Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

ISIS
Innovative Solutions In Space B.V.

M.Sc. Thesis

System-level Fault-Tolerance Analysis of Small Satellite On-Board Computers

Dmitry Burlyaev

Abstract

Commercial Off-The-Shelf (COTS) electronic components offer cost-effective solutions for the development of On-Board Computers (OBCs) in the small satellite industry. However, the COTS parts are not originally designed to withstand the space radiation environment. Traditional fault-tolerance practices rely on expensive radiation tests or are based on circuit-level knowledge which are not easily available. This work proposes a novel simulation-based statistical approach to assist the satellite designers in performing OBC fault-tolerance analysis.

The presented novel approach is based on high-level system modeling and an object-oriented fault injection mechanism. Such a technique allows the comparison between fault-tolerance techniques and reveals the consequences of radiation effects in the COTS parts at early development stages.

The work covers the implementation of the proposed simulation framework which includes the OBC and fault modeling. The fault models are based on the conducted radiation environment analysis. The range of software and hardware fault detection and mitigation techniques are investigated as case studies. They include time and hardware Triple-Modular Redundancy (TMR), FPGA-based memory scrubbing with Hamming encoding, and watchdog/co-processor monitoring. The case studies reveal that the proposed approach can be used to choose suitable fault-tolerance techniques, increase their efficiency, and reduce the required hardware resources.

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science Delft University of Technology

System-level Fault-Tolerance Analysis of Small Satellite On-Board Computers

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Embedded Systems

by

Dmitry Burlyaev
born in Kamensk-Uralsky, Russian Federation

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"System-level Fault-Tolerance Analysis of Small Satellite On-Board Computers"** by **Dmitry Burlyaev** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 2 July 2012

Chairman:

Prof. dr. ir. Alle-Jan Van der Veen, CAS, TU Delft

Advisor:

Dr. ir. Rene van Leuken, CAS, TU Delft

Committee Members:

Dr. ir. Said Hamdioui, CE, TU Delft

Ir. Maxime Castera, Innovative Solutions In Space B.V. (ISIS)

Abstract

COTS electronic components offer cost-effective solutions for the development of OBCs in the small satellite industry. However, the COTS parts are not originally designed to withstand the space radiation environment. Traditional fault-tolerance practices rely on expensive radiation tests or are based on circuit-level knowledge which are not easily available. This work proposes a novel simulation-based statistical approach to assist the satellite designers in performing OBC fault-tolerance analysis.
The presented novel approach is based on high-level system modeling and an object-oriented fault injection mechanism. Such a technique allows the comparison between fault-tolerance techniques and reveals the consequences of radiation effects in the COTS parts at early development stages.
The work covers the implementation of the proposed simulation framework which includes the OBC and fault modeling. The fault models are based on the conducted radiation environment analysis. The range of software and hardware fault detection and mitigation techniques are investigated as case studies. They include time and hardware TMR, FPGA-based memory scrubbing with Hamming encoding, and watchdog/co-processor monitoring. The case studies reveal that the proposed approach can be used to choose suitable fault-tolerance techniques, increase their efficiency, and reduce the required hardware resources.

Acknowledgments

I would like to express my deep gratitude to my university thesis advisor Dr. ir. Rene van Leuken, CAS, TU Delft for his invaluable input to my research progress and support in the academic world. I also would like to thank the company ISIS for the given freedom in the thesis topic choice and the industrial insight into satellite-related problems. I am very grateful to Software and Simulation departments of ISIS company and their leaders: Maxime Castera and Arthur Overlack.

I express the warmest thanks to my parents for everyday support and help in the choice of my path.

Dmitry Burlyaev
Delft, The Netherlands
2 July 2012

Contents

List of Figures

List of Tables

Acronyms

ADC	Analog-to-Digital Converter.
ADCS	Attitude Determination and Control System Algorithm.
ASIC	Application-Specific Integrated Circuit.
BRAM	Block Random-Access Memory (RAM).
CalPoly	California Polytechnic State University.
CMOS	Complementary Metal-Oxide-Semiconductor.
COTS	Commercial Off-The-Shelf.
CPU	Central Processing Unit.
DAC	Digital-to-Analog Converter.
DFF	D Flip-Flop.
DMIPS	Dhrystone MIPS(Million Instructions Per Second).
DRAM	Dynamic RAM.
ECC	Error-Correcting Code.
EDAC	Error-Dectection And Correction.
ESA	European Space Agency.
FG	Floating Gate.
FIM	Fault Injection Module.
FMEA	Failure Mode and Effect Analysis.
FPGA	Field-Programmable Gate Array.
FSM	Finite State Machines.
GCRs	Galactic Cosmic Rays.
GG	Guard Gate.
GPP	General-Purpose Processor.
GPR	General-Purpose Register.
GPS	Global Positioning System.
GUI	Graphical User Interface.
IC	Integrated Circuit.
IP	Intellectual Property.
ISIS	Innovative Solutions In Space B.V..
JTAG	Joint Test Action Group.
LCI	Logic Cell Like-Inverter.
LEO	Low Earth Orbit.

LET	Linear Energy Transfer.
LMA	Load Memory Address.
LSI	Large Scale Integration.
MBU	Multiple-Bit Upsets.
MCU	Multiple-Cell Upsets.
MLC	Multi-Level Cell.
MPU	Micro-Processor Unit.
NASA	National Aeronautics and Space Administration.
OBC	On-Board Computer.
OS	Operating System.
OTP	One-Time Programmable.
OVP	Open Virtual Platforms.
PC	Personal Computer.
PCB	Printed Circuit Board.
PROM	Programmable Read-Only Memory.
QoS	Quality of Service.
RAM	Random-Access Memory.
RTL	Register-Transfer Level.
RTOS	Real-Time Operating System.
SAA	South Atlantic Anomaly.
SDRAM	Synchronous Dynamic RAM.
SEE	Single-Event Effect.
SEFI	Single-Event Functional Interrupt.
SEL	Single-Event Latchup.
SEP	Single-Event Phenomenon.
SERVIS	Space Environment Reliability Verification Integrated Systems.
SET	Single-Event Transient.
SEU	Single-Event Upset.
SHE	Single Hard Error.
SIFT	Software Implemented Fault Tolerance.
SLC	Single-Level Cell.
SoC	System-on-Chip.
SPENVIS	The Space Environment Information System.
SPI	Serial Peripheral Interface.

SPR Special-Purpose Register.
SRAM Static Random-Access Memory.

TID Total Ionizing Dose.
TLM Transaction-Level Modeling.
TMR Triple-Modular Redundancy.

VMA Virtual Memory Address.

1 Introduction

The On-Board Computer (OBC) of a satellite is its central subsystem that processes information transmitted to the satellite and information provided by other on-board subsystems (radio, power, payload, etc.)[7, p.348]. In spite of adverse effects of harsh radiation environment, the OBC performance degradation should be minimized and the efficiency of software/hardware fault-tolerance techniques have to be analysed to provide clear understanding about satellite mission capabilities.

The radiation fault-tolerance techniques have become an especially active research topic when newly introduced small satellites and CubeSats[8] began to use low-cost Commercial Off-The-Shelf (COTS) components that are not designed to work in the space radiation environment and, as a result, susceptible to the radiation effects.

In the next Sections the research motivation is given, the need to develop a simulation-based approach for the OBC fault-tolerance analysis is explained, the main contributions of this work are listed, and the thesis organization is presented.

1.1 Motivation

The main requirement for small satellite OBCs is their tolerance towards the faults induced by radiation. While the complexity of each component and whole systems are growing, the existing analytical methods for fault-tolerance analysis became infeasible for complex heterogeneous systems and for COTS components whose Integrated Circuit (IC)-level is unknown for the satellite designers.

The face-off between the requirements on the fault-tolerance and low-cost (COTS-based design) are partially solved by using COTS parts with flight heritage and satellite aluminium shielding. The former practice limits the OBC power efficiency and performance, while the latter one increases the price of a satellite launch. The up-to-date COTS devices can be used but require additional fault-tolerance analysis and clear understanding of the radiation effects consequences. Thus,an important existing question is how to assess the efficiency of the applied fault-tolerance techniques for particular applications and device set.

The benefits and correctness of implemented fault-tolerance techniques can be assessed under the radiation tests at the final development stage. However, if the radiation test is failed, the expensive and time-consuming re-design is required. The comparative analysis of the mitigation techniques and system-level debugging for fault-tolerance are impossible due to high cost of radiation tests. Moreover, radiation testing does not guarantee the injection of specific errors, nor explain their nature or the system behavior[9, 10].

Additionally, the existing simulation methods for the OBC fault-tolerance analysis are based on IC-level which is unknown for the final user of COTS components.

Moreover, circuit-level simulations are time-consuming and cannot be used for extensive statistical analysis.

Due to the complexity of each electronic component and the system as a whole, the utilization of high-level abstraction modeling language, such as SystemC, is imperative. The system-level fault-tolerance analysis through simulation is examined in this work.

1.2 Thesis goal

The goal of this thesis is the creation of the simulation framework that enables statistical system-level fault-tolerance analysis of the OBC. The framework should assist software developers and system designers to write OBC software and conduct Failure Mode and Effect Analysis (FMEA) at early development stages. It should clarify radiation effect consequences for typical CubeSat missions. Consequently, the evaluation of the space radiation environment is required and the corresponding fault models have to be built.

The work should include several case studies where the framework utilization is explained and the interpretation of the simulation results are given.

The framework should support software portability to real hardware. It should be scalable and general enough to be easily adapted for other OBC architectures.

1.3 Contributions

The contributions of this thesis are:

- A novel statistical approach for system-level fault-tolerance analysis of satellite OBCs is built (Section 6.1) and based on the high-level simulation framework of two main components: SystemC-based OBC model (Section 4.1-4.4) and C++-based fault injection mechanism (Section 4.5). The requirement on the hardware-software co-design and co-simulation is met with Transaction-Level Modeling (TLM) methodology [11]. The provided portability enables software-hardware co-simulation and co-design in a way that the written software and Field-Programmable Gate Array (FPGA) configuration are portable to the real OBC hardware.

- The radiation environment for typical CubeSat missions is assessed using European Space Agency (ESA) The Space Environment Information System (SPENVIS) project(Section 3.1.1). The interconnection between the assessed radiation environment, the electronic components' cross section , and fault-rates is defined (Section 3.1.2).

- The radiation sensitivity of modern electronic components is investigated using the published empirical observations. The maximum fault-rates are calculated for different types of electronic parts (SRAM, DRAM, Flash memories, etc.)(Section 3.1.3). The corresponding fault-models of possible radiation effects are built: the models of Single-Event Upset (SEU), Multiple-Cell Upsets (MCU), and Single-Event Functional Interrupt (SEFI) (Chapter 5).

- The model of the OBC based on SmartFusion System-on-Chip (SoC) is built(Section 4.3). It incorporates Cortex-M3 ARM core (an instruction-accurate model obtained from Open Virtual Platforms (OVP) project [12]), Flash-based FPGA co-processor, the central AMBA bus (replaced by the Decoder unit), several memory storages, watchdog, and timers. It is also shown how to build the stacked OBC model with I2C-controller utilization(Section 4.4).

- Using the created OBC model, the work investigates the fault consequences for the recursive and compression algorithms (Sections 8.1-8.3) as well as for the Kalman and adaptive filters of Attitude Determination and Control System Algorithm (ADCS)(Sections 8.4-8.5).

- The work shows how to compare the efficiency of such software and hardware mitigation techniques as time Triple-Modular Redundancy (TMR) and FPGA-based memory scrubbing (Section 8.5.2).

- The work shows that the simulation results can be used for the optimization of mitigation techniques, e.g. for FPGA-based memory scrubbing with Hamming encoding and watchdog monitoring(Sections 8.5.4, 8.1.2). The dedicated clustering algorithm has been developed for this purpose (Section 8.5.4).

Three conference papers have been written based on the results presented in this thesis. The work has been appreciated in Europe and USA where it was presented at ESA 4S Symposium[13] and North-Atlantic Testing Workshop[14] in 2012. The third written paper [15] is being under the review of XXVII Conference on Design of Circuits and Integrated Systems by the moment of this MSc thesis defence.

1.4 Outline

The rest of the work is organized as follows:

Chapter 2 observes small satellite and CubeSat missions; it also discusses the typical OBC properties. Chapter 3 describes the space radiation environment, provides the assessment of CubeSat radiation conditions , and briefly observes well-known fault-tolerance techniques. Chapter 4 explains the principles of the chosen OBC modeling approach and introduces the OBC model based on SmartFusion SoC[6]. Chapter 5 explains the fault models used in the simulation framework. Chapter 6 generalises the FMEA with the proposed simulation approach and multidimensional analysis. Chapter 7 observes the OBC model validation, verification, and limitations. Chapter 8 contains the case studies, their interpretation, and explanation how to use the framework for the OBC fault-tolerance analysis. In the last chapter of this work conclusions are drawn and directions for future work are given.

Overview: OBC for Small Satellite missions

2

The previous chapter explains the need to create a new approach for the fault-tolerance analysis of small satellite OBCs. Hereafter, the overview of small satellite missions is presented(Section 2.1), and the typical OBC properties are discussed (Section 2.2).

2.1 Micro and Nano satellite mission overview

Nano and Micro satellites (for brevity small satellites in this work) are satellites with mass ranges of [1..10) kg and [10..100) kg consequently [16, p.30]. COTS-based Nano-satellites(or CubeSats)(Figure 2.1) were introduced in 1999 by California Polytechnic State University (CalPoly) and Stanford University [17, 18]. CubeSats became very popular in the scientific community due to the mission low cost. The low mission cost dictates the use of COTS components and the reduction of satellite mass by the elimination of heavy shielding. Both factors make satellite subsystems highly susceptible to the space radiation. However, the satisfactory results of missions like Hiten [19], TSUBASA [20], and Space Environment Reliability Verification Integrated Systems (SERVIS) [10] certified the feasibility of COTS parts utilization in space.

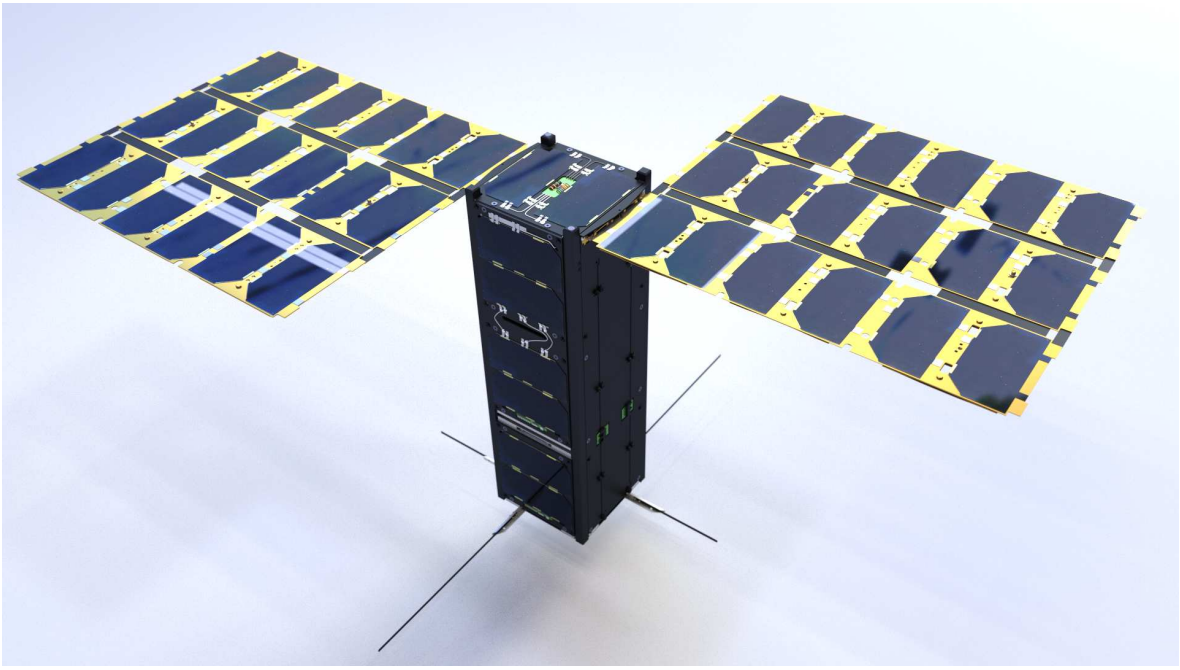

Figure 2.1: 3U CubeSat with deployed antenna and solar panels

Nowadays, small satellites start to be used in long-term research projects, including

commercial ones where the mission success depends not only on the launch success but on the provided Quality of Service (QoS)[21, 22, 23]. Despite of high radiation-tolerance requirements, system engineers are compelled to use COTS components due to their low cost, availability on the electronic market, and cutting edge performance[5].

A satellite system can be always divided into two main parts: a satellite platform and a payload, in other words, a core component and a variable sybsystem[24]. The core component consists of the communication, power, altitude control, and the OBC subsystems[25]. These subsystems should possess re-usability property and meet possible requirements of future missions to reduce time-to-market and non-recurrent costs.

One of the distinguishing characteristics of small satellite missions is the short period of satellite visibility to the ground station: with the typical altitudes of 300-750 km above the sea level, the direct satellite visibility is limited to 3-14 minutes for one ground station[26]. In conjunction with the limited communication data-rate[27, 28], the short pass-by time introduces the higher requirements on the on-board processing power[23]. This requirement is enforced by advancements in payload technology that result in high demand of powerful on-board processing[2].

As for any autonomous vehicle with a rechargeable energy source, satellite subsystems have to consume as low power as possible. This requirement is dictated by the limited number of charge/discharge cycles for lithium ion batteries and solar panels efficiency. A power efficient OBC reduces the amplitude of lithium ion batteries charge/discharge cycles, which prolongs the batteries operational life[29].

All the telemetry and OBC commands are transmitted between the subsystems via a *command and data handling bus*. In CubeSats this bus is standardized by PC/104 bus standard[30] as well as the used Printed Circuit Board (PCB) size is limited by 10x10 centimeters[31, 8].

The kernel part of the satellite that performs the majority of mission tasks is the OBC. The OBC takes responsibility for the satellite auto-operation within the power, mass, size limitations and radiation-tolerance requirements discussed above. In details, the typical OBC of small satellites is analyzed in the next Section 2.2.

2.2 OBC satellite subsystem

The satellite OBC is a computer or a system of computers that processes various information transmitted to the satellite or from other on-board subsystems[7, p.348](see Figure 2.2). The OBC performs all main operations, stores on-board data, and executes the telecommands sent by the ground station[32, 24]. The OBC architecture has to be flexible and general-purpose to increase its reusability[33].

The OBC should have the characters of future small satellites: short time-to-market, high functional integration, flexibility, and low expenses with maximized fault-tolerance[34, 24, 32]. Since the physical upgrade and repair of satellite electronics are impossible after its launch, there is great need for condition-based maintenance, self-repair and upgrade capabilities [35, 23, 36].

