the interactions between PS, specifically cortex R5 and SEM-IP controller:

The figure 4.5.1 portrays the working of the SEM-IP.When the processing board is switched on, the SEM-IP is initialized by transferring the control of configuration access port from PCAP to ICAP. This is done by enabling bit zero of PCAP control register. The SEM-IP will be completely initialized when the ICAP access is granted to the SEM-IP, and the clock is supplied. The SEM-IP enters the observation state after initialization. As mentioned earlier in 2.5.1, in this state the SEM-IP is performing error detection and correction of the CRAM. The SEM-IP is used in monitor interface mode. In order to navigate the SEM-IP to different states, ASCII commands are sent through the UART. Table 4.5.1 gives a list of commands to make the SEM-IP transition to different states. For instance, to shift the SEM-IP from observation to Idle state, the ASCII value of uppercase I is sent via the UART connecting Cortex-R5 core and SEM-IP controller. Similarly, to peek into the values of a frame, the controller must be in the Idle state, followed by the transfer of the command sequence `Q <11-digit hex value>`.

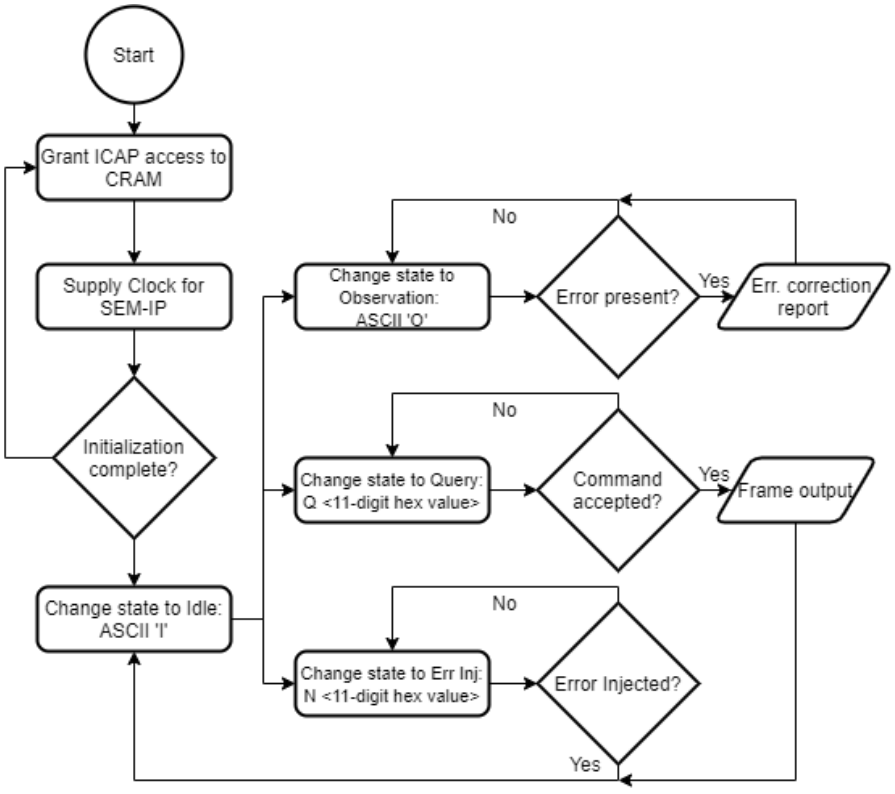

Figure 4.5.1: Configuration scrubbing flow chart

Command	Description
Idle state	I(ASCII)
Observation state	O(ASCII)
Query state	Q <11-digit hex value>
Error Injection	N <11-digit hex value>

Table 4.5.1: ASCII commands for SEM-IP state transitions

4.5.1 Hardware/Software requirements

Category	Description
Hardware	Zynq Ultrascale+ MPSoC
Peripherals	UART
IP	Soft-Error Mitigation v3.1 UART helper block
Software	Vivado 2018.3
Terminal	Minicom

Table 4.5.2: Hardware/Software requirements for FPGA configuration scrubbing

Table 4.4.1 provides an overview of the hardware/software requirement. The SEM-IP controller and UART helper block run at 100 MHz clock frequency. The UART module of the processor system uses a default speed of 115200 Bauds, and the SEM-IP controller is used in mitigation and testing mode in order to test the design using an

error injection mechanism. The ARM Cortex-R5 RPU is used to control the SEM-IP controller in dual lockstep mode.