The overview of main requirements to the OBC design is presented in the next Section 2.2.1.

Figure 2.2: Existing OBC for Nano satellites [1]

2.2.1 Overview of OBC requirements

To define the requirements on the OBC subsystem it is necessary to analyze the functionality it should perform:

2.2.1.1 Processing power

The execution of an existing ADCS requires at least 80 DMIPS(according to an internal Innovative Solutions In Space B.V. (ISIS) document). The unavailability of the correct ADCS results for more than 5 minutes may cause the satellite orientation loss with following communication loss and solar panels efficiency reduction.

The camera NanoCam[37] can be taken as a case study to conduct an analysis of possible workload by a imaging payload. NanoCam can provide two pictures per second through an I2C interface. The picture JPEG compression will take 317 ms and 45 kB Random-Access Memory (RAM) on modern processors as Cortex-M3[38].

According to the aforementioned application requirements, the appropriate processor will have at least 100-110 DMIPS of performance. One of such processors is ARM Cortex-M3 with 1.25 DMIPS/MHz and frequencies higher than 80 MHz[39].

2.2.1.2 Memory capacity

Based on the NanoCam and altitude/orbit control algorithm characteristics, the OBC needs at least 32 Mbytes of RAM to temporary keep and process pictures, measurements, and payload data. Additionally, non-volatile high-capacitive memory storage is needed to keep the telemetry and payload data until sending it to the ground station. High-capacitive SD Flash memory is usually utilized for this purpose [1, 40].

2.2.1.3 Peripherals

Missions are becoming more diverse and complex; thus, satellites should be able to accommodate various payloads[41]. Consequently, the OBC should demonstrate high flexibility in term of available peripherals. The *command and data handling bus* (see Section 2.1) requires one I2C bus; another I2C will be needed for a Global Positioning System (GPS) receiver[42], camera, or Nano-RTU[43]. Two Serial Peripheral Interface (SPI) interfaces can be utilized for communication with payloads. Sometimes both SPI are used for one payload subsystem, e.g. in Nano satellites Tracking system[44].

2.2.1.4 Software

The principal requirement on the OBC software is bug-free. Additionally, in-flight software update techniques and corresponding boot loader capabilities are desirable[32].This approach will make the OBC architecture capable to recover from software bugs by software in-flight reloading. Since except health-checking functionality the OBC should perform ADCS, payload data processing,etc., the OBC tasks have different importance. Utilization of a Real-Time Operating System (RTOS) provides several priority levels for tasks execution[32].

2.2.2 OBC fault-tolerance

The previous Sections 2.1-2.2.1 explained the OBC importance and its main functional characteristics. However, no OBC functionality can be provided if the OBC is not a fault-tolerant system. The definitions of fault-tolerance and faults are discussed in a few next paragraphs.

On one hand, a fault can be defined as the primary cause of changes in the system structure or parameters that eventually lead to a degraded system performance or even system functional loss[45, p.1-2]. Several examples of fault are listed hereafter[45, p.1]:

- An internal event in the system which breaks an information link.
- A wrong control action given by the human operator that brings the system out of the required point.
- It may be a system design error that remained undetected until the system came into a certain operation point where this error reduced the performance considerably.

In order to avoid the negative fault consequences, the faults have to be found as quickly as possible and the system has to be returned the the correct state as soon as possible . These two steps: detection and recovery are carried out by different fault-detection and fault-mitigation techniques (see Section 3.2). In conjunction these techniques form fault-tolerance techniques. Systems use these techniques to meet the limitations on the performance degradation caused by the introduced faults. As a result, a system becomes fault-tolerant if the balance between the applied fault-mitigation techniques and the faults' severity is kept.

On other hand, the term "fault" can correspond to the representation of a radiation effect at the abstracted function level. In the contest of system defects, the term fault corresponds to the representation of a defect at the abstracted function level [46, p.57-58]. The terms "radiation fault modeling" and "radiation fault models" will be used in this work with the meaning of the representation of a radiation effect at the abstracted function level. The term "fault" in the space industry is more commonly used according to the aforementioned definition from[45, p.1-2].

In the case of small satellite OBCs, the fault-tolerant OBC is the systems that uses fault-tolerance techniques to minimize the negative consequences of the radiation effects.

2.3 Conclusions

This chapter presented the main characteristics of the appropriate OBC for small satellite missions. The characteristics are summarized in Table 2.1.

OBC characteristic	**Required value**
Processing power	$\geq$ 100 MIPS
RAM (volatile memory)	$\geq$ 32 Mbytes
Non-volatile memory storage	SD card ($\geq$ 1 Gb)
Peripherals	$\geq$ two I2C, $\geq$ two SPI
Digital-to-Analog Converter (DAC) and Analog-to-Digital Converter (ADC)	preferably for flexibility
FPGA	preferably for flexibility
Termal conditions	industrial standard, [-40..+70] Celsius
Total power consumption	$\leq$ 1 Watt

Table 2.1: Main OBC requirements

The processing power concentration in the OBC imposes the strict requirement on correct and stable OBC operation. Taking into account the diverse radiation-induced effects in COTS parts (see Chapter 3) and the absence of an appropriate shielding protection, the adequate fault detection and mitigation techniques[41] have to be used (see Section 3.2).

Space Radiation Effects, Faults Detection and Mitigation

3

Space radiation influence on electronics has always been the central issue in the space industry. However, it is necessary to assess the radiation environment particularly for 300-750 km of altitude (see Chapter 2) to understand the impact of radiation on different OBC components (Static Random-Access Memory (SRAM)-based and Flash-based memories, FPGA,etc.)(Sections 3.1.3.1 - 3.1.3.5).

3.1 Space Radiation Environment Overview

Space environment significantly differs from terrestrial conditions, which introduces serious problems for the OBC development and COTS components utilization.

Space-based radiation comprises atomic particles that have been spread by stellar events within the solar system or beyond it (e.g. from The Milky Way galaxy)[2]. The particles stream generated within our solar system is referred to as *the solar wind* (Figure 3.1).

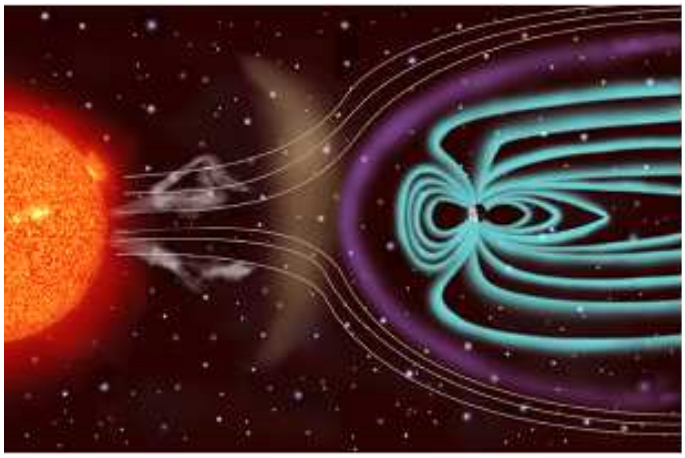

Figure 3.1: Solar Wind [2]

The Earth's magnetosphere traps, slows, or deflects the electrons,protons, and heavy ions (isotopes of atom from Helium to Uranium) emitted during solar events such as solar flares and mass coronal ejection[2]. As a result of protons and electrons interactions with the Earth's magnetic field, they are trapped within the Van Allen belts (Figure 3.2). Thus, the orbits with altitude from 500 to 13 000 km are populated with high energy protons[47].

The inner proton Van Allen belt has a region, South Atlantic Anomaly (SAA), where the belt extends downwards the Earth. Consequently, the high concentration of

protons is observed in this region at lower altitudes(see Figure 3.3) and constitutes a serious danger for small satellite missions.

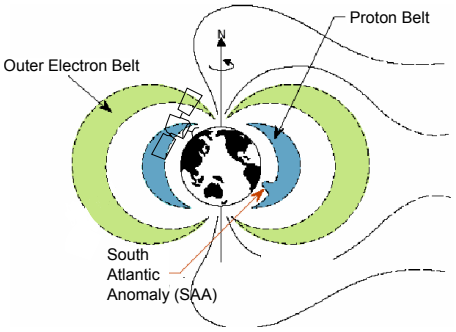

Figure 3.2: Van Allen Belts and SAA[2]

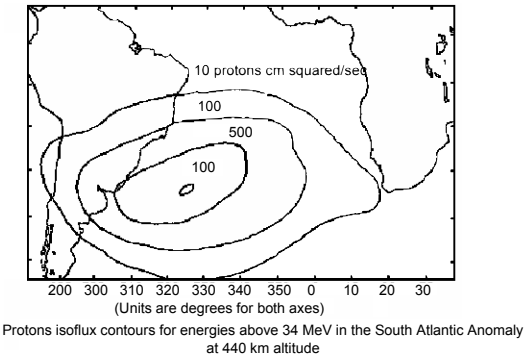

Figure 3.3: Proton concentrations within SAA[2]

Moreover, the statistical correlation between radiation-induced faults in the OBC and solar activities was revealed by the Hiten satellite mission[35].

Solar flares or spatial anomalies cause variances of one or two orders of magnitude in environmental effects. They cannot cause a system failure but the OBC limited performance degradation is highly expected [5].

3.1.1 Space Radiation Environment Assessment

The radiation effects in electronic components can be classified into Single-Event Phenomenon (SEP)(or Single-Event Effect (SEE)) and Total Ionizing Dose (TID) over the mission lifetime [5, 25, 4]. SEP(or SEE) describes the reaction in a part or a system caused by the impact of radiation[2]. TID is a total amount of radiation that a part or a system is subjected to[2].

3.1.1.1 Total Ionizing Dose (TID)

Protons, heavy ions, and Galactic Cosmic Rays (GCRs) contribute to TID which is measured in "rad" units. TID affects Complementary Metal-Oxide-Semiconductor (CMOS) devices by creating new electrical paths and depositing charge within the component[2](e.g. in the oxide layers over the silicon[48]). Consequently, the electricity can flow in unexpected way.

In general, TID contributes to the device deterioration over time[34]. Despite the ability of some TID effects to self-anneal, the time needed to recover can be longer than a mission can afford. The OBC parts selection, parts derating, and shielding prevent potential catastrophic failures due to TID[5].

Using ESA SPENVIS[49], TID has been estimated for the typical mission lifetime. The mission lifetime is set to be three years (other simulator settings are described in Appendix A). Various possible mission scenarios (altitude, inclination, and shielding width) have been simulated and presented in Figure 3.4.

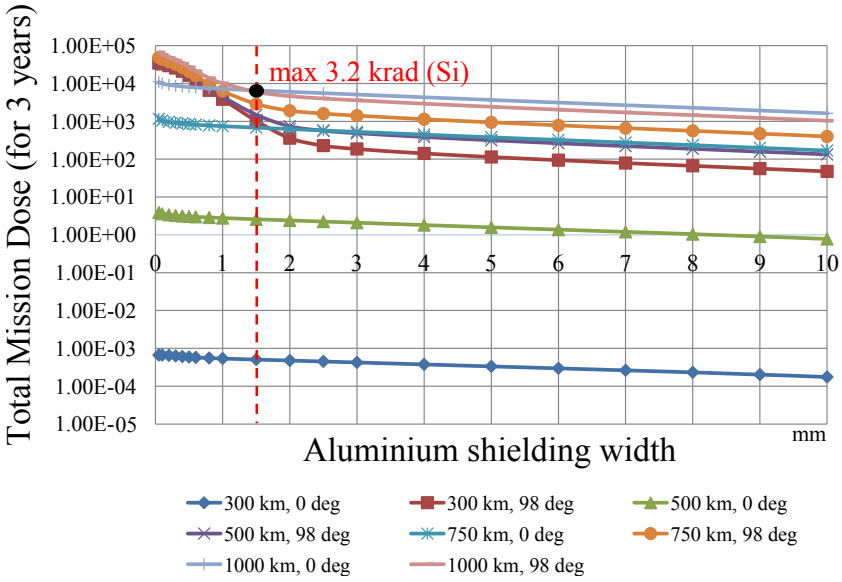

Figure 3.4: Simulation results: TID vs aluminium shielding width

Since 1.5 mm aluminium shielding is usually supported by a standard CubeSat structure, 1.5 mm is chosen as a shielding width for small satellites in this work. According to Figure 3.4, the worst case TID scenario for a 3-years mission within 300-750 km altitude corresponds to 3.2 krad. The correctness of this simulation result is supported by the empirical National Aeronautics and Space Administration (NASA) model APEXRAD [50].

3.1.1.2 Single-Event Effects (SEE)

SEEs (or SEPs) are the radiation effects that occur unpredictably with a range of consequences[34]. The consequences depend on the damage caused by the passage of a single high-energy particle through an electrical node[51]. Such particles are protons, heavy ions, and GCRs. Depending on the device electric field configuration, the created charge disposition can cause SEP or it can recombine without any effects [52].

Linear Energy Transfer (LET) is the rate at which ions or other particles loose their energy when penetrating the material[2]. LET is defined as energy deposited per traversing length per material specific density ($MeV \cdot cm^2/mg$)[53, p. 11]. Incident energy, particle mass,and material density influence LET and the results of penetration.

Using ESA SPENVIS[49] tool, the LET spectrum of particles inside the satellite shielding has been found for typical CubeSat missions. As previously, 1.5 mm aluminium shielding was taken as a case study. By comparing different orbital scenarios, the worst case was also found at 750 km and 98 deg inclination (Figure 3.5).

The particle LET spectrum can be considered as equal to zero when the LET is higher than 32.5 $MeV \cdot cm^2/mg$ (see Figure 3.5).

Single-Event Effects (SEE) classification

SEE can be classified to subcategories based on the type of consequences [2, 34, 33]:

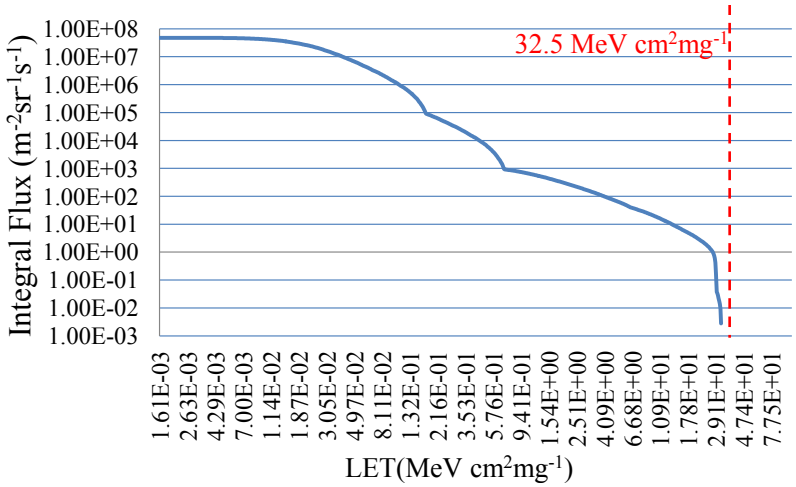

Figure 3.5: Simulation results: Spacecraft shielded LET(Si) spectrum

- Single-Event Upset (SEU), or "soft effect" according to [25]
- Single-Event Transient (SET)
- Single-Event Functional Interrupt (SEFI)
- Single-Event Latchup (SEL) , or "destructive effect" according to [25]
- Multiple-Cell Upsets (MCU) are topological multiple SEUs [54]
- Multiple-Bit Upsets (MBU) are logical multiple upsets: error bits belong to the same bit-word[54]
- Single Hard Error (SHE) or "hard effect", which denotes a stuck bit [25]

SEU happens when a single particle deposits enough charge at a sensitive node in a microcircuit to cause the circuit state change. SEU appears as "soft error" such as a bit-flip or a spurious command [52]. Thus, the data state of the device is corrupted without the permanent damage of the device. The nature of SHE is the same as SEU has, but the bit cannot be returned to the normal operation by re-writing.

SET corresponds to the ion-strike-induced transient pulse[48]. The pulse can propagate through the network and cause SEU in a storage element. Another SET effect is the interaction with the internal clock signal by widening/narrowing the pulse signal. Consequently, SET influences the speed of systems that are dependent on the clock signal[53, p.12] and may corrupt the synchronized communication between two nodes. The number of errors induced by SET is operating frequency dependant[55].

SEPs also incorporate SEFI that leads the electronic device to enter an unknown state and stop responding[56].

Ionizing interaction of CMOS junction with charged particles causes SEL[53, p.12]. Burnout and gate rupture are possible SEL consequences[5]. The component hangs up too in the case of SEL, but the part consumes excessive current and does not work until the device is power cycled[33].

3.1.2 Fault rate Assessment

Since the testing results about the error cross section are published for many COTS components, the rates of radiation effects occurrence(fault rates) can be calculated according to the methodology presented hereafter.

A differential cross section with respect to primary particle flux is defined as[57]:

$$\sigma(\Lambda;\theta,\varphi) = \frac{dN(\Lambda;\theta,\varphi)}{\Phi(\Lambda;\theta,\varphi)dod\Lambda}, \tag{3.1}$$

where do is the solid angular element, dN is the error number due to particles with LET in the range $\Lambda...\Lambda + d\Lambda$, $\Phi[cm^{-2}srerad^{-1}(MeV \cdot cm^2/mg)^{-1}]$is differential fluency per LET unit per solid angle.

Consequently, the full error rate can be calculated as:

$$R = \int\int\int \sigma(\Lambda;\theta,\varphi)\phi(\Lambda;\theta,\varphi)d(cos\theta)d\varphi d\Lambda \tag{3.2}$$

where ϕ is flux ($\phi = d\Phi/dt$)

Instead of the cross-section $\sigma(\Lambda;\theta,\varphi)$, it is possible to introduce the averaged cross-section $<\sigma(\Lambda)>$ over the full solid angle 4π. According to the usual assumption that the direction distribution of cosmic ray and corresponding LET spectrum are isotropic, the flux can be averaged: $\phi(\Lambda) \cong 4\pi\phi(\Lambda;\theta,\varphi)$

Then, the next formula can be derived from (3.2):

$$R = \int <\sigma(\Lambda)>\phi(\Lambda)d\Lambda \tag{3.3}$$

As a result, if at least some discrete values of cross-section measurement results and the flux simulation data are known, the fault rate can be calculated by polynomial approximation and the following integration. The corresponding code has been written in MATLAB to ease the fault rate calculation and presented in Appendix B.1.

3.1.3 Radiation effects in COTS components

Modern OBCs use different COTS components (see Chapter 2). Thus, the radiation impact estimation is required for the range of COTS parts. Hereafter, such estimations are presented; the fault rates are calculated according to the methodology of Section 3.1.2 and Appendix B.1.

3.1.3.1 SEE in DRAM

Two opposite trends are observed in DRAM memory[58]:

- the shrinking junction volumes that decrease the collected charge
- the relatively high node-capacitance due to an external three-dimensional cell capacitor

As a result, DRAM fault-tolerance has remained roughly constant over many generations. However, soft errors in the bitline and sense amplifiers become dominant because of the increased operating frequency[58].

The Table 3.1 contains the DRAM fault-tolerance information and its source:

Device/device component	TID, krad(Si)	Possible SEE	Cond.	Fault rate (worst 5 minutes)	Source
Latest generation of Synchronous Dynamic RAM (SDRAM)	50	SEU and SEL			[59]
Peripherals circuits of DRAM		SEU			[60]
64 Mbytes SDRAM (Micron company)	20	SEL followed by SHE SEFI followed by MCU	51 $MeV - cm^2/mg$		[61] [61]
64 Mbytes SDRAM (Elpida company)	40	no SEL even at: SEU SEFI followed by MCU	85 $MeV - cm^2/mg$	$1.4519 \cdot 10^{-8}$ upset/bit/day zero	[61]
Micron MT47H256M8HG-37E		SEU SEFI followed by the whole page or column corruption		$3.9306 \cdot 10^{-9}$ upset/bit/day $1.3373 \cdot 10^{-1}$ event/dev/day	[62]
Elpida EDE2108ABSE-8G-E		SEU SEFI		$3.6796 \cdot 10^{-11}$ upset/ bit/ day $1.5115 \cdot 10^{-2}$ event/dev/day	[62]

Table 3.1: DRAM radiation sensitivity - empirical data

As a result, DRAM memory is tolerant towards the expected total mission dose of 3.2 krad. According to the estimations, modern DRAM memory components will not experience SEL during usual small satellite missions. However, DRAM memory is

highly susceptible to SEU, SEFI, and MCU. If the OBC contains 32 Mbytes of SDRAM memory, then the memory will experience 4 upsets per day in the worst case.

3.1.3.2 SEE in SRAM

Beginning from the late 1990s, terrestrial neutron-induced soft errors have become the main reliability issues in SRAM industry [60]. Research [63] explicitly shows that due to technology scaling the SRAM memory is becoming less SEU sensitive but the ratio and the size of MCU are increasing.

The Table 3.2 contains the SRAM fault-tolerance information and its source:

Device/device component	TID, krad(Si)	Possible SEE	Conditions	Fault rate (worst 5 minutes)	Source
16-Mbit 130 nm SRAM		MCU growing with particle energy	Particle energy: 21, 46, 96, and 176 MeV		[64]
65 nm SRAM memory		45% of SEE - SEUs, 55% -MCUs; MCU is shorter 20 bits			[65]
45 nm Single port SRAM memory		SEU		$1.2777 \cdot 10^{-4}$ upset/bit/day	[54]
		no SEL even at:	60 $MeV \cdot cm^{-2}/mg$, 125 C, voltage 110% of nominal one		
65 nm Single port SRAM memory		SEU		$1.2788 \cdot 10^{-4}$ upset/bit/day	[54]
90 nm SRAM memory		SEU	800 km with 2.5 mm Al shielding	the same as in 2 previous cases	[66]
		no SEL even at:	117 $MeV \cdot cm^{-2}/mg$, voltage 120% of nominal one		

Table 3.2: SRAM radiation sensitivity - empirical data

As a conclusion, modern SRAM memory will not suffer from SEL and TID during small satellite missions. However, SRAM memory is more susceptible to SEU and especially MCU (more than 50% of all SEE). Taking into account that the amount of SRAM memory(e.g. in Micro-Processor Unit (MPU)) is usually limited by a hundred

of kilobytes, the expected SEU and MCU rate is about 105 upsets per day in the worst 5 minutes of the satellite mission. It corresponds to one upset every 15 minutes.

3.1.3.3 Flash memory radiation induced effects

Flash memory is a non-volatile, electronically erasable and programmable memory[67]. The basic storage element includes a control gate stacked over an isolated polysilicon gate in the gate oxide (named Floating Gate (FG)), a source, and a drain[68]. Internal charge pump generators provide higher voltages than external operating supplies for programming and erasing Flash memory . The generators are radiation sensitive parts of Flash-based memory[69] . Another vulnerable part of COTS Flash devices is the complex control circuitry(Finite State Machines (FSM), output buffers).

COTS NAND flash memory is widely used in commercial and space application as a mass storage device due to its high density, high I/O bandwidth, good retention properties, and non-volatility[68, 70]. According to the experiments [71], NOR Flash memory is much more prone to SEU and SEFI effects than NAND one. It can be explained by the relative simplicity of control circuit in NOR memory in comparison with NAND Flash memory.

Critical TID level is directly connected with the reduction of FG retention capability by heavy ions[70].

Typically SEFI corrupts the large part of the memory when the read/write operation is happening. Some SEFIs will self-recover once the device is re-read, other require a power cycle or even re-initialization to return to normal operations.

The Table 3.5 contains the information about the fault-tolerance of Flash-based memory.

As a result, Flash technology offers non-volatility and high density but limited in terms of number of writes cycles and TID(mainly because of the charge pump circuit). Therefore, Flash memory is usually used for long-term storage of critical system configuration data (e.g. programme code or FPGA configuration). Flash memory is usually kept unpowered to extend its life expectancy. The main source of faults in Flash-based devices is complex peripheral circuit.

3.1.3.4 FPGA components radiation induced effects

FPGAs are user-programmable devices that perform the functions of Large Scale Integration (LSI) circuitry. They include from thousands to millions programmable logic elements, each capable of performing any logic function[34]. FPGA reconfigurability, high performance, low cost [4], and relatively low-power consumption are fruitful properties that can be used to develop innovative space systems [72].

FPGA reconfigurability lets system engineers to use the most current configuration of FPGA-based processors and change their configuration during on-orbit stages[34, 73]. Moreover, FPGA enhances the OBC flexibility and adaptivity in terms of peripherals and functionality[24].

FPGA on-the-flight reconfigurability is based on the volatile (SRAM) or non-volatile(Flash) memory cells that store device programming information. These two types of memory technology (SRAM and Flash) separate reconfigurable FPGA devices

into two groups: SRAM-based FPGA and Flash-based FPGA. The third group of FPGAs is based on Anti-fuse technology and presents One-Time Programmable (OTP) devices.

The scaling decrease of Vcc alleviated the SEL susceptibility in newer devices[48]. Meanwhile, due to scaling, functional, and performance advancements of FPGA, SEU susceptibility is growing up.

Fundamentally, the radiation effects in FPGAs are the same as in any other CMOS-based digital IC, but differences of radiation sensitivities originate in switch types[48]. Overview of the FPGA groups for space applications are presented hereafter:

SRAM-based FPGA

Volatile configuration memory of SRAM-based FPGA provides high speed for reconfiguration and unlimited number of reconfiguration cycles[74]. On the other hand, such memory is sensitive to SEU, SET, and SEFI[72, 73]. When SEU hits the configuration memory cell, the logic or routing is corrupted[75]. Moreover, SET can be transformed into permanent SEU if it changes the configuration data[74, 48].Changes in the configuration memory can lead to SEFI of the whole component[72, 73]. Since both user and configuration memories are susceptible to SEU and SET, the whole OBC becomes very radiation sensitive [74].

However, the SRAM-based FPGAs are tolerant to TID up to 100-200 krad(Si) [73]. Moreover, their TID tolerance is becoming higher: from functionality loss at 50 krad level ten years ago[48] to 500 krad at Virtex-5 65 nm CMOS FPGA nowadays [53, p. 13].

Toshinori Kuwahara[53] evaluated the device failure rate due to radiation-induced effects at the altitude of 900 km with 99.03 degree of inclination. According to the results, Virtex-II family (XC2VP 50) has 7.4610^{-5} device/sec failure rate due to SEU in configuration memory and 2.1010^{-5} device/sec failure rate inside Block RAM (BRAM), the user SRAM-based memory. Thus, the configuration of SRAM-based FPGA is corrupted every four hours in space, when the user memory will have an upset approximately twice a day.

Flash-based FPGA

Non-volatile configuration memory of Flash-based FPGA is less sensitive to SEU since the FPGA uses FG switches that are more tolerant to charge injection [72, 76]. Flash switches are immune to SEP and only logic modules determine the FPGA sensitivity [48].

FGs directly control the programmable interconnection points and the logic blocks in Flash-based FPGAs[77]. However, a FG switch (e.g. in ProASIC3E FPGA) contains two NMOS transistors and their threshold voltage is determined by the stored charge[78]. Testing results showed that the induced charge can cause SET that may temporarily alter the correct circuit implementation in Flash-based FPGA[77].

Since each programming point is potentially sensitive to SET, a temporary change in the routing structure, implemented logic function, or in driven signal value can be provoked. SET can lead to multiple faults when the transient pulses are sampled by the

user memory elements (flip-flops and embedded SRAM blocks)[79]. But Flash-based FPGA configuration cannot contain permanent radiation effects, such as SEU[72, 4].

TID can lead to a variation in the threshold voltage of FG transistors disabling their reprogrammability [77, 78](see Figure 3.6).However, the most dangerous TID effect is the malfunction of the programming and erasing FPGA circuit caused by the radiation-sensitive charge pump (similarly to Flash-based memory)[70].

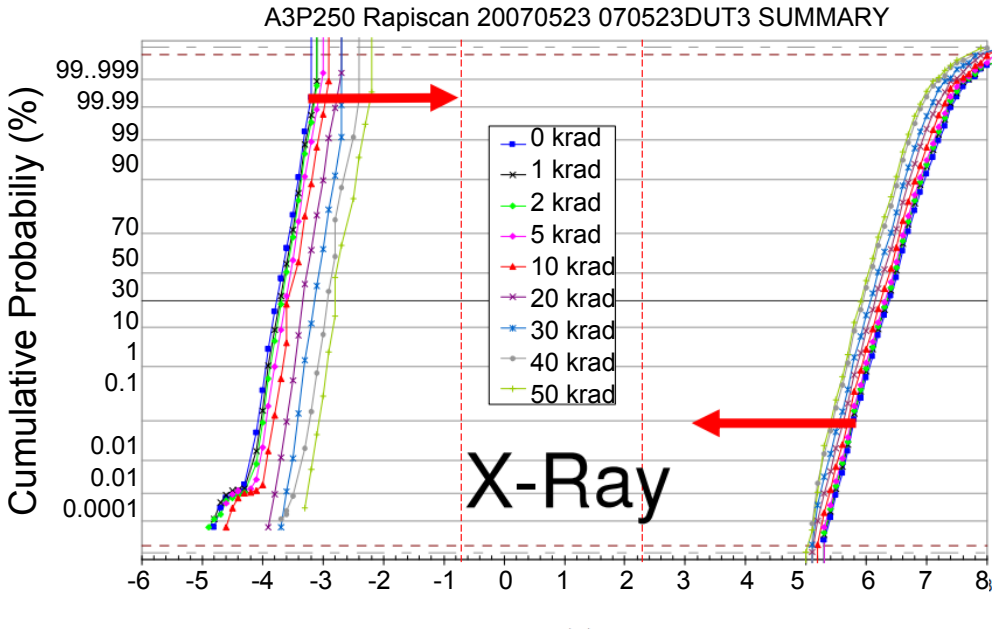

Figure 3.6: Threshold voltage shifting at different TID levels [3]

The Table 3.3 contains the information about the radiation sensitivity of Flash-based FPGAs.

In summary, logical mapping, placement, and routing of Flash-based FPGAs almost do not influence the device radiation susceptibility if TID $<$ 20 krad(Si). The propagation delay and transistor threshold voltage are slowly increasing until from 20 to 40 krad(Si). The majority of devices experience a functional failure at 40 krad(Si). User memory (flip-flops and embedded SRAM) can also be corrupted when SET are sampled by the user memory elements[79].

Anti-fuse FPGA

As opposed to previous two FPGA types, Anti-fuse FPGAs can be programmed only once. As a result, the configuration cannot be lost due to radiation effects[73].

TID tolerance of Anti-fuse Actel FPGAs is 100-300 krad(Si),depending on the device and lot. However, Antifuse FPGAs can loose functionality at 30 krad because of TID sensitivity of a charge pump circuit[48]. The sensitivity of the Antifuse devices is determined by the CMOS logic part.

While both Anti-fuse and Flash-based FPGAs are susceptible to SET and SEU in user memory, the price of Anti-fuse FPGAs are unaffordable for small satellite industry.

Device/ device component	TID krad(Si)	Possible SEE or other effects	Condition	Source
Flash-based ProASIC3	20			[53, p.13]
	50-60	delay degradation within 10%		[48]
	40	SET linearly increases with frequency	frequency >50 MHz(see Figure 3.7)	[4]
	30-40	functional failures		[80]
		SEL	68 $MeV \cdot cm^2/mg$	[81]
A3P600 ProASIC3	20-24	delay degradation $\leq$ 10%		[82, 83, 84, 78]
Flash-based ProASIC3, 130-nm	40 55	21% delay degradation; Icc growth, due to FG leakage current growth Icc is 200% of nominal one		[78]
		SET	$1.7298 \cdot 10^{-6}$ upsets/Logic Cell Like-Inverter (LCI)/day	

Table 3.3: Flash-based FPGAs radiation sensitivity - empirical data

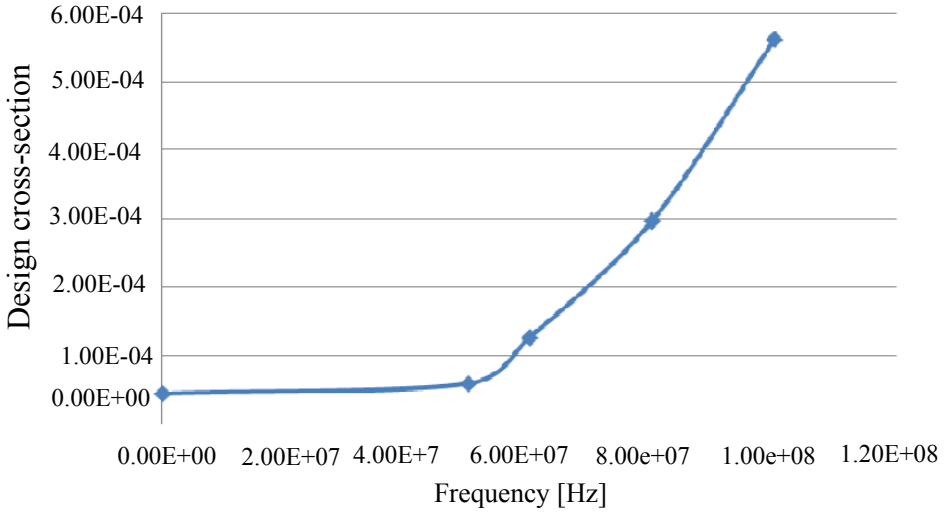

Figure 3.7: SEP sensitivity wrt frequency [4]

3.1.3.5 MPU radiation induced effects

SEU and SEFI threshold for commercial multiprocessors range from 0.2 to 9 $MeV \cdot cm^2/mg$ [85]; such low threshold causes from multiple upsets per day to a single upset per year.

SEU can occur in MPU causing data errors. MPU is also susceptible to SEFI[35]. In comparison with memory storage, SEUs are rare events because of MPU's small sensitive cross-section.

MPU contains General-Purpose Register (GPR) and Special-Purpose Register (SPR) and the corresponding consequences of their corruption are very different. SEU in program counter causes the wrong program flow, meanwhile the SEU in GPR causes the data corruption of the algorithm being executed.

Although, the probability of MPU SEP is small in comparison with memory faults, the number of mitigation techniques were created for MPU due to its importance. Since the cache memory occupies the majority of MPU area, proper mitigation techniques have to be implemented to avoid data corruption caused by cache SEUs and MCUs.

3.1.4 Conclusion

According to the aforementioned analysis, the majority of modern COTS parts are tolerant to the mission TID (see Section 3.1). Meanwhile, SEUs and MCUs stay the main issues in volatile memory when SEFI poses a thread in Flash-based memory .

According to the investigation of published radiation test results, the estimated TID and LET cannot lead to destructive consequences in COTS components(e.g. SEL or SHE). Thus, the presented fault models are focused on SEE (see Chapter 5).

The fault-rates (8.9) are calculated using the output LET spectrum from SPENVIS system and cross-sections from empirical observations; the worst case scenario is presented in Table 8.9.

Component type	Malfunction at TID (krad(Si))	SEL at LET ($MeV - cm2/mg$)	SEU rate (upset/ bit/ day)	SEFI rate (event/ device/ day)
DRAM	> 20	51	$1.45 \cdot 10^{-8}$	0.26
SRAM	20	117	$1.27 \cdot 10^{-4}$	not observed
Flash NAND	15	-	$1.38 \cdot 10^{-9}$	0.013
Flash NOR	10-20	-	tolerant	0.0013

Table 3.4: Radiation sensitivity of COTS components, the worst case

From radiation-immunity point of view, Flash-based FPGAs better suit small satellite applications in comparison with SRAM FPGAs. At the same time, it is almost inevitable to use hard Central Processing Unit (CPU) to provide high power-efficiency and general-purpose capabilities. Consequently, a heterogeneous SoC with the hard core and Flash-based FPGA fabric is a suitable choice for the OBC of small satellites. SmartFusion SoC[6] meets these two requirements and incorporates Cortex-M3 ARM hard core[86] and ProASIC Flash-based FPGA fabric [87].

Device/ device component	TID, krad (SiO2)	Possible SEE or worn-out	Cond.	Fault rate (worst 5 minutes)	Source
NAND		worn-out	(10^5)-(10^6) Program / Erase (P/E) cycles		[67]
4, 8 Gb 60 nm NAND		errors during P/E cycles	after 10^6 P/E cycles and 100 krad		[88]
Samsung 90nm	15-28	some blocks can be programmed again after 500 hours unbiased			[68, 2]
NAND	50				[89]
Other tests		failure of charge pump circuits 814 krad			[89]
Other tests		SEU in control logic which is SEFI during R/W cycles,			[68]
4 Gb NAND Micron		SEU SEFI		$1.3871 \cdot 10^{-9}$ upset/ bit/ day 0.0130 event/ dev/ day	[90]
8 Gb Samsung NAND		Single-Level Cell (SLC) SEU Multi-Level Cell (MLC) SEU SEFI		$5.1449 \cdot 10^{-10}$ upsets/bit/day $4.8235 \cdot 10^{-9}$ upsets/bit/day 0.0131 event/ dev/ day	[88]
8 Gb Samsung NAND		SEU		$5.1449 \cdot 10^{-10}$ upsets/bit/day	[88]
Modern NAND	600				[91]
64Mb NOR Spansion		Charge Pump failure SEU tolerant SEFI	51.5 $MeV \cdot cm^2/mg$	 0.0013 event/dev/day	[71]

Table 3.5: Flash memory radiation sensitivity - empirical data

3.2 Radiation fault-tolerance techniques

The OBC must be robust against the radiation-induced effects (see Section 3.1.3). Currently used space qualified computers are based on custom-built microprocessors that utilize radiation hardening techniques such as semiconductor process flow modifications. However, this approach results in the performance degradation, increase of power consumption and the overall cost of the chip[56]. Other OBCs are based on space heritage processors that still may provide robust and predictable performance in radiation environment [25]. However, the processors with space heritage have higher power consumption and may hardly meet the processing power requirements of future satellite missions in comparison with state-of-art COTS processors.