Chapter 5

Results

This section discusses the results and analysis of the implementation stated in the previous section of the report. Execution times are calculated for the implementations carried out in the Single-core ARM Cortex-M4 VA41630 MCU. For the implementations carried out in the Programmable Logic part of the Processing board, latency and area of the design are calculated. The outputs of the implementations are shown and described in this section as well.

5.1 Communication

Operation	Execution Time (ms)
Space packet reception	97
Space packet parsing	98
Crc verification	857 µs

Table 5.1.1: Execution time for Communication implementation

The Supervisor receives a space packet of 1024 B takes on an average of 97 ms. Once the reception is complete, 98 ms is spent on converting the bytes into a space packet. The CRC verification in order to check the packet integrity takes 857 µs as shown in the table 5.1.1.

The figure 5.1.1 depicts the reception of a space packet from the OBC. On parsing the received bytes into space packet, it is found that the packet is meant for the MPSoC and hence the Supervisor forwards the packet.

The Supervisor receives a packet and decodes it. An error is found in the packet data

```
00> DEBUG: TC Received..
00> DEBUG: SpacePacket parsed...
00> packet_version_no 00000000
00> packet_type 00000001
00> secondary_header_flag 00000000
00> apid src 00000001
00> apid dest 00000000
00> apid name 00000000
00> sequence flags 00000011
00> seqnece count 0001
00> packet length 1018
00> recv crc 197F
00> DEBUG: Checking CRC..
00> DEBUG: Generated CRC 197F
00> DEBUG: CRC Verified
00> DEBUG: To be routed to SoC
```

Figure 5.1.1: Excerpt of output of the Communication implementation

whilst performing 16-bit CRC verification. The error was induced into the data by the OBC in order to check that the Supervisor performs the integrity check on the incoming packets, as shown in 5.1.2. The CRC that was received from the OBC was `0x197F` while the CRC generated by the Supervisor was found to be `0xE9E3`. This indicates that an error has occurred in the packet data, and the Supervisor requests the OBC to re-send the packet.

```
00> DEBUG: TC Received..
00> DEBUG: SpacePacket parsed...
00> packet_version_no 00000000
00> packet_type 00000001
00> secondary_header_flag 00000000
00> apid src 00000000
00> apid dest 00000001
00> apid name 00000000
00> sequence flags 00000011
00> seqnece count 0001
00> packet length 1018
00> recv crc 197F
00> DEBUG: Checking CRC..
00> DEBUG: Generated CRC E9E3
00> DEBUG: CRC Verification fail...
```

Figure 5.1.2: Excerpt of output of the Communication implementation

5.2 Watchdog

Operation	Execution Time (μs)
Initiating reset	13792
Reset Counter	< 1
Counter expiration	60000000

Table 5.2.1: Execution time for watchdog implementation

The table 5.2.1 shows the execution times for the watchdog functionality implemented in the Supervisor. The time taken in order to initiate a reset for the processing board in the absence of a heartbeat signal from the PS/PL is seen to be 13 792 µs. The execution time to reset a counter is always less than 1 µs. Whereas, the counter expiration is always 60 s implying that the counter waits for 60 s before triggering a reset for the processing board.

```
DEBUG: Received Heartbeat from PS
DEBUG: TIM1(PS) stopped at 58secs
DEBUG: Timer Reset initiated...
DEBUG: Received Heartbeat from PS
DEBUG: TIM1(PS) stopped at 58secs
DEBUG: Timer Reset initiated...
DEBUG: Received Heartbeat from PS
DEBUG: TIM1(PS) stopped at 58secs
DEBUG: Timer Reset initiated...
DEBUG: PL Timer Rolled off
DEBUG: Power cycling the board...
DEBUG:
```

Figure 5.2.1: Excerpt of output of the watchdog implementation

Figure 5.2.1 depicts the execution of watchdog functionality where the Supervisor receives a heartbeat every second from the PS of the processing board and resets the counter related to PS heartbeat. However, the PL counter expires after 60 seconds without the reception of the heartbeat signal for the PL system of the processing board. The counter/timer interrupt service routine sends a reset pulse to the processing board.

5.3 Memory scrubbing

The table 5.3.1 provides the execution times for memory scrubbing implementation in the Supervisor. It takes 0.675 ms to read 1024 B of data from the NOR flash and 0.868 ms to generate a CRC checksum. Scrubbing process for the whole 32 MB of the memory takes 5056 ms on an average.

Operation	Execution Time (ms)
Single Read from the memory (1024 B)	0.675
Single CRC generation (1024 B)	0.868
Scrubbing (32 MB)	5056

Table 5.3.1: Execution time for memory scrubbing implementation

```
DEBUG: DevID Check: 19
DEBUG: dataSegment: 32768
DEBUG: dataRemain: 0
DEBUG: Performing Memory Scrubbing...
DEBUG: timeDiff: c532
DEBUG: CRC 1824c727 Matched, check successful!
DEBUG: CRC of whole mem: 1824c727
DEBUG: Performing Memory Scrubbing...
DEBUG: timeDiff: c532
DEBUG: CRC 1824c727 Matched, check successful!
DEBUG: CRC of whole mem: 1824c727
DEBUG: Performing Memory Scrubbing...
```

Figure 5.3.1: Excerpt of output of the memory scrubbing implementation

Figure 5.3.1 depicts the working of the memory scrubbing. The Supervisor reads the data from memory 1024 bytes at a time until it reaches 32 MB. The Supervisor generates a CRC checksum every time it reads 1024 bytes of data from memory. Finally, it compares the generated CRC checksum with the stored CRC to detect the presence of any errors.

```
DEBUG: Performing Memory Scrubbing...
DEBUG: Injecting Multi-bit errors
DEBUG: CRC 40a3c9eb not Matched, expected CRC 1824c727
DEBUG: Check failed!
DEBUG: CRC of whole mem: 40a3c9eb
```

Figure 5.3.2: Excerpt of output of the memory scrubbing implementation

When the Supervisor finds an error in the memory, it generates a Telemetry packet to the OBC reporting the error and requesting to re-write the memory content, as shown in figure 5.3.2.

5.4 Software/Firmware update

Operation	Execution Time (μs)
Page write	456
Sub-sector erase	250
Single CRC generation (32 B)	28
Single CRC read (32 B)	21

Table 5.4.1: Execution time for software/firmware update/verification implementation

The update and verification operation of a software/firmware makes use of important operations such as page-write, sub-sector erase, and CRC checksum generation. Table 5.4.1 provides the execution times for these operations for any image that is to be updated or verified. Writing 256 B into NOR flash memory takes 456 µs. Whereas, performing an erase operation on 4096 B of the memory takes 250 µs. Reading 32 B and generating a single CRC checksum for the same 32 B of data takes 21 µs and 28 µs, respectively. However, the execution of the whole software/firmware update or verification process varies according to different image sizes.

Operation	Execution Time (s)
1 KB	0.128
4 KB	0.482
13 KB	1.523
24KB	2.792
1 MB	126.910

Table 5.4.2: Update function execution time for various images

The time taken to update and verify various binary images are given in tables 5.4.2 and 5.4.3. As the size of the images increase, time taken by the Supervisor to execute it increases as well. It takes 0.128 s to update a binary image of 1 kB and it takes 126.910 s (roughly 2 minutes) for a 1 MB image. Similarly, in the scenario of verifying the updated images the time taken by 1 kB, 1 kB, 4 kB, 13 kB, 1 MB are 2.600, 9.958, 29.517, 53.117 and 1580.093 ms, respectively.

Operation	Execution Time (ms)
1 KB	2.600
4 KB	9.958
13 KB	29.517
24KB	53.117
1 MB	1580.093

Table 5.4.3: Verification function execution time for various images

The figures 5.4.1 and 5.4.2 show the update process of 24 kB image, where it can be seen that the binary images are split into pages and written to the memory. Also, the Supervisor waits for the binary image data from OBC until the update is finished. The Supervisor receives the image in parts of 25, as shown in the sequence count (in hexadecimal) in figure 5.4.2.

```
00> packet_version_no 00000000
00> packet_type 00000001
00> secondary_header_flag 00000000
00> apid src 00000000
00> apid dest 00000001
00> apid name 00000100
00> sequence flags 00000000
00> seqnece count 0001
00> packet length 1018
00> crc A162
00> CRC Verified
00> DEBUG: Status OK
00> DEBUG: In Update state machine
00> DEBUG: starting address: 0
00> DEBUG: Writing to memory 0
00> DEBUG: Writing to memory 100
00> DEBUG: Writing to memory 200
00> DEBUG: ending address: 3f8
```