Kaschmitter et al.[5] described how radiation effects influence the design process of COTS-based systems (see Figure 3.8).

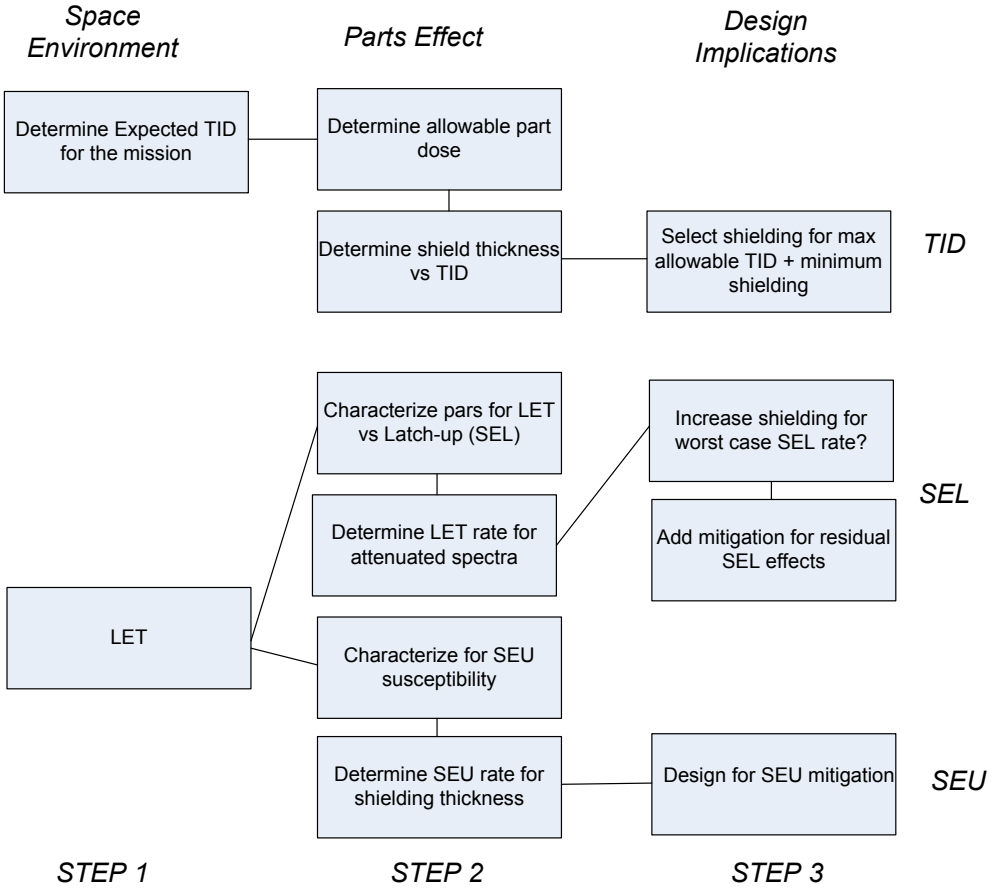

Figure 3.8: Design process for use of commercial parts in LEO [5]

First two step (see Figure 3.8) have been conducted in the previous Sections 3.1-3.1.4. The chosen Aluminium shielding width (1.5 mm) is enough to prevent SEL and significantly reduce SEE fault rate(see Section 3.1-3.1.2). This Section is dedicated toward the Design Implication according to Kaschmitter et al.[5], particularly to the overview of common radiation fault-tolerance techniques.

All radiation fault-tolerance techniques can be classified into three classes[9]:

- improved material hardness
- design techniques
- software methods, or Software Implemented Fault Tolerance (SIFT) techniques [92]

Since the radiation hardened components are not affordable for small satellite missions, this chapter will make an overview of Design and Software mitigation techniques. However, the most popular mitigation techniques (e.g. hardware or time redundancy) can be applied on different architectural levels (e.g. PCB level, component level, FPGA configuration level). Consequently, the system-level fault-tolerance techniques are described first in Section 3.2.1, then Section 3.2.2 is focused on the component-level mitigation techniques.

The presented overview of fault-tolerance techniques cannot be considered as the complete one. The overview is required to understand Chapter 8 where the efficiency of fault-tolerance techniques is investigated (see Chapter 4).

3.2.1 Architectural-level techniques

3.2.1.1 Shielding

Shielding is the engineering technique of providing radiation protective material between at-risk parts and incoming radiation. One of the main disadvantages of this approach is shielding mass that considerably increases the launch cost[2]. However, the shielding can be performed on the satellite level, subsystem level, or component level. Shielding is a countermeasure to TID effects; however, it does not guarantee SEP prevention[35].

3.2.1.2 Resetting, Power Cycling by Watchdog Monitoring

Watchdog capability can provide resetting and power cycling on the microcircuit, component, and/or system level in the case of its faulty operation or freezing(hang)[25]. Power cycling and reset are proven state-of-art methods of component and system recovery from SEFI[56, 93, 61]. A drawback of these techniques is that the processing unit, microcircuit, or the whole system is unavailable and inoperable during the reset. Instead of power-cycling, customized interrupt handlers can be used; however, the recovery through interrupt or even reset is not always successful[56, 61].

3.2.1.3 Windowing

Frequent erroneous communication requests of one component can cause faulty behavior of the whole OBC (e.g. if the communication requests constantly cause interrupt processes). The recipient device can close communication channel to the faulty device to prevent wasting resources[35]. Another form of Windowing is described in [94] where a RTOS task is considered to be faulty if its execution takes more time than predefined.

In general, windowing techniques can be described by the statement: an event is considered to be faulty if the event is happening outside the pre-define time period or

is utilizing hardware resources it should not normally use. This principle is used in Greenhills RTOS [95] that protects memory by memory pool restrictions to particular tasks[32].

3.2.1.4 SEL mitigation

Quick and reliable detection of SEL is necessary to prevent burnout and gate rupture. SEL can be detected by monitoring power consumption of the chip[53]. If anomalous current is detected, the component/sub-systems should be power cycled[35, 33] or separated from the power line [53, p.12].

One of the obstacles of reliable SEL detection is high dependency of current consumption of CMOS logic from processing load. Thus, one of the ways to detect SEL is to place the processor in a known processing state and sample the power-supply current[5].

3.2.1.5 N-modular redundancy

N-modular redundancy can be taken as a systematic technique and be applied to the processor cores, the memories, FPGA co-processor configuration, and the whole subsystems. But the significant hardware overhead has to be taken into account.

TMR is the the most popular sub-type of N-modular redundancy. In the majority of cases, TMR is performing voting mechanism based on the assumption that only one fault a time can occur[33, 60, 19, 25].

Two-modular redundancy can be utilized to determine the correct result of instruction execution and mitigate SEU. However, a software routing has to handle the detected fault in this case. This approach can be named "backward error recovery with recovery point" technique[5, 94, 93].

3.2.1.6 Time redundancy

Time redundancy means performing the same computation over and over again until the confidence in the result validity is gained[56]. The drawback of this method is long-term resource occupation to verify that the result has not been corrupted by SEP.

SET can be detected by time redundancy with subsequent output comparison. Time redundancy can be implemented on the task level by the programmer[96](or Operating System (OS)) and on the instruction level during program compilation[97].

3.2.2 Component-level techniques

This Section is focused on component-level fault-tolerance techniques.

3.2.2.1 Mitigation techniques in MPU

The MPU protection from SEFI consequences can be performed by the MPU reset and power-cycling. MPU can be reset by External or Internal watchdog as well as by an exception handler process[35].

Errors in the cache memory can be recovered by Error-Dectection And Correction (EDAC) that identifies SEU and rewrites the radiation affected cache blocks[5]. MCUs also can happen in modern 90-nm electronics (see Section 3.1.3.2). Thus, the additional fault-tolerance analysis is required to be sure that cache MCU are properly mitigated and will not cause significant performance degradation.

Software mitigation techniques

Software mitigation techniques are an inherent part of the OBC software. They are implemented to detect and recover from the radiation-induced faults.

Multiple independent but functionally equivalent software implementations help to catch software design fault as in the case of N-version programming [98] and recovery blocks[99]. Control flow faults can be detected by a number of techniques that are based on the control flow signatures [100, p. 30].

Time redundancy on assembly instructions, procedure calls, and program levels [100, p.41] can detect and fix the SEU induced during the program execution.

Illegal memory reference technique[101] is also a typical software fault detection method, but it can be considered in Windowing Section(Section 3.2.1.3).

Critical recovery information must be kept in non-volatile memory preferably in several copies to provide fast software recovery process("warm" start) to increase the OBC availability [5]. Another approach is to keep the recovery information in storage that is not power cycled, but in this case the storage have to be SEL immune.

3.2.2.2 FPGA Reprogrammability and MPU Upgradability

Hardware reprogrammability of FPGA components and MPU software upgradability can be considered as crucial properties for the survival of future spacecrafts in ultra-long-life missions [102].

The main issue in software upgrade process is the performance loss, the OBC unavailability, and the potential mission-failure risk caused by interrupt services of upgrade procedure[102].

Hardware reprogrammability of FPGA requires a new configuration uploaded from the ground station. The configuration file can be big, hence the upload process may take several satellite orbit periods. For example, Xilinx Virtex 1000 requires 6 Mbytes of configuration and four orbit rotations are required (as minimum) to upload the configuration file.

On the other hand, reprogrammability and upgradability give an opportunity to update the software and FPGA configuration with newer, optimized, and bug-less versions.

3.2.2.3 Mitigation techniques in Memory

Memory scrubbing is a background function of reading memory locations. If SEU or MCU was detected, the data is corrected. The detection and correction can be performed only if extra bits are allocated for this purpose[103]. Hamming code[25] is one of the most popular linear error-correcting codes that are used nowadays, e.g. in Power PC

SRAM. EDAC with sufficient scrubbing frequency prevents SEU accumulation and SEU consequences[33, 5]. However, MBU cannot be corrected by simple Error-Correcting Code (ECC) scheme since the bits in error belong to the same bit-word [54].

As it is observed in [97], EDAC techniques and used periodic memory scrubbing in DRAM improved the availability of the COTS board significantly: continues running for a month with EDAC and a few days without this technique.

3.2.2.4 Mitigation techniques in FPGA

The Joint Test Action Group (JTAG) circuitry of all COTS FPGAs is not radiation tolerant . Thus, FPGA test logic reset must be hardwired to the ground to provide immunity to JTAG SEUs [73].

SRAM-based FPGA

SRAM-based FPGA are tolerant enough from TID point of view (100-300 krad usually). To mitigate SEP effects hardware redundancy [104] and/or time redundancy[79] are employed: the design modules are replicated and SEP-filtering modules are added to FPGA configuration.

TMR is the most widely used SEU mitigation technique[72]in SRAM-based FPGA design. But at least 3.2-4x hardware overhead should be expected (triple logic replication plus voters)[104, 73, 93]. Moreover, the maximum circuit performance is decreased at least by 10-20% when using TMR since the voters have to be inserted in the system critical path[93, 73].

Due to SRAM-based FPGA susceptibility to SEUs, the OBC has to include a configuration memory scrubbing circuitry for fault correction in the configuration memory [74, 93]. According to the recommendation of NASA[73], memory scrubbing should be performed at 10x more frequent than the expected upset rate. Some Xilinx and Virtex FPGA already have such scrubbing capabilities, others can use small Actel Flash-based FPGAs for this purpose[73, 34].

Consequently, TMR utilization, configuration memory scrubbing, and additional support from such parts as Programmable Read-Only Memory (PROM)(to keep original configuration data), and watchdog timers are highly recommended to make SRAM-based FPGAs SEU-tolerant [73].

Flash-based FPGA

The configuration memory of Flash-based FPGAs is immune to SEP (see Section 3.1.3.4). But the combinational logic is SET sensitive while the sequential logic (D Flip-Flop (DFF)) could have SETs and SEUs.

The SET and SEU susceptibility can be mitigated by time redundancy tuned to the width of the induced transient pulse [4].

Since SET is a pulse signal with very short width that propagates trough the combinational logic, it can be filtered by the delay elements introduction. A SET filter consists of SET transition delay(e.g. an inverter chain) and a Guard Gate (GG) that filters out SETs. But the main condition of SET filtering out is that the delay has to be longer than the pulse width [53].

For SEU mitigation TMR principle can be applied: each combinational logic and following DFF are triplicated.

SET filtering is more attractive for large combinational logics because it will not take so much hardware recourses as TMR. However, SET filtering can introduce three times higher performance penalty than TMR[3, p. 26].

3.3 Conclusions

This chapter gave an overview of space radiation environment for typical small satellite missions and explained the possible radiation effects in COTS parts. The overview of fault-tolerance techniques is presented in Section 3.2.

The verification that the mitigation techniques are working can be done through radiation test that imitates the space environment. However, radiation tests are expensive for small satellite industry. The comparative analysis between mitigation techniques is impossible due to low interpretability of radiation test and uncontrolled fault-injection procedure.

This work presents a simulation-based approach to solve the aforemention problem of the fault-tolerance techniques comparison and the fault-tolerance analysis in general. The next Chapters 4-5 will be dedicated to the explanation of the proposed approach.

4 OBC modeling

Previous Chapter 3 gives an overview of possible radiation effects in COTS electronics (Section 3.1.3) and observes existing fault-tolerance techniques (Section 3.2). This chapter proposes the SystemC-based simulation framework that can be used to determine the influence of the radiation effects on the OBC functionality and compare the efficiency of applied fault-tolerance techniques.

4.1 SystemC TLM modeling

As it was explained in Section 1.1, the radiation tests cannot be used for the comprehensive fault-tolerance analysis of the OBC. The utilization of high-level abstraction modeling language, such as SystemC, is imperative in any simulation-based approach due to high complexity of each electronic component and the OBC as a whole[105].

SystemC is a system-level modeling language that utilizes a mixture of various abstraction levels[106]. SystemC-based design platforms take advantage of Intellectual Property (IP) reuse to decrease the design complexity, development cost and time.

The requirement on the hardware-software co-design and co-simulation can be met with TLM methodology[11]. The TLM goals is to reduce the modeling complexity and increase the simulation speed while keeping enough accuracy for the design task. The TLM function-communication separation allows a breakthrough in verification time[107, p. viii-ix], which gives an opportunity to incorporate the fault-robust design at early development phases.

The requirement on the advanced OBC fault-tolerance makes FMEA essential to reveal and reduce the OBC vulnerability [108]. FMEA methods can be applied at different abstraction levels: Register-Transfer Level (RTL) [109], TLM, etc. However, the RTL application domain is limited due to the time and efforts spent on such detailed models. Moreover, the RTL modeling is unapplicable for the OBC due to the unknown internal structure of the COTS parts. SystemC TLM modeling level does not have these disadvantages since it operates with the higher level of abstractions.

Based on the fault injection procedures of FMEA, the group of faults with significant influence on the system behavior can be identified. Such an approach helps a designer to focus on the most crucial failure cause and reduce resource investments[110].

4.2 Related Works

The simulation approach for the fault-tolerance analysis has already been used for satellite sub-systems in works [111, 112]. These works investigate the fault-tolerance through the fault injection into system models. However, the works are based on

circuit-level knowledge that are not available for the COTS parts. Moreover, the low-level simulation is time-consuming; hence, it cannot be used for an extensive statistical analysis. Since CubeSats utilize COTS parts, all electronic components should be considered as black boxes.

A common existing SystemC approach to analyze fault effects is based on the insertion of Fault Injection Module (FIM) into the interconnections of the functional blocks. FIM plays a role of the fault injection controller that can be centralized [113, 114] or distributed [115, 110, 116]. No modifications to the SystemC model source code are required when FIM is introduced. However, such approach has been applied without hardware-software co-design, fault-tolerance techniques analysis, and a comprehensive fault model library.

The works [116, 110] use the TLM SystemC modeling with faults injection to research only SoC-scale systems. In the case of the satellite OBCs, the SoC level stays an essential part of the design modeling but not sufficient to estimate the efficiency of fault-tolerance techniques and identify possible hazards. The OBC system consists of several components with different possible faults, so the simulation capabilities have to cover both software and diverse hardware OBC components.

This work proposes an innovative statistical method to analyze the fault-tolerance of the COTS-based OBCs and compare hardware or/and software fault-tolerance techniques' efficiency at early development stages.

4.3 OBC architecture modeling - SmartFusion SoC model

A SoC with CPU and FPGA is chosen as a core component of the OBC model to cover the most general OBC architecture. The SmartFusion SoC [6] (see Figure 4.1) is one of the most appropriate candidates for the OBC (Section 3.1.3). It meets the requirements on the performance, power consumption[117], and peripherals (see Section 2.3). It includes Flash-based FPGA that is more immune to radiation effects than SRAM-based FPGAs [118].

The SmartFusion SoC incorporates the majority of on-chip subsystems that can be found in the advanced COTS electronics: a hard-core ARM Cortex-M3, Flash-based FPGA fabric, Embedded Flash and SRAM memories, Analog subsystem, AMBA bus, etc. As a result, the examination of the OBC with the SmartFusion device is a good basis for the comprehensive analysis of other OBC configurations.

The corresponding model structure of the OBC is presented in Figure 4.2. The model structure includes the above-mentioned ARM Cortex-M3 CPU, memories, the central bus (the Decoder plays the role of the central AMBA bus), FPGA fabric, a watchdog, and timers. TLM is used for the interconnection (with target and initiator sockets) between functional blocks (CPU, FPGA, Memory blocks, Timers, Decoder as a central bus, etc.). The functional blocks can be considered as "black boxes" since their full configuration in COTS parts are unknown for satellite designers.