Figure 5.4.1: Excerpt of output of the image update (24 KB)

The figure 5.4.3 shows that the Supervisor reads the memory location where the updated image of size 24 kB, generates the CRC checksum and performs a CRC verification using the CRC checksum that arrived with the Telecommand packet as reference. The operation is successful as the CRC checksum passes the verification test.

```
00> packet_version_no 00000000
00> packet_type 00000001
00> secondary_header_flag 00000000
00> apid src 00000000
00> apid dest 00000001
00> apid name 00000100
00> sequence flags 00000011
00> seqnece count 0019
00> packet length 226
00> crc 4BDD
00> CRC Verified
00> DEBUG: Status OK
00> DEBUG: In Update state machine
00> DEBUG: starting address: 5f40
00> DEBUG: ending address: 6020
00> total bytes written 24608
00> DEBUG: Ending the update..
```

Figure 5.4.2: Excerpt of output of the image update (24 KB)

```
00> packet_type 00000001
00> secondary_header_flag 00000000
00> apid src 00000000
00> apid dest 00000001
00> apid name 00000101
00> sequence flags 00000011
00> seqnece count 0001
00> packet length 1018
00> crc D030
00> CRC Verified
00> DEBUG: Status OK
00> DEBUG: In Update Verify state machine
00> Final CRC on reading memory 5008
00> DEBUG: CRC check succesful, sending ack
```

Figure 5.4.3: Excerpt of output of the image verification (24 KB)

The verification process was tested by injecting an error while the image update (24 KB binary image) function was being performed. The CRC verification failed as the CRC generated was `0xD7F9` instead of `0x5008`. The Supervisor requested the OBC to initiate the update of the 24 kB binary image again, as shown in the figure 5.4.4.

```
)0> DEBUG: Status OK
)0> DEBUG: In Update Verify state machine
)0> DEBUG: RECV CRC 5008
)0> Final CRC on reading memory d7f9
)0> DEBUG: CRC check failed, sending ack
)0> packet_version_no 00000000
)0> packet_type 00000001
)0> secondary_header_flag 00000000
)0> apid src 00000000
)0> apid dest 00000001
```

Figure 5.4.4: Excerpt of output of the image verification (24 KB)

5.5 FPGA configuration study

The SEM-IP takes 56 ms to browse through its configuration frames in order to detect errors. In case it stumbles upon any errors in the frames, it takes around 88 µs to correct a single bit error. The boot and initializing times of the design are also mentioned in table 5.5.1.

Operation	Execution Time
Boot	254 ms
Initialize	142 ms
Error detection	56 ms
Error correction	88 µs

Table 5.5.1: SEM-IP execution time

The area consumption of the SEM-IP controller is shown in table 5.5.2. This indicates that the SEM-IP uses less than 0.5 % of the available resources in the Zynq Ultrascale+ MPSoC which can be calculated with the help of table 2.3.1.

Resource	Number
Look-up tables	430
Flip-flops	530
Input/Output	64
BRAM	4
DSP	1

Table 5.5.2: SEM-IP area analysis

Table 5.5.3 provides the necessary information to understand the reports of the SEM-IP controller. SC (State Change) gives the current state of the SEM-IP out of the nine

possible states. The four states which are relevant to the results are alone represented in the table 5.5.3. For further information regarding the entire list of states, refer to Ultrascale architecture SEM-IP documentation [28]. FC (Flag Change) reports the importance of the affected bit(s) and also mentions whether the affected bit(s) is correctable or not. There are two types of errors that can be identified by the SEM-IP controller: CRC-based error and ECC-based error. The ECC acronym signifies the presence of an ECC-based error. The presence of a CRC-based error was not encountered in the results hence they are outside the scope of this thesis project. TS provides a timestamp of the event of error correction. PA and LA represent the physical address and the linear address of the frame, respectively. COR followed by END signifies the end of the error correction process. The acronyms WD and BT denotes the word and bit position of the detected error in the frame.

Report acronyms	**Description**
SC 00	Idle state
SC 02	Observation state
SC 04	Correction state
SC 08	Classification state
SC 10	Injection state
FC 00	Flag change: correctable, non-essential bit
FC 40	Flag change: correctable, essential bit
RI 00	Reserved Information
ECC	ECC-based error
TS {8-digit hex value}	Timestamp
PA {8-digit hex value}	Physical address
LA {8-digit hex value}	Linear address
COR END	Correction ended
WD {2-digit hex value}	Word in the frame
BT {2-digit hex value}	Bit position in the word

Table 5.5.3: SEM-IP report description

The results of latency and area of SEM-IP show the swiftness in terms of error detection and correction and efficiency in terms of resource utilization. Figure 5.5.1 shows the error being injected into the design. The error was injected in the bit 0 of the 46^{th} word of the frame number 7000. Hence, the linear frame address equals to the value `0xC00070005C0`. It can also be viewed from the same figure 5.5.1 that the error in the stated frame can be viewed by sending Query command with the appropriate frame address.

Figure 5.5.2 shows the error injected in the middle of the frame number 7000. The

```
I> N C00070005C0
SC 10
SC 00
I> Q C0007000000
00000000
00000000
```

Figure 5.5.1: Excerpt of output of the configuration scrubbing implementation

```
00000000
00000000
00000000
00000000
00000001
00000000
00000000
00000000
00000000
```

Figure 5.5.2: Excerpt of output of the configuration scrubbing implementation

```
I> O
SC 02
O>
RI 00
SC 04
ECC
TS 00005221
PA 00102612
LA 00007000
COR
WD 2E BT 00
END
FC 00
SC 08
FC 40
SC 02
O>
```

Figure 5.5.3: Excerpt of output of the configuration scrubbing implementation

table 5.5.3 helps in understanding the report shown in the figure 5.5.3. When the SEM-IP controller was put in the observation state, the error was detected, and the controller moved into the correction state (SC 04). The detected error was an ECC-based error which is denoted by the acronym ECC. The physical and linear frame address where the error was present is shown to be `0x102612` and `0x7000`, respectively. The error detected was present in the bit 0 of the word 2E of the frame `0x7000`. The SEM-IP identifies the error as a correctable error and the bit as a non-essential bit (FC 00). Since the error classification was not used in the implementation, the SEM-IP controller jumps from the classification state (SC 08) to the observation state (SC 02) without performing error classification.

Chapter 6

Conclusion and Future works

This chapter provides a conclusion for this thesis work and also discusses the limitations faced during the course of action. Finally, the chapter ends with the future works that can be carried out as an extension of this master thesis work.

6.1 Conclusion

Using space-grade components in CubeSats increases the cost of the mission tremendously. This makes it hard for institutions and organizations involved in small-scaled CubeSat missions to make use of expensive space-grade components. This thesis work aids in designing a CubeSat hardware architecture using COTS MPSoC as a processing center and an affordable radiation-hardened MCU that acts as a Supervisor for the processing center. This architecture will potentially be applied in the EIVE CubeSat mission. However, this Cubesat architecture is not restricted to one particular application because it was implemented, keeping in mind the adaptability to incorporate it with any application. This is attributed to the versatility of the Zynq Ultrascale+ MPSoC and the resilient supervision by Vorago VA41630 MCU.

This thesis provides insight into how the Supervisor safeguards the COTS MPSoC by maintaining reliable communication and watchdog monitoring to ensure the continuous operation of the processing board. It also performs memory scrubbing for the primary NOR flash memory to ward away single-event upsets in the memory. The Rad-hard MCU maintains all these operations external to the MPSoC.

Internally, the COTS MPSoC performs configuration scrubbing to monitor single-event upsets in CRAM of the PL with the help of the PS present in the MPSoC itself.

6.2 Limitations

The CubeSat hardware architecture was designed and developed using individual evaluation kits and components such as Zynq Ultrascale+ MPSoC and Vorago VA41630 Cortex-M4 based MCU, 32 MB SPI NOR flash. This leads to connections being made using jumper cables. Also, all the peripherals of the devices were not accessible at the same time because of the availability restricted number of I/Os. This made the development of the application difficult, and the testing of all the applications simultaneously was not possible. Developing a custom kit with both processing board and Supervisor on the same board will prove useful in such situations.

6.3 Future Work

The project work presented can be extended by incorporating several things. Firstly, implementing and testing latch-up monitoring functionality to avoid latch-up in the Zynq Ultrascale+ MPSoC can be added to the design. The reason is that this work discusses latch-up monitoring as one of the many functionalities of the Supervisor in the architecture section. The same reason can be applied to the implementation of data logging in MRAM. Secondly, the work makes use of bare-metal implementation of functionalities implemented in the Supervisor. Instead, an RTOS, such as FreeRTOS, can be used to make the design efficient and robust by scheduling the functionalities as different tasks and assigning appropriate priorities to them.