The existing instruction-accurate model of the processor ARM Cortex-M3 has been obtained from OVP project[12]. The choice of OVP is dictated by two main factors:

1. Besides the model of Cortex-M3 processor, OVP offers a range of models for other

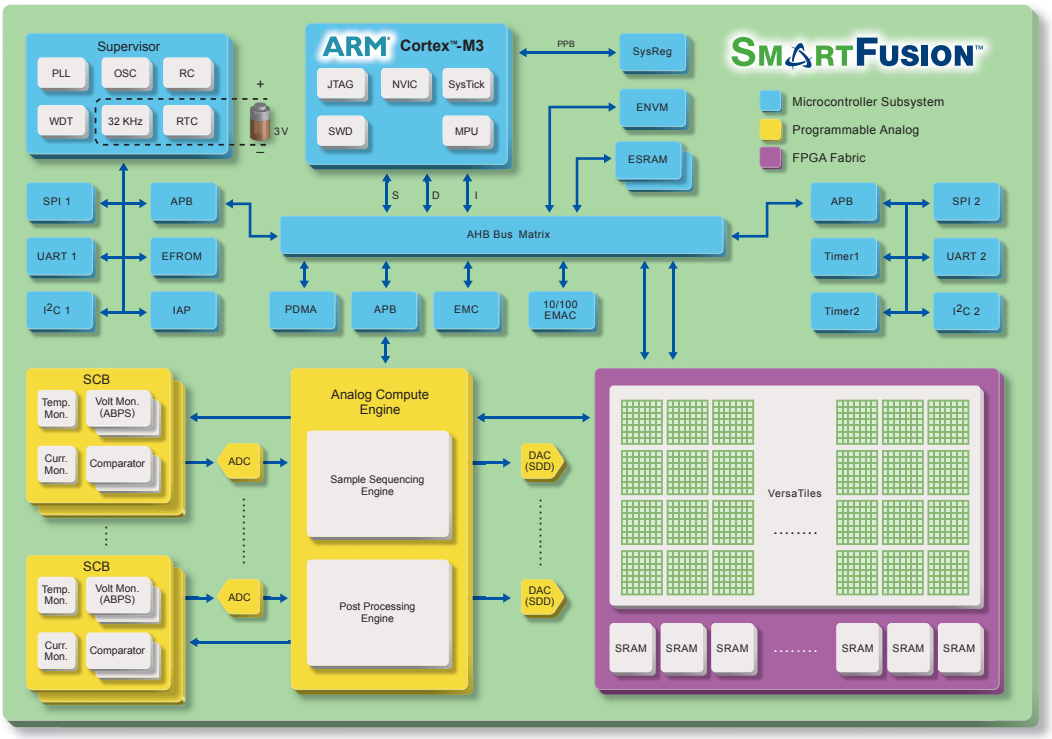

Figure 4.1: SmartFusion SoC Block Diagram[6]

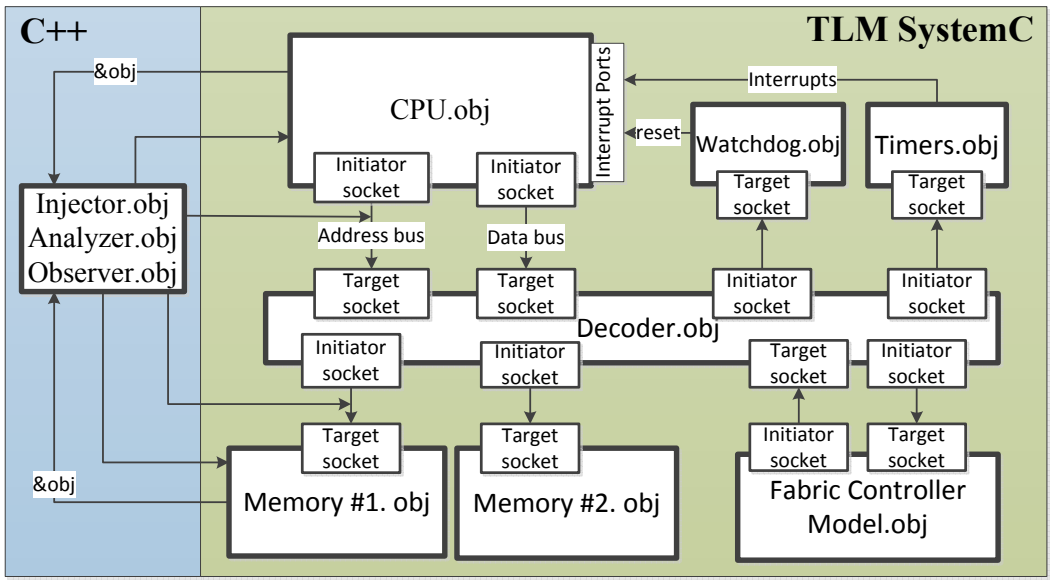

Figure 4.2: Framework model structure: OBC model and supporting modules

modern popular CPUs. It provides high flexibility for the OBC model since the processor can be easily replaced.

2. OVP provides SystemC TLM wrapper[119] for the ARM Cortex-M3 model which

makes its integration process into the OBC system model easier and faster.

Although the SystemC TLM wrapper is given, it has been changed to provide fault injection procedures and higher model observability.

The Decoder object plays the role of the central AHB AMBA bus connecting all other SoC components by initiator and target sockets. It translates the addresses and transfer the data between the functional blocks. The Decoder has an ability to introduce communication delays as well as faults.

The Memory object represents a memory storage. One of the parameters that is required for the memory object creation is the memory type (e.g. Flash, SRAM) that automatically defines the type of possible memory faults. For SRAM and DRAM memory types the physical layout characteristics(number of rows and columns) has to be specified for two-dimensional MCU introduction procedures.

The Watchdog and the Timers are connected to the Decoder as slaves but they are also connected to the CPU as signal initiators.

The Watchdog has a configuration registers according to the SmartFusion specification[86, p.163-171]. It is connected to the reset port of the CPU and clocked at 100 MHz. When the Watchdog timeouts, the reset signal is raised and the simulation of the CPU power-cycling (or reset) procedure happens. The Watchdog object also signalizes to volatile memory (the volatility is set during memory object creation) when the memory has to be flashed because of the simulated power-cycle. If the OBC has an external watchdog timer too, the same Watchdog object class can be used to instantiate the external watchdog.

The Timers.obj corresponds to two System Timers of SmartFusion device [86, p.302-304] and provides interrupt capabilities. The Timers.obj is connected to the Decoder as a slave and to the CPU interrupt ports as a signal initiator.

Although Figure 4.2 mainly represents the SmartFusion architecture, off-chip components (e.g. external SDRAM) can be also simulated by their connection to the Decoder. However, the communication delay of such components should be higher than on-chip components have.

Except SmartFusion device, the OBC incorporates external 32 Mbytes SDRAM, Flash SD-CARD as a memory storages. 32 Mbytes of SDRAM can be addressable through SmartFusion memory map [86, p.20] beginning from the address 0x601D0000, when SD-CARD is connected through SPI controller that can be a separate object in the model.

The next Section is dedicated to the integration of the FPGA-fabric model to the OBC model. Such integration plays an essential role in software-hardware co-design and co-simulation.

4.3.1 FPGA fabric modeling as a SoC component

SmartFusion device has an embedded Flash-based FPGA fabric. The FPGA significantly increases the flexibility of the OBC and expands the design space of possible fault-tolerance techniques.

One of the prospective approaches is to outsource the fault detection or/and mitigation functionality to the FPGA. For instance, the FPGA can make copies of the data in

radiation vulnerable SRAM/DRAM memories and continuously compare them. Thus, the FPGA can incorporate the memory scrubber functionality. Another example of such functionality is EDAC techniques e.g. EDAC based on Hamming code (see Chapter 8).

However, the portability of the written software (and the FPGA configuration) is reduced if the communication protocols are modeled with a high level of abstraction (e.g. TLM-based). The intermediate layer "Fabric Controller Model" has been introduced between the Decoder and the modeled FPGA fabric to keep the TLM simplicity and make the FPGA co-processor portable to the real hardware (see Figure 4.3).

The Fabric Controller Model (see Figure 4.3, red zone) is the TLM wrapper of the synthesizable RTL SystemC FPGA configuration (see Figure 4.3, blue zone). The portability of the RTL FPGA configuration to the real hardware is guaranteed by the Fabric Controller Model because the Fabric Controller Model operates with the signals according to the protocol standards (AMBA AHB and FIFO).

Figure 4.4 presents the process when one byte is read and written to two other locations by the FPGA-based co-processor according to the AMBA AHB protocol.

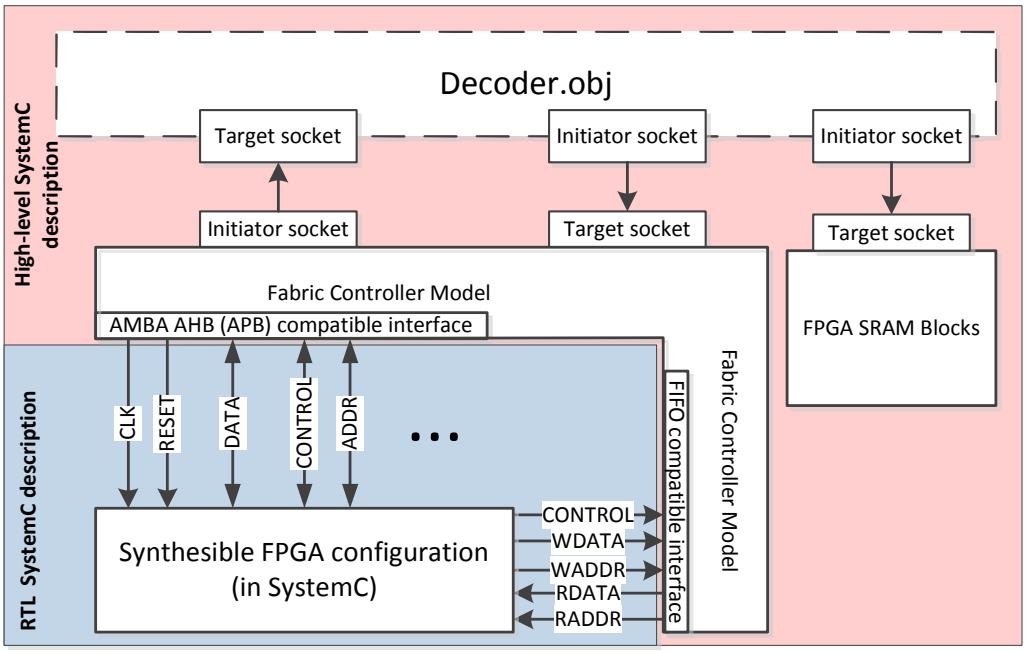

Figure 4.3: The FPGA fabric model as a part of the OBC model

The FPGA fabric has Dual-Port SRAM blocks (Figure 4.1). This memory is accessible through the central AMBA AHB bus and through the Embedded FIFO Controller from the FPGA fabric [87, p.22]. The direct access to on-chip SRAM blocks through the FIFO controller is again modeled by the Fabric Controller Model (Figure 4.3). Figure 4.5 presents the read and write procedures that take one clock cycle.

As a result, the OBC model incorporates the communication channels that correspond to the real ones. The hardware-software co-design approach was introduced: the CPU software and RTL SystemC FPGA configuration are simulated together and keep the portability to the real hardware.

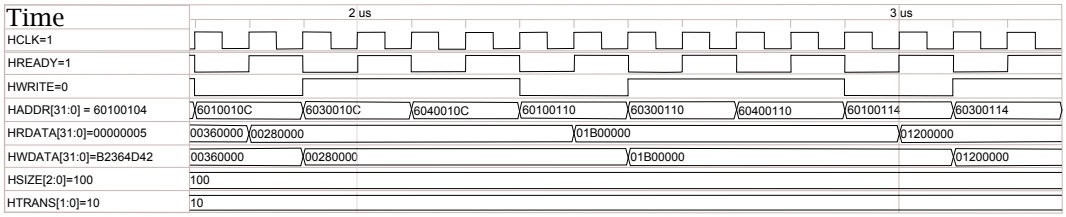

Figure 4.4: AMBA AHB protocol consistency for single write and read cycles

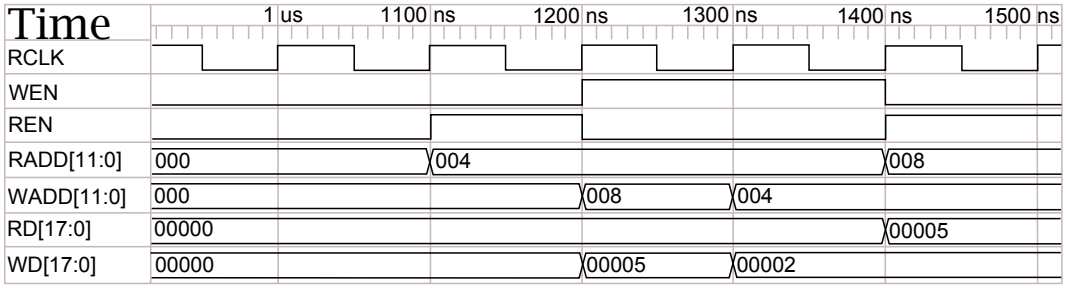

Figure 4.5: Example of the communication between the FPGA model and modeled SRAM blocks

4.4 OBC system-level redundancy - I2C communication

This Section is dedicated to the stacked OBC modeling and I2C interconnection modeling. The stacked OBC is a computer system that contains two or more autonomous OBCs. The stacked OBC can be considered as a typical example of hardware redundancy. The central bus for CubeSats is I2C bus according to the CubeSat standard. Thus, I2C controller and I2C bus modeling are the central topics of this Section.

4.4.1 OBC model extension by I2C controller model

The previous Section was dedicated to the OBC modeling (Figure 4.2). Figure 4.6 represents the extended model of the OBC. The I2C.obj is added to the OBC model and connected to the next OBC functional blocks:

1. to the CPU.obj, particularly to the CPU interrupt port to provide interrupt capabilities that are required for the I2C communication
2. to the Decoder.obj: the I2C configuration and data registers are included in the model memory map according to the SmartFusion documentation[86]
3. to I2C Decoder.obj, which is similar to the Decoder.obj of the OBC. Additional properties are added: e.g. pull down that corresponds to the I2C bus occupation

This object I2C.obj represents the I2C controller in the real device. The I2C controller can be replaced or added by any other controller for networking, such as SPI or CAN.

The model is written in SystemC TLM. The target and initiator sockets represent the communication peripherals whose internal structure is unknown. While the I2C.obj

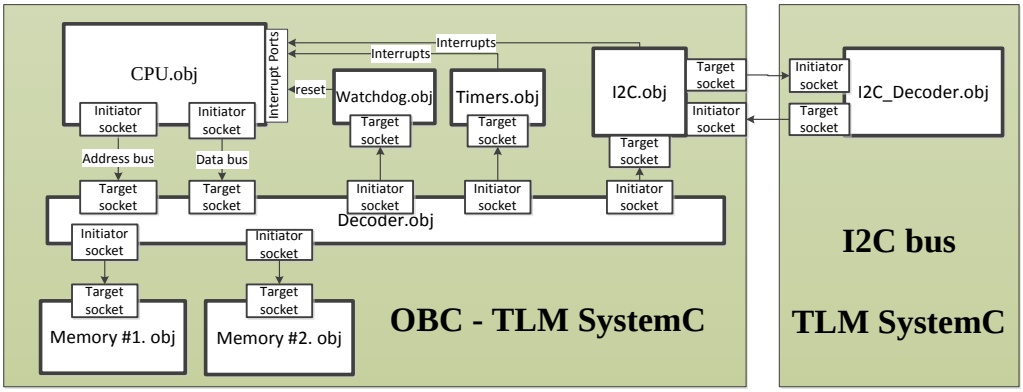

Figure 4.6: Model of the OBC connected to I2C CubeSat bus

represents the I2C controller, the I2C.obj is built as a FSM that corresponds to I2C driver. The original I2C driver from Microsemi Corporation is used and guarantees the software portability to the real device.

4.4.2 Model of interconnected OBCs (stacked OBC)

Figure 4.7 shows the simple interconnection model of two OBCs (the I2C Decoder plays the role of I2C bus). The I2C Decoder does not limit the model to two computers. The number of computers connected to the Decoder can be much more. Of course, the OBC can have more than one I2C.obj (a I2C controller/interface in reality). Thus, the modeling of internal satellite networks can be done by the creation of separate I2C Decoder objects.

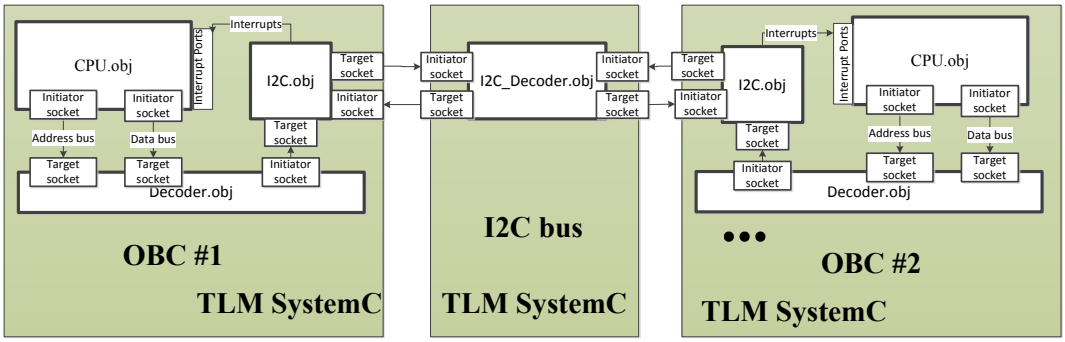

Figure 4.7: Model structure of two interconnected OBCs through I2C CubeSat bus

The FSM of the I2C.obj has been created based on the I2C controller datasheet [86]. The correctness of the I2C controller modeling is proved by the data sending/receiving simulations (see Appendix C).

4.5 Supporting modules

The simulation framework has three interrelated supporting modules: Injector, Analyzer, and Observer (see Figure 4.2). Figure 4.8 shows the main functionality of these three modules and their relationship to the OBC model. Each of them will be discussed in the next subsections.

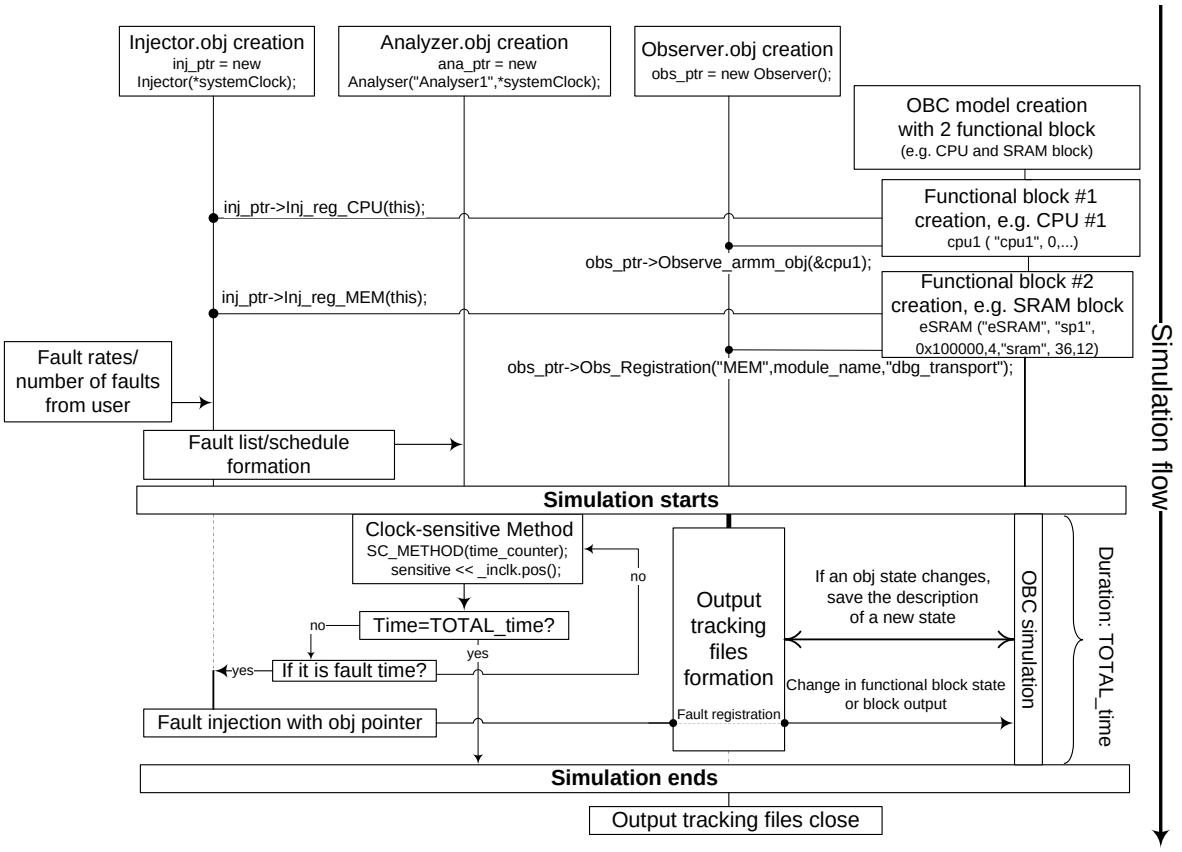

Figure 4.8: Functional diagram: OBC model and supporting objects in the simulation flow

4.5.1 Injector

The fault injection module or the Injector (see Figure 4.2 and Figure 4.8) is responsible for the fault list/schedule creation, fault injection, and fault registration in the Observer object. It contains the library of the predefined fault models that is used to create the schedule of the induced faults per component(see Section 4.9). The Injector takes the input parameters that define fault rates for each fault type.