Bibliography

[1] Adell, Philippe C., Moro, Slaven, Gouyet, Lionel, Chatry, Christian, and Vermeire, Bert. “Single event effect assessment of a 1-Mbit commercial magneto-resistive random access memory (MRAM)”. In: *IEEE Radiation Effects Data Workshop* 2017-July (2017). DOI: 10.1109/NSREC.2017.8115456.

[2] Anderson, Jordan D., Leavitt, Jennings C., and Wirthlin, Michael J. “Neutron Radiation Beam Results for the Xilinx UltraScale+ MPSoC”. In: *2018 IEEE Nuclear and Space Radiation Effects Conference, NSREC 2018* (2018), pp. 1–7. DOI: 10.1109/NSREC.2018.8584297.

[3] Apu, Processing Unit. “Zynq UltraScale + MPSoC Data Sheet : Overview Processing System (PS) Arm Cortex-A53 Based Application Dual-core Arm Cortex-R5 Based On-Chip Memory”. In: 891 (2018), pp. 1–42.

[4] Bez, R., Camerlenghi, E., Modelli, A., and Visconti, A. “Introduction to flash memory”. In: *Proceedings of the IEEE* 91.4 (2003), pp. 489–502.

[5] CCSDS. “Space Packet Protocol”. In: *Ccsds 133.0-B-1* September 2003 (2003), p. 49.

[6] *CubeSat Overview*. https://www.nasa.gov/mission_pages/cubesats/overview. Accessed: 03-08-2020.

[7] Dong, Z., Guo, Y., Gong, Y., and Li, C. “A High Reliability Radiation Hardened On-Board Computer System for Space Application”. In: *2016 Sixth International Conference on Instrumentation Measurement, Computer, Communication and Control (IMCCC)*. 2016, pp. 671–674.

[8] Fuchs, Christian M., Murillo, Nadia, Plaat, Aske, Van der Kouwe, Erik, Harsono, Daniel, and Stefanov, Todor. “Fault-Tolerant Nanosatellite Computing on a Budget”. In: March 2019 (2019). arXiv: 1903.08781. URL: http://arxiv.org/abs/1903.08781.

[9] Glorieux,
Maximilien, Evans, Adrian, Lange, Thomas, In, A. Duong, Alexandrescu, Dan, Boatella-Polo, Cesar, Alia, Ruben Garcia, Tali, Maris, Ortega, Carlos Urbina, Kastriotou, Maria, Fernandez-Martinez, Pablo, and Ferlet-Cavrois, Veronique. "Single-Event Characterization of Xilinx UltraScale+ ® MPSOC under Standard and Ultra-High Energy Heavy-Ion Irradiation". In: *2018 IEEE Nuclear and Space Radiation Effects Conference, NSREC 2018* (2018). DOI: 10.1109/NSREC.2018.8584296.

[10] Hiemstra, David M., Kirischian, Valeri, and Brelski, Jakub. "Single event upset characterization of the Zynq UltraScale+ MPSoC using proton irradiation". In: *IEEE Radiation Effects Data Workshop* 2017-July (2017), pp. 4–7. DOI: 10.1109/NSREC.2017.8115448.

[11] Ikegawa, Sumio, Mancoff, Frederick B., Janesky, Jason, and Aggarwal, Sanjeev. "Magnetoresistive Random Access Memory: Present and Future". In: *IEEE Transactions on Electron Devices* 67.4 (2020), pp. 1407–1419. ISSN: 15579646. DOI: 10.1109/TED.2020.2965403.

[12] Irom, Farokh and Nguyen, Duc N. "SEE and TID response of Spansion 512Mb NOR flash memory". In: *IEEE Radiation Effects Data Workshop* (2011), pp. 143–146. DOI: 10.1109/REDW.2010.6062520.

[13] Le, Robert. "Soft Error Mitigation Using Prioritized". In: 538 (2012), pp. 1–11. URL: http://www.xilinx.com/support/documentation/application%7B%5C_%7Dnotes/xapp538-soft-error-mitigation-essential-bits.pdf.