The fault injection technique implemented in the proposed framework mainly operates with object pointers. When any functional block is created (e.g. SRAM block or a CPU block as in Figure 4.8), it sends its object pointer to the Injector (by calling the dedicated procedure of the Injector object). Consequently, when the OBC model is initialized, the Injector associates the available fault models from the predefined fault library and the components (functional blocks) where particular faults can happen.

The fault rates or the number of faults are defined by a user (see Figure 4.8). The Analyzer creates the fault list/schedule based on this information.

The location of the fault (memory cell, CPU register, particular bit, etc.) is chosen randomly for the fault list. The seed of randomization algorithm is saved into the report file. This seed is necessary to reproduce the same experiment to debug the OBC and fault models as well as to investigate the system behavior with more details for particular cases (details are provided by the Observer).

The fault probability distribution in time is assumed to be uniform due to the neglectable difference of the radiation conditions that can happen during the execution time of any OBC algorithm. The fault locations are also chosen with uniform probability distribution within the components of the same type(e.g. SRAM memory block, Flash memory) since the expected fault rates for different components are different (see Section 3.1.3).

The fault-selection criteria are introduced to let a user limit the space of possible faults. These criteria for fault time and fault location are used when a particular component or a particular time moment is under investigation.

```
36346:  NS:  SEFI:  CPU:  cpu1:                          (time:err_type:Module_name)
52232:  NS:  SEFI:  CPU:  cpu1:                          (time:err_type:Module_name)
163147: NS:  SEU:   CPU:  cpu1:   9:     14:             (time:err_type:Module_name:register:bit)
494789: NS:  SEU:   MEM:  eSRAM:  3948:  4:               (time:err_type:Module_name:address:bit)
```

Figure 4.9: Example of the fault list generated by the Injector module

As Figure 4.9 shows, the fault list consists of the description of the fault time (in nanoseconds), the fault type, the location where the fault occurs. For example, Figure 4.9 describes four faults: two SEFI and two SEUs (or bit-flips). The first SEU happens in the 9th GPR register, the 14th bit; the second SEU happens in embedded SRAM (eSRAM) memory storage in 3948 byte, the 4th bit. Due to the simplicity of the fault list it can be written manually by a framework user.

When the OBC model and the fault list/schedule are built, the simulation of TOTAL_time can be started. The simulated time is limited because the OBC works according to particular time schedule and the case with infinite available time does not correspond to reality.

According to the signal from the Analyzer object, the Injector introduces the fault into the simulated OBC and reports about the fault injection to the Observer that includes it in the report file.

In summary, the Injector is the library of fault injection procedures that are executed according to the Analyzer's signal and the fault list/schedule. If no input parameters are specified during the program execution, the Injector does not introduce fault, the fault list stays empty, the Analyzer does not send any signals, and no fault injection occurs.

4.5.2 Analyzer

The Analyzer is an object that has a method sensitive to the System Clock (SC_METHOD(time_counter), see Figure 4.8). The System Clock has been set to 1

ns on the assumption that there is no modeled OBC components with working frequency more than 1 GHz.

The Analyzer takes the fault list created by the Injector and compares the current time with the time of the next scheduled fault. If the Analyzer detects that a fault is planned this cycle, it sends all necessary data to the Injector to introduce the fault. In comparison with the fault injection unit in [115], the created centralized injection unit is not checking the whole fault list each cycle to decide about the fault introduction. The Analyzer keeps only the time of the next fault event.

The Analyzer also finishes the simulation when the predefined period of time, TOTAL_time ns, has been simulated.

4.5.3 Observer

The Observer object has a capability to trace signals, components states, fault introduction, and save all these information in separate files. For example, the Observer saves the memory accesses, instruction execution, and register values. The Observer also interleaves the tracking information with the time and the location of the fault injection.

Since the simulation time is increased when the Observer writes a lot of tracking information to files, the amount of the saved tracking data should be minimized.

4.6 Conclusions

This Chapter observed the main principles for the OBC modeling and the framework operation. The simulation framework for the OBC fault-tolerance analysis was discussed :

- The simulation framework should be parallelizable and fast enough for the extensive simulation-based statistical analysis and repetitive simulation runs. This requirement is mainly satisfied with high-level TLM SystemC modeling
- The portability of the written CPU software and the FPGA configuration is guaranteed by the model design: the intermediate TLM wrappers are used for RTL SystemC FPGA configuration and original vendor drivers and compilers are used.
- High-level system/sub-system models can be built based on the information from the device datasheets, as it was done with I2C controller (I2C.obj)

SmartFusion SoC is chosen as a case study for the OBC design since the SoC represents the most general architecture that incorporates different memory types (volatile, non-volatile), FPGA, a hard ARM Cortex-M3 CPU, and extensive range of the required peripherals. Moreover, SmartFusion SoC meets the OBC requirements formulated in Section 2.3.

The stacked OBC model was presented as an example of a multi-processor redundant computer system. The explanation of the top framework object is presented in Appendix D

5 Radiation effects modeling

This Chapter describes how the radiation effects (Section 3.1) are modeled and integrated to the simulation framework alongside with the OBC model(Section 4.3).

It is also necessary to distinguish the fault meanings/definitions in the contest of system defects [46, p.57-58] and in the contest of radiation space environment (see also Section 2.2.2). A radiation fault can be defined as the radiation cause of changes in the system structure or parameters that eventually leads to a degraded system performance or even system functional loss[45, p.1-2]. The terms "radiation fault model" or "fault model" will be used in the meaning of the representation of a radiation effect at the abstracted function level.

5.1 Radiation fault modeling

Since the internal structure of COTS parts are unknown, the radiation effects can be modeled by changing the output or/and the state of a simulated functional block.

- The change of block outputs is done at TLM initiator sockets. Particularly, TLM transaction objects (e.g. *tlm_generic_payload*) is changed upon the fault introduction. TLM transaction objects correspond to the data saved in memory cells in real hardware.
- The change of the block state is limited by the knowledge about the real component the block corresponds to. In the case of the CPU block, the block state change corresponds to the register change; in the Memory block - to the change of its memory array. However, the knowledge about the control logic of COTS parts are limited and the creation of dedicated fault models for the control logic is impossible.

Internal fault consequences may be projected to the output of the function block when the information about the block internal structure is limited.

The accuracy of the fault-tolerance analysis in the proposed method depends on how well the high-level fault models correspond to the real system behavior. For small satellites at typical orbits (lower 750 km) the fault models for the next radiation effects are valid based on the published empirical observations [61, 66, 90, 2] and the conducted radiation effects analysis (Chapter 3):

- SEU - or bit-flip in a memory cell (CPU registers, memory arrays, block outputs, etc.)
- MCU - or multiple SEUs in adjacent memory cells (CPU registers, memory arrays, block outputs, etc.)

- SEFI- the functional block/electronic component is put into an unknown state or frozen in reality. SEFI corresponds to the radiation-induced fault in the control circuit of a component. SEFI is simulated by freezing of a functional block or randomization of its state

SETs are also expected in COTS components. However, unless SET is latched in a memory cell, it is not causing any changes in the OBC's operations. If SET is latched, it changes the memory cell content. The memory content change corresponds also to SEU/MCU fault models and SEFI fault models (if the changed memory cell belongs to control logic). Consequently, additional SET fault models are not required in high-level modeling of digital components.

Hereafter, fault models for different components are specified separately.

5.1.1 CPU fault models

SEU fault model is implemented as the content change in a memory cell of the CPU registers (see Figure 5.1). MCU fault models are represented by the models of SEUs in adjacent bits beginning from the random bit at a random register (or the fault location is user defined) (see Figure 5.2). The length of all registers in ARM Cortex-M3 core is 32 bits. In comparison with SRAM/DRAM memory, it is difficult to say how one MCU can flip the bits of several registers because their physical locations are unknown. Consequently, if MCU starts at 32nd bit (the last one), it can be considered as SEU. Such approach is not correct for SRAM and DRAM memories: if MCU happens in the last bit of a byte, the rest bit-flips of MCU will happen in the next byte (if the byte, where MCU starts, is not the last one in the memory storage).

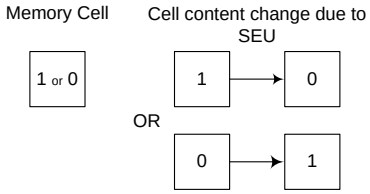

Figure 5.1: SEU model representation

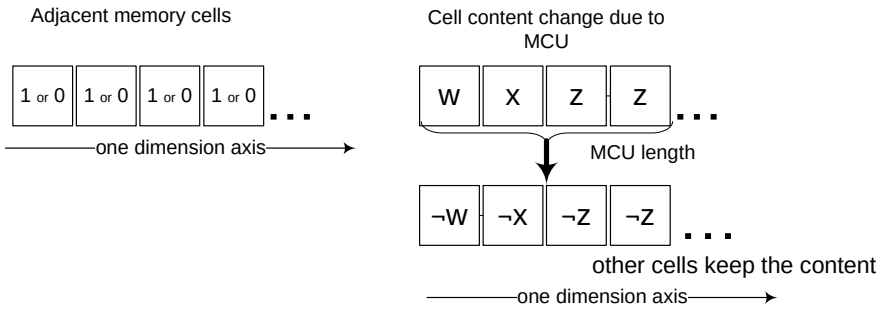

Figure 5.2: One-dimensional MCU model representation

The SEFI fault model corresponds to the simulation behavior when CPU functional block stops instruction execution (freezes) [35]. Power cycling and CPU reset are simulated when the reset signal is raised at the CPU block port. After the reset signal, the CPU starts execution from zero address.

Although radiation faults inside CPU registers are very low probable due to the small size of the CPU area, the CPU fault models can also represent the SEEs in control and bus logic due to the fault propagation. Such SEEs are much more probable than SEUs in registers by direct particle influence.

5.1.2 SRAM and DRAM fault models

SEU fault model and MCU fault model are implemented in the same way as it is done for CPU registers. Additionally, the fault model for spatial (two-dimensional) MCUs is added (see Figure 5.3).

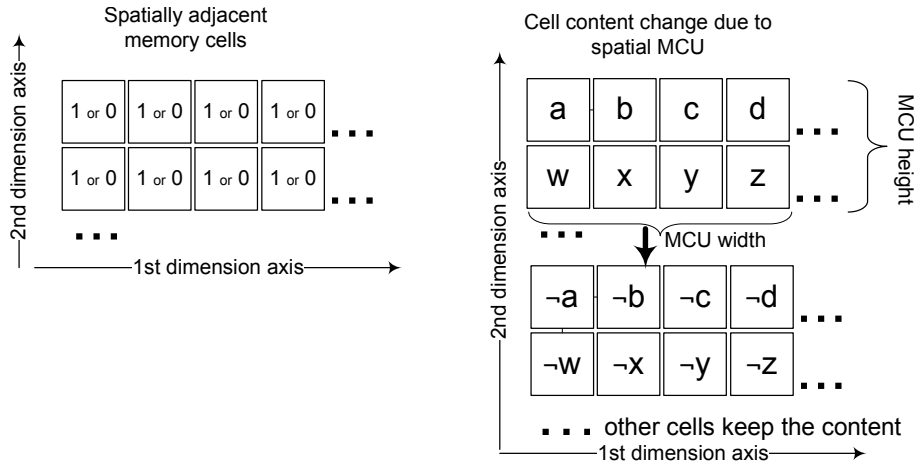

Figure 5.3: Spatial MCU model representation

The fault model for a spatial MCU requires the information about the physical layout of the used memory storage, in particular, the number of rows and columns. Other two extra parameters are required to specify the shape of the MCU: MCU width and height. Consequently, the spatial MCU may corrupt data not only on the same row but also in the same column.

SEFI in memory can be modeled by the memory functional block that stopped responding[35].

5.1.3 Flash-based memory fault models

As it was identified in Section 3.1.3.3, SEFI occurs during read/write process and the whole memory block can be corrupted in Flash-based memory storages. Consequently, the fault injection in this case is not time dependent but dependent on the read/write cycles . The period of Flash SEFI occurrence with a random component (to introduce randomness) is predefined. The dedicated counter of the Analyzer object is incremented each read/write cycle and the decision about the simulated fault introduction is made.

The alternative way is the fault introduction according to the fault list/schedule defined by a user.

5.1.4 Flash-based FPGA fault models

As it was identified in Section 3.1.3.4, the corruption of the user memory (flip-flops and embedded SRAM) are possible by SEE in Flash-based FPGAs. However, the used memory resources for FPGA co-processors cannot be defined before the co-processor is mapped. Consequently, the fault models can be injected only in the modeled communication channels of the FPGA functional block (particularly corrupting the content of TLM transaction objects).

As a result, faults inside the user memory of Flash-based FPGA fabric can be modeled by utilizing the Fabric Controller Model as a usual FIM.

5.1.5 Failure modes

The OBC fault-tolerance analysis is based on the FMEA method [108, 110]. The potential failure modes of particular OBC configuration can be found from the fault injection procedure during each simulation iteration. All system failure modes can be classified as:

1. Incorrect data/Correct time (ID/CT): a benchmark execution is finished normally on time (takes $\leq$ the allocated simulated time), but the final execution results are wrong.

2. Correct data/Incorrect time (CD/IT): execution results are correct, but the process takes longer time than originally allocated.

3. Incorrect data/Incorrect time (ID/IT): execution results are not correct, the simulated execution is not finished as expected by the program flow and was interrupted externally.

4. Processor Deadlock (PD)

5. Read Align Exception (RAE)

6. Write Align Exception (WAE)

7. Arithmetic Exception (AE)

8. Read, Write, or Fetch Privilege Exception (RPE, WPE, or FPE consequently)

PD, RAE, WAE, RPE, WPE, FPE and AE may belong to ID/IT or CD/IT [120, p. 29-30], but can be researched separately to see the overall picture more clearly.

5.2 Conclusions

This chapter discussed two main topics: fault modeling of the possible radiation-induced effects that were identified in Chapter 3.

The fault models for SEU, MCU, and SEFI are built for main COTS components (e.g. CPU, volatile and non-volatile memory storages). However, the limitation of the knowledge about the internal structure of COTS parts (e.g. control logic) introduces inaccuracy to the whole simulation framework. The limitations of the presented framework are mainly discussed in the Chapter 7 with following case studies in Chapter 8.

6 Simulation steps

This Chapter introduces the statistical fault-tolerance analysis(Section 6.1) and its generalization, Multidimensional analysis of memory fault consequences (Section 6.2). The simulation flow has already been discussed in the context of the supporting modules' functionality (Section 4.5). This Chapter explains the sequences of simulation steps that have been used to obtain the simulation results presented in Chapter 8.

6.1 Statistical fault-tolerance analysis

The simulation approach is based on the assumption that only one fault can happen during the algorithm execution in the OBC. The expected fault rates are neglectable in comparison with CPU execution speed for small satellites at typical orbits less than 750 km. The maximum estimated fault rate is not more than one fault in 15 mins (512 kbytes SRAM block) for modern COTS components. Consequently, the one fault injection per algorithm execution is a justified approach.

Since only one fault for algorithm execution (corresponds to the simulation run) is possible, then the iterative approach is used: the simulation with one random fault introduction is repeated to see the influence of the fault injection on the simulated computational output.

The usual simulation sequence of one iteration is represented in Figure 6.1.

Afterwards, the fault mitigation techniques can be applied and the iterative simulation should be repeated. The simulation output files (see Figure 6.1) can be analysed and the simulation results can be compared for different OBC configurations, e.g. with and without fault-mitigation techniques.

If several fault models are available, then other fault model is chosen and the simulation is repeated. Thus, the trends and the overall picture can be understood.

In some cases, the introduction of more than one fault is required to understand where the threshold of the algorithm tolerance/robustness is located. Such an experiment can provide satellite designers with additional data for the fault-tolerance analysis.

The iterative nature of the statistical simulation can be realized with shell scripting as it is done in this work. Shell scripting provides a full external control of the simulation process and supports several simultaneous simulations, the output files manipulation and processing.

6.2 Multidimensional analysis of memory fault consequences

The proposed multidimensional analysis is the generalization of the statistical analysis discussed in the previous Section 6.1.

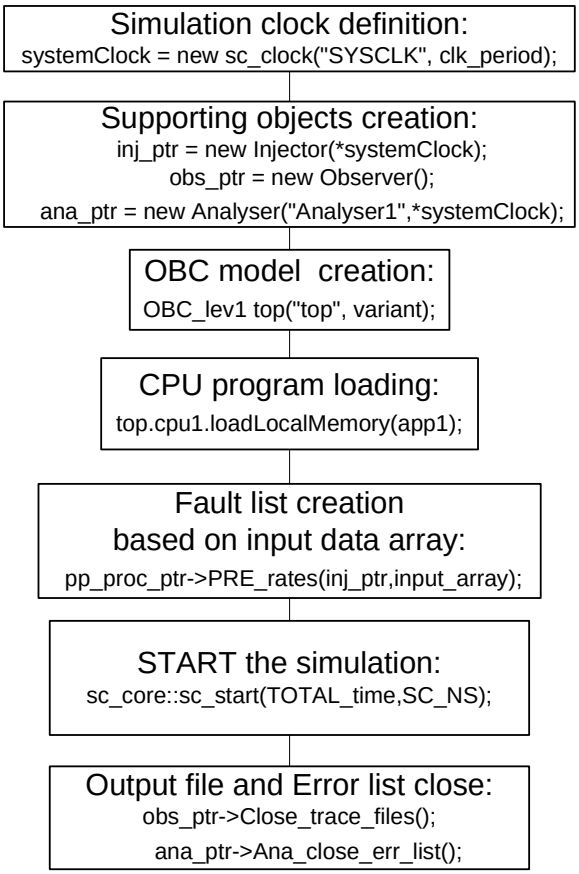

Figure 6.1: Steps of one simulation iteration

Building the OBC model, it is possible to identify used memory locations (e.g. memory ranges, list of registers). Collecting the corresponding memory locations, the memory address axis can be formed(see Figure 6.2). The second axis corresponds to the time of a memory fault introduction. Other dimensions (the third and higher) correspond to the computational quality measurements (introduced by OBC designers) that represent the correctness of the computation output. For different programs the quality measurements can be different: a total execution time, an output value deviation from the correct result, the final computation fault mode [108, 110], etc. The introduction of one or several quality measurements allow the comparison between fault-tolerance of different OBC implementations.

The whole OBC software can be divided into sub-programs that are executed sequentially by a CPU. If there is a program flow dependency between the execution of the previous sub-program f_1 and the next sub-program f_2 (e.g. the shared data in the stack, see Figure 6.2), the influence of a fault, that happened during the execution of f_1, on the output of f_2 sub-program can be investigated by the fault introduction to the commonly used memory locations just before f_2 starts execution. Thus, the fault injection into the memory ranges used by different sub-programs can be limited to the injection when the sub-program (f_0, f_1, or f_2) is being executed.

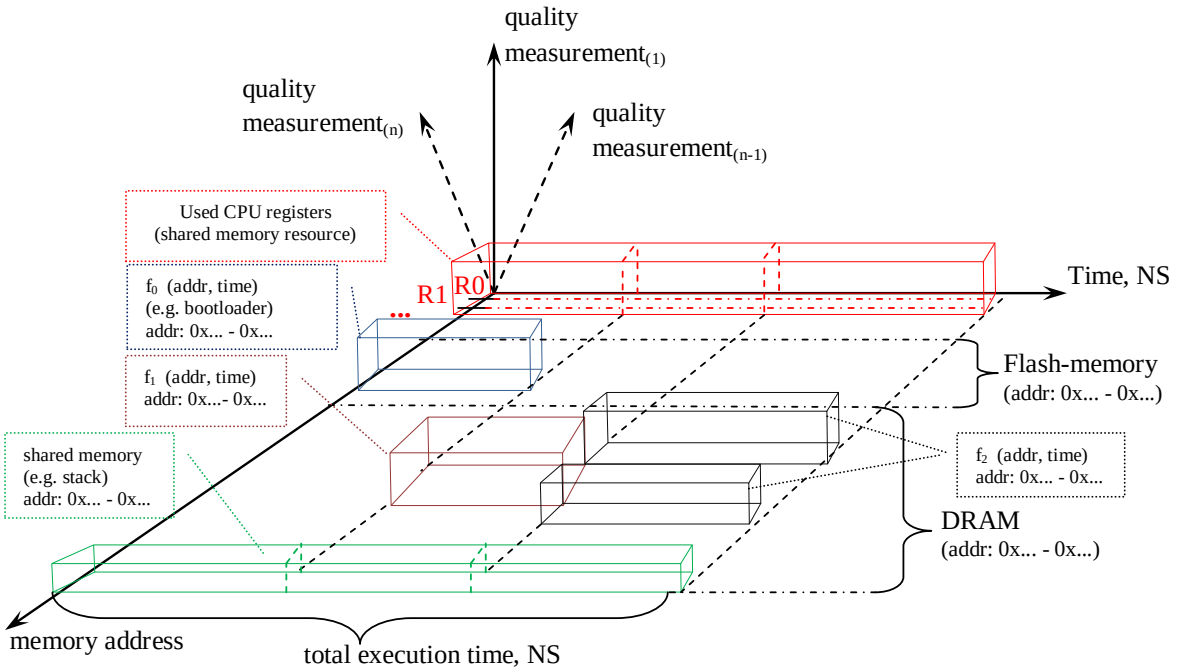

Figure 6.2: The representation of the multidimensional analysis

Introducing the fault models in the different memory addresses at different time iteratively, it will be possible to see the picture how the introduced faults influence the program output. Additionally, after the implementation of the fault-mitigation techniques and the repetition of the experiment, the real effect of the used techniques can be seen. The whole picture will assist satellite designers to optimize the fault-tolerance techniques and pay attention to the areas which corruption significantly damage the output results.

6.3 Conclusions

This Chapter gives an overview of the proposed iterative simulation approach that allows the statistical and multidimensional fault tolerance analysis. The presented simulation steps have been followed to obtain the results presented in Chapter 8.

Model Verification,Validation, and Limitations

7

Model Verification and Validation are discussed in this Chapter since they are important steps of the model development process. A model hardly ever starts to be used in decision making without its verification and validation. This Chapter also discusses the limitations of the presented framework.

7.1 Model Verification

Model verification ensures that the model specification is complete and that mistakes have not been made in the model implementation. Verification does not ensure that the model corresponds to the real world - Validation is responsible for such checking.

The model verification can be done by extensive testing trying to find possible errors, bugs, or algorithmic mistakes. The verification of the model also can be done by peers' review.

In this work the model verification is performed each time after a new model component is introduced (e.g. functional blocks like memory storages or CPU, supporting objects). Testing have been applied to each model component to check the correctness of its functionality. For instance, the fault injection mechanism was tested according to the next operation sequence:

- a memory location reading
- a fault injection to the memory location
- the memory location reading again
- comparison and checking if the fault was really inserted

Additionally, the peers review happened after the aforementioned functionality testing.

7.2 Model Validation

The model validation is answering on the questions if the model represents/corresponds to the reality and what the credibility of the model is.

The first obstacle for the model validation is the absence of the controlled experiment that can prove that the created model (the system model plus the fault model) corresponds to the real radiation effect in the system. Such controlled experiment does not exist since the radiation testing does not provide controlled fault injection capabilities.

The high-level system modeling operates with the abstract representations of the OBC and radiation effects. Consequently, the simulation results' accuracy and credibility are limited by the accuracy of the OBC and fault modeling. Thus, the limited knowledge about COTS components and the whole range of possible radiation effects decreases the model accuracy. On the other hand, the accuracy cannot be increased even with the knowledge about the components' IC-level because the detailed modeling slows down the simulation procedure and makes the extensive statistical analysis impossible.

Since the OBC and fault models are used in conjunction for the simulation, the inaccuracy of one component (e.g. a fault model) can be partially compensated by the accuracy of other one (e.g. the OBC model). For example, the limited knowledge about the control circuit can be replaced by the comprehensive fault model library that accurately describes the possible component-level behavior scenarios.

The additional validation of SEU and MCU fault models is not required since these fault models are generally acknowledged. It benefits the proposed multidimensional fault-tolerance analysis(see Section 6.2). The discussed SEFI fault-models are based only on the published empirical observations and cannot be strictly validated.

In summary,the created SEFI models cannot be fully validated. However, the models and the presented methodology can be used to maximize the OBC fault-tolerance within the available knowledge. SEU and MCU modeling are carried out on the functional level which limits the validation to the correctness checking of the injection procedure. Thus, the proposed simulation-based fault-tolerance approach has the limitation that are discussed in the next Section 7.3

7.3 Model Limitations

The absence of knowledge about the structure of control logic makes SEFI modeling almost impossible. Some observations about SEFI consequences (CPU freezing, not responding memory) can be found in the published literature (as it was presented in Chapter 3). However, all published materials can be considered as one-time observation of the radiation effect that might not happen again. The diversity of SEFI consequences in the control logic is directly connected with the size of FSM that represents this control logic.

Another limitation is the inaccuracy in performance analysis for models of complex systems. The high-level modeling significantly simplifies the internal and communication processes. E.g. OVP instruction-accurate model of ARM Cortex-M3 core is utilized in this with work, so the time accuracy of the model cannot be more than the time of one instruction execution. However, some components may work with much higher frequency than the instruction-execution time. Consequently, the additional analyses is required for the interaction of subsystems with different speeds.

Inaccuracy of the model again can be solved by making each model component more accurate and detailed, which slows down the simulation and prevents the statistical analysis. As a result, the well-known trade-off between fast inaccurate models and slow accurate ones exist.

8 Case studies

This Section presents several case studies that show how the created OBC model can be used for hardware-software co-design and the OBC fault-tolerance analysis.

8.1 Case Study: Recursive algorithm

The calculation of a Fibonacci sequence is chosen as a benchmark algorithm because it represents the family of recursive algorithms (Recursive Fast Fourier Transform Algorithm is another representative of this family[121]). The execution lasts till the 15^{th} Fibonacci element. Each element is calculated in recursive way which guarantees that the calculation of the n and $n+1$ element are data independent and the later calculation takes longer than the former one. Each simulation is repeated for 300 times to obtain a statistical overview of possible influence of SEU induced in the CPU registers. The simulation shows the dependence between the fault modes and fault rates as well as the dependence of the incorrect result ratio on the fault rates.

The Cortex-M3 processor performance is 100 Dhrystone MIPS(Million Instructions Per Second) (DMIPS) for these simulations.

8.1.1 SEU injection into CPU registers

Figure 8.1 shows the dependency of the incorrect calculated values of Fibonacci sequence on the SEU rate in CPU control and general purpose registers. It explicitly presents that the ratio of the incorrect results is growing with the fault rate growth (from right to left- from blue to red) and the execution time duration.

On the other hand, it is essential that the calculation of the first Fibonacci members are less susceptible to the randomly introduced errors. The fact, that the code to calculate these first members is executed first and faster, in comparison with calculation of other Fibonacci member with higher indexes, reduces the probability that the result for the first members will be incorrect.

Picture 8.2 shows how the ratio of different failure modes (see Section 5.1.5) depends on the SEU rate. One of the important observation is the visible growth of Processor Deadlock (PD) failure mode with higher fault rates.

Meanwhile the percentage of Incorrect Data/Correct Time(ID/CT) (or Silent Data Corruption[110]) stays approximately the same. This observation means that for this particular program the ratio of faults that do not cause the CPU freezing or CPU exceptions, but do corrupt the final results, stays approximately the same. This simulation results can be explained by the small number of GPR registers used for the algorithm execution and the increasing probability that the CPU will freeze at higher fault rates, which increases PD ratio and masks ID/CT faults.

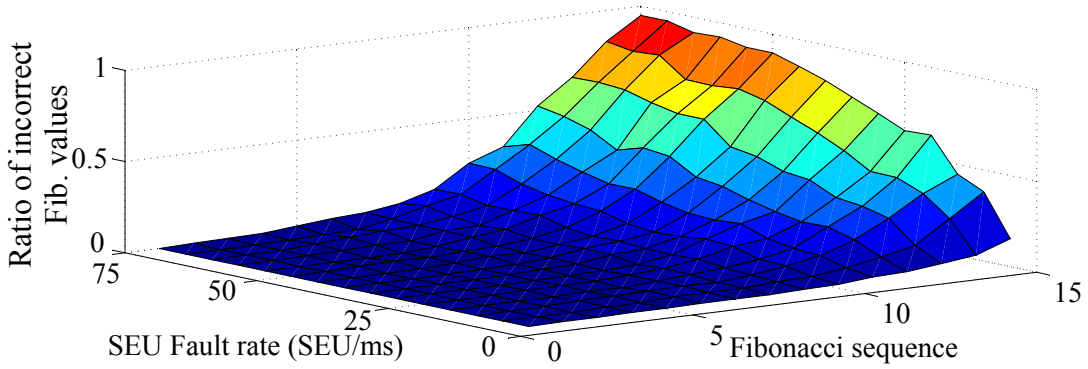

Figure 8.1: Influence of SEU rate in CPU on incorrect result ratio

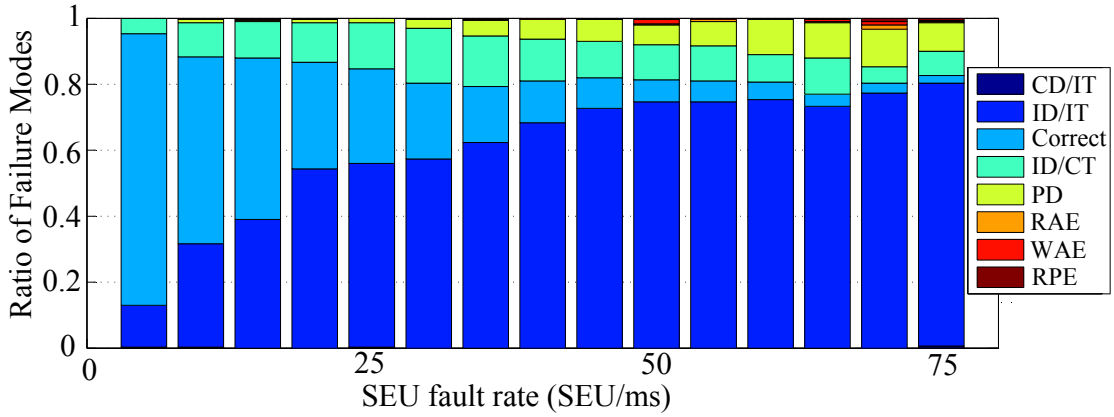

Figure 8.2: Influence of SEU rate in CPU on Failure Mode ratio

Additionally, it is necessary to say that if CPU deadlock appears during the calculation of n^{th} Fibonacci element, the element *n+1* during this simulation iteration will not be calculated at all and the result for *n,(n+1),...,* n^{max} will be wrong. Consequently, the exponential growth of incorrect results is expected.

8.1.2 SEFI injection into CPU

In the next experiment, the same Fibonacci sequence calculation is executed but the CPU resets the watchdog after each member computation. SEFI (or CPU hanging) are randomly introduced into the CPU during execution. Consequently, if the CPU freezes(SEFI happens), the watchdog resets the CPU which restarts the program.

The simulated watchdog provides this functionality. It is counting down at the frequency of 10 MHz. When the watchdog has counted down to zero, it resets the CPU. The period of the watchdog is set to 409.5 us, which corresponds to *0xfff* in the configuration register of the watchdog.

Since the CPU is unavailable for some time after SEFI introduction, the correct results cannot be calculated within the same allocated time as it happens without

SEFI. Consequently, a three time longer period is allocated for the execution.

Figure 8.3 shows the dependency between the fault rate, execution time, and the ratio of incorrect results.

Figure 8.4 presents the case (marked with "1") where the watchdog has a longer timeout period than the calculation requires (timeout period equals 409.5 us).The calculation of the first Fibonacci elements is almost always correct: with the given SEFI rate it is always possible to execute some instructions (including first element calculation) till the next SEFI happen.

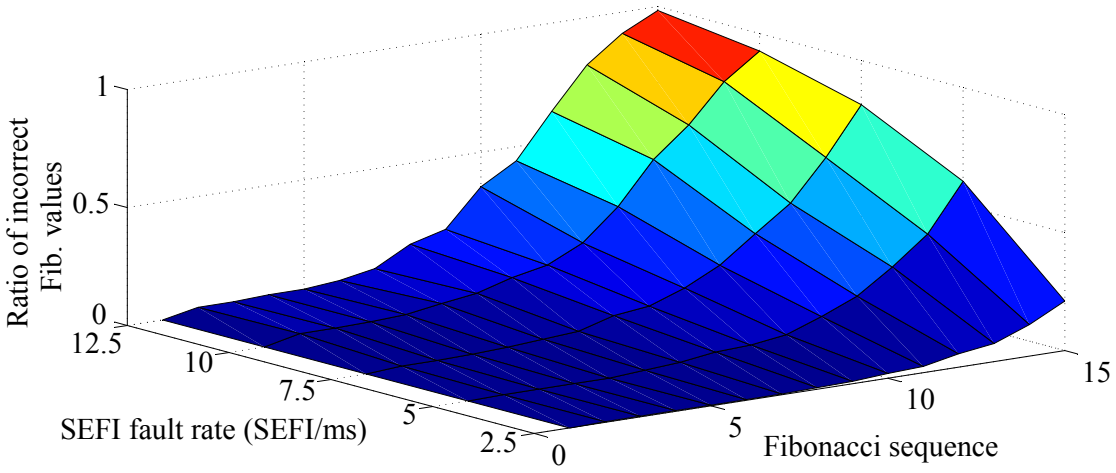

Figure 8.3: Influence of SEFI rate on incorrect result ratio - untuned watchdog

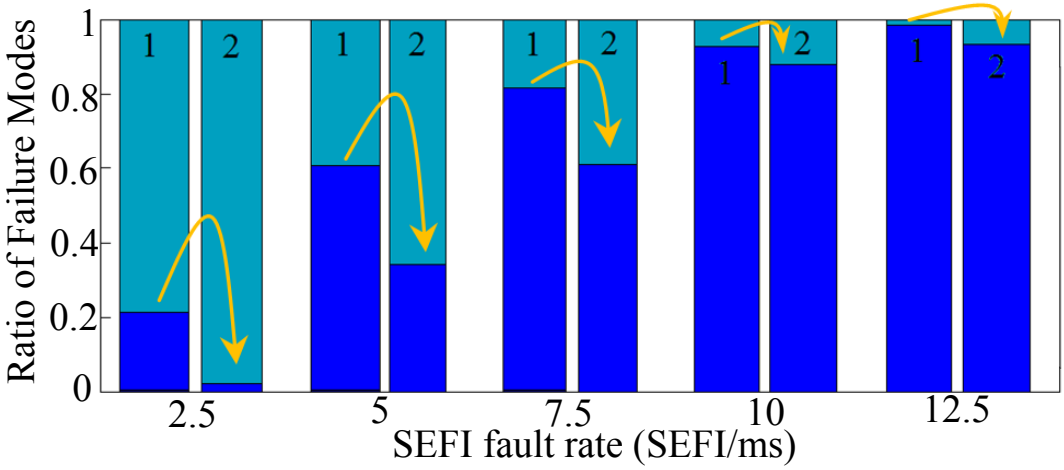

Figure 8.4: Influence of SEFI rate on Failure Modes with the unturned (marked with "1") and tuned watchdog (marked with "2")

As a result, CPU waits for an extended period of time as opposed to the second case, marked with "2" (see Figure 8.4). In the 2nd case, the watchdog timeout period has been reduced by approximately four times, to 102.3 s (by setting configuration register

to *0x3ff*). This watchdog period is more tuned to the algorithm; hence in the case of SEFI the CPU is halted for a shorter period of time before the watchdog times out and resets the CPU. Consequently, the allocated time becomes sufficient to execute the whole algorithm at low fault rates, which is indicated by the pulled-down right corner of the surface (see Figure 8.5).