[14] Maurer, Richard H., Fraeman, Martin E., Martin, Mark N., and Roth, David R. "Harsh environments: Space radiation environment, effects, and mitigation". In: *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)* 28.1 (2008), pp. 17–29. ISSN: 02705214.

[15] Nagarajan, Chandrasekhar, D'Souza, Rodney Gracian, Karumuri, Sukumar, and Kinger, Krishna. "Design of a cubesat computer architecture using COTS hardware for terrestrial thermal imaging". In: *Proceeding - ICARES 2014: 2014 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology* (2014), pp. 67–76. DOI: 10.1109/ICARES.2014.7024379.

[16] Nguyen, N. "Radiation Effects on Advanced Flash Memoriest D." In: 46.6 (1999), pp. 1744–1750.

[17] Osman, Duaa A.M. and Mohamed, Sondos W.A. “Hardware and software design of Onboard Computer of ISRASAT1 CubeSat”. In: *Proceedings - 2017 International Conference on Communication, Control, Computing and Electronics Engineering, ICCCCEE 2017* (2017), pp. 1–4. DOI: 10.1109/ICCCCEE.2017.7867654.

[18] Perez, Arturo, Otero, Andres, and Torre, Eduardo de la. “Performance Analysis of SEE Mitigation Techniques on Zynq Ultrascale + Hardened Processing Fabrics”. In: Aug. 2018, pp. 51–58. DOI: 10.1109/AHS.2018.8541490.

[19] *Radiation Hardened ARM® Cortex-M4*. https://www.voragotech.com/products/va41630. Accessed: 03-08-2020.

[20] Schoch, B., Chartier, S., Mohr, U., Koller, M., Klinkner, S., and Kallfass, I. “Towards a CubeSat Mission for a Wideband Data Transmission in E-Band”. In: *2020 IEEE Space Hardware and Radio Conference (SHaRC)*. 2020, pp. 16–19.

[21] Sheng-Ju, S. “Implementation of Cyclic Redundancy Check in Data Communication”. In: *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*. 2015, pp. 529–531.

[22] Singh, A. K. “Comprehensive study of error detection by cyclic redundancy check”. In: *2017 2nd International Conference for Convergence in Technology (I2CT)*. 2017, pp. 556–558.

[23] Skauen, Andreas Nordmo, Jahnsen, Berit, Smestad, Tore, Grimstvedt, Eirik Skjelbreid, Gulbrandsen, Fredrik, Cotten, Brad, and Zee, Robert E. “NorSat-3 – next generation Norwegian maritime surveillance”. In: ().

[24] Stoddard, Aaron Gerald. “Configuration Scrubbing Architectures for HighReliability FPGA Systems”. In: (2015), p. 159.

[25] Stoddard, Aaron, Gruwell, Ammon, Zabriskie, Peter, and Wirthlin, Michael J. “A Hybrid Approach to FPGA Configuration Scrubbing”. In: *IEEE Transactions on Nuclear Science* 64.1 (2017), pp. 497–503. ISSN: 00189499. DOI: 10.1109/TNS.2016.2636666.

[26] Toorian, A., Diaz, K., and Lee, S. “The CubeSat Approach to Space Access”. In: *2008 IEEE Aerospace Conference*. 2008, pp. 1–14.

[27] Wikipedia contributors. *CubeSat — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-October-2020]. 2020. URL: https://en.wikipedia.org/w/index.php?title=CubeSat&oldid=978683366.

[28] Xilinx. "UltraScale Architecture Soft Error Mitigation Table of Contents". In: (2017), pp. 1–178. URL: https://www.xilinx.com/support/documentation/ip%7B%5C_%7Ddocumentation/sem%7B%5C_%7Dultra/v3%7B%5C_%7D1/pg187-ultrascale-sem.pdf.

[29] Xilinx. "UltraScale Devices Maximize Design Integrity with Industry-Leading SEU Resilience and Mitigation". In: 462 (2015), pp. 1–11. URL: http://www.xilinx.com/support/documentation/white%7B%5C_%7Dpapers/wp462-ultrascale-SEU.pdf.

[30] Xilinx Inc. "Zynq UltraScale+ Device TRM". In: 1085 (2019), pp. 1–1177. URL: https://www.xilinx.com/support/documentation/user%7B%5C_%7Dguides/ug1085-zynq-ultrascale-trm.pdf.

[31] "Zynq ® UltraScale +™ MPSoCs CG". In: (2018).

TRITA-EECS-EX-2020:815