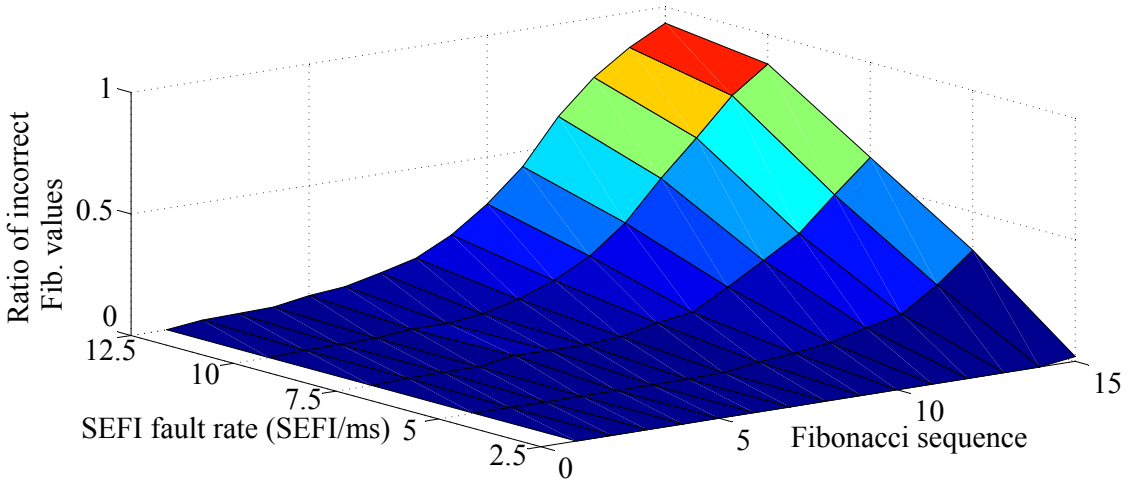

Figure 8.5: Influence of SEFI rate on incorrect result ratio - tuned watched

The results of the simulations with SEFI injection shows that the watchdog timeout tuning to the algorithm is indispensable for OBC fault-tolerance.

8.1.3 SEU injection into SRAM memory

Instruction and data code are located in SRAM for this simulation(in general, it is not necessary, and the Instruction code can be executed from embedded Flash-memory in SmartFusion device). Consequently, introducing SEU to SRAM we can corrupt both data types. Since Fibonacci benchmark program does not operate with a lot of data, only array of Fibonacci sequence, the ratio of Incorrect Data/Correct Time(ID/CT) failure mode is relatively small and prevails only until SEUs start to cause Processor Deadlock (PD) (12.5 SEU/bit/sec, see Figure 8.7). The grows of incorrect results at higher fault rates (left surface corner, Figure 8.6) is caused by instruction corruption at the beginning, which causes PD since CPU cannot decode the instruction.

8.2 Case Study: JPEG image compression

JPEG image compression standard introduced by Joint Photographic Experts Group (JPEG) is one of the most wide-spread digital image processing algorithms with lossy compression. The flexibility of the algorithm is provided by the adjustable compression degree, which gives an opportunity to reduce the picture size without visible quality loss (up to 10 times compression).

As it was mentioned in Section 2.2.1, the radio downlink of Micro and Nano satellites is limited. Thus, the JPEG compression plays an important role in the on-board data

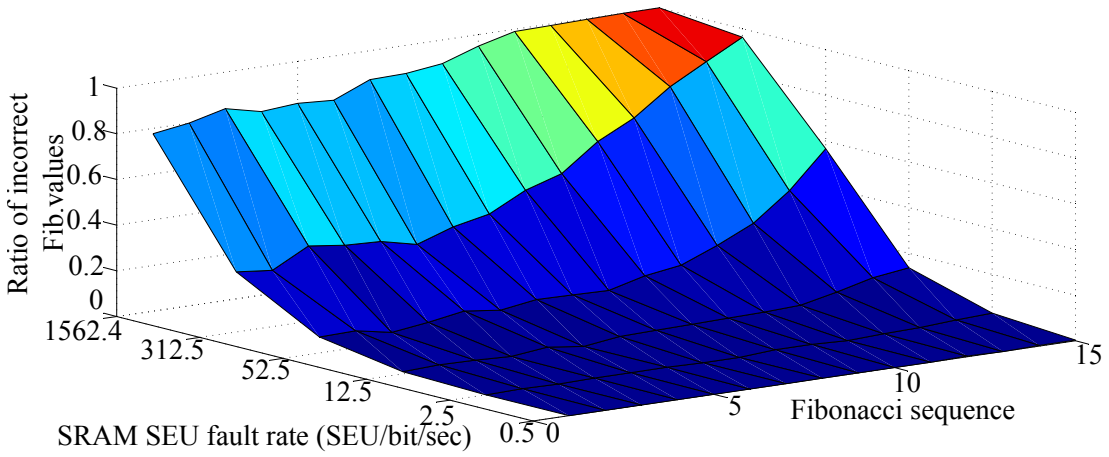

Figure 8.6: Influence of SEU rate in SRAM on Failure Mode, without mitigation techniques

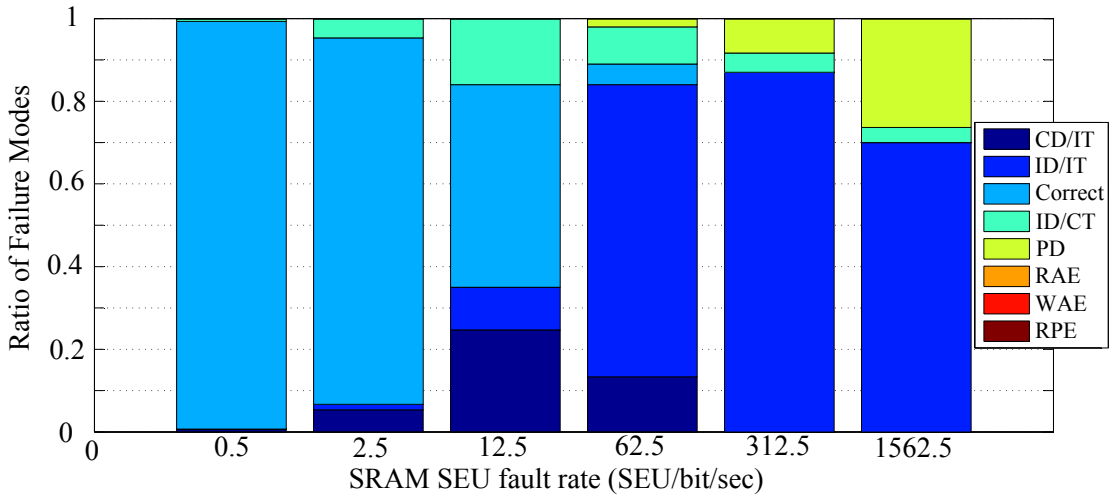

Figure 8.7: Influence of SEU rate in SRAM on Failure Mode, without mitigation techniques

processing. This Section presents the case study with more than 30 times compression and slightly visible image quality degradation to emphasize the high JPEG capabilities.

For the case study, the source picture (see Figure 8.8) was chosen with parameters presented in Table 8.1

Initial Extension	Bitmap Picture(BMP)
Initial size	373 302 bytes
Width	432 pixels
Height	288 pixels

Table 8.1: The source image parameters

JPEG compression algorithm is well-known and its source code can be easily found

Figure 8.8: The source image for JPEG compression procedure

on the Internet. For this particular implementation, JPEG algorithm structures have been taken from Embedded JPEG Codec Library[122] of Joris van Emden. However, while the core algorithmic part is unchangeable, the code has to be cross-compiled for the chosen ARM-based platform. Consequently, some changes were introduced to the original code:

- The input and output procedures have been changed in a way to save the original picture to the external DRAM (address *0x60100100*) and to write the output pictures just after the original one in the memory map - address *0x6015B336*.
- The algorithm has been partially changed mainly because of the different interpretations of *char* data types by ARM processor cross-compiler and standard *gcc* compiler (char = unsigned char in *gcc* and char = signed char in ARM compiler).

To verify the correctness of the final version of JPEG library for the chosen ARM-based platform, the output picture has been compared byte-by-byte with the result of original code execution on PC with *gcc* compiler utilization.

The compressed picture is presented in Figure 8.9 and its main parameters are observed in Table 8.2.

Final Extension	JPEG
Final size	10 647 bytes
Width	432 pixels
Height	288 pixels

Table 8.2: The output image parameters

As a result, the pictures was compressed about 35 times with visible picture quality degradation.

Figure 8.9: The output image after JPEG compression procedure

The essential difference of this algorithm and the Fibonacci sequence calculation is the image processing. The image is located in SEU-sensitive Dynamic RAM (DRAM) memory. Thus, the next simulation is aimed to investigate the influence of the introduced bit-flips in DRAM memory on the final picture appearance.

8.3 Memory scrubbing technique implementation

Fault-mitigation techniques can be implemented in the Flash-based FPGA fabric of SmartFusion device to release the General-Purpose Processor (GPP) Cortex-M3. The simplest version of memory scrubbing is based on several copies of the original information and continuous checking if copies are the same.

As a case study, the discussed JPEG compression algorithm is used and SEU are injected in the memory where the original picture (not compressed picture) is located. The purpose of such experiment is to identify the influence of SEU on the final JPEG compression output. It is assumed that the faults are equally distributed in time and in space. SEU are not injected in the instruction data in this case.

Figure 8.10 presents the source picture after SEU injection. Some of the faulty pixels are marked with the orange outline.

It is difficult to visually assess the differences between the damaged source picture and the original image. So a byte-by-byte comparison has been conducted which has shown exactly 40, 400, and 2000 corrupted bytes (since each contains a faulty bit). If we build the distribution of the the absolute difference between bytes that are not equal each other (see Figure 8.11), it will be discrete at particular values: 1, 2, 4, 8,..., 128.

The output pictures are presented in Figure 8.12 and the corresponding bytes' difference distribution in Figure 8.13. Taking into account the fact that the whole size of the output picture is 10647 bytes, it is possible to state that even with 40 corrupted bytes in the source image the output picture differs significantly. The statistical difference can

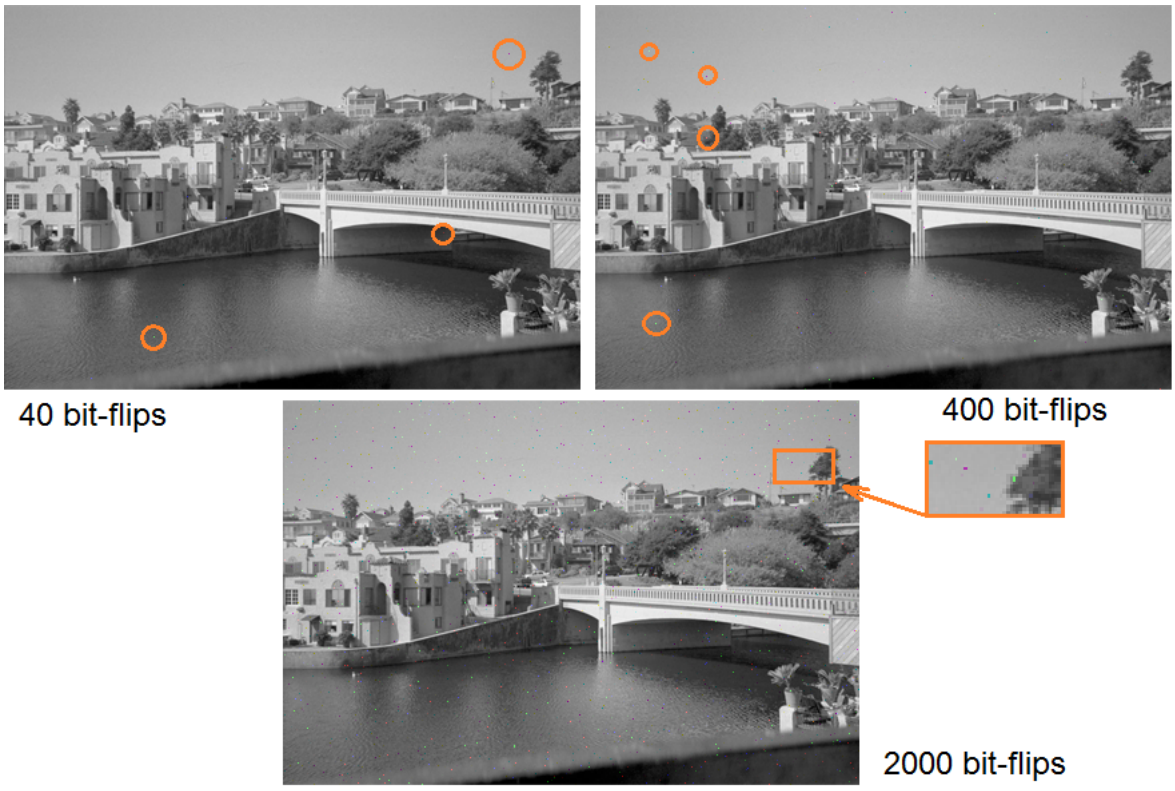

Figure 8.10: Example of the source image SEU corruption - w/o mitigation techniques

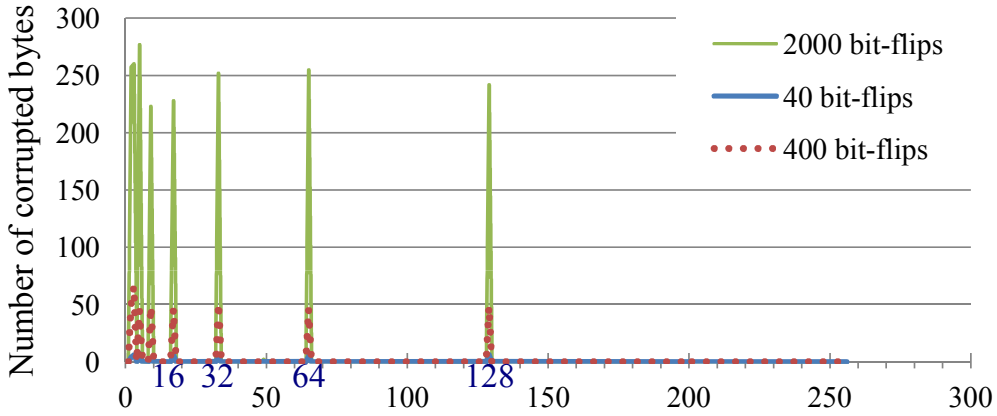

Figure 8.11: Difference distribution of unequal bytes in the source images w/o mitigation techniques

be explained by the fact that JPEG compression algorithm averages the picture, and faulty source pixels slightly change the average value. The difference distributions(see Figure 8.13) show that the majority of the pixels differ on small absolute values, which is not visually noticeable.

To prevent corruption that is visible in Figure 8.12, the FPGA scrubbing technique is implemented. After the CPU starts the JPEG compression execution, the scrubbing unit, implemented in the FPGA, begins to copy the source picture to two backup

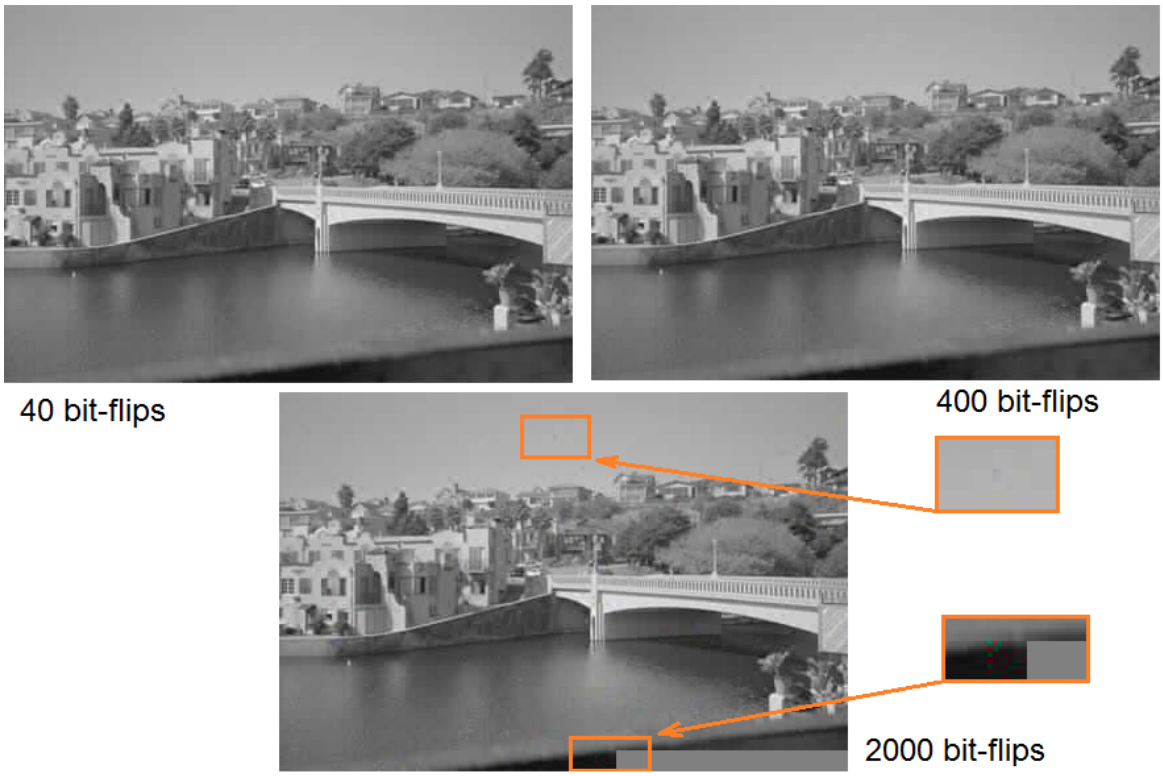

Figure 8.12: JPEG compression output under the source image corruption by SEUs w/o mitigation techniques

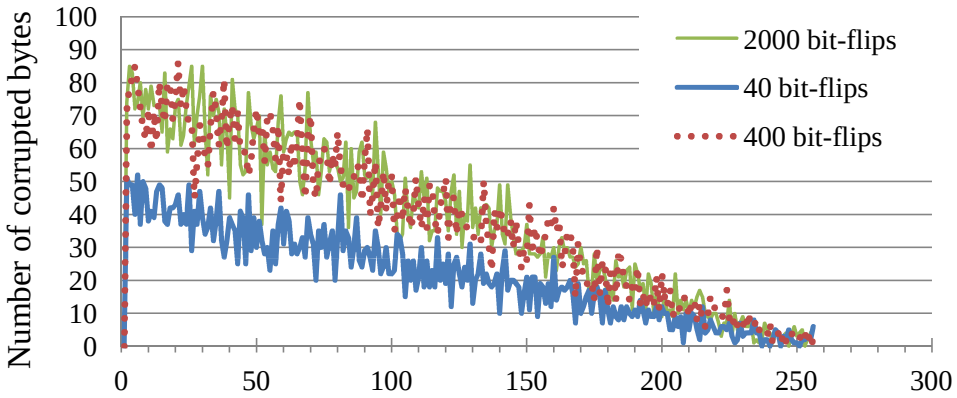

Figure 8.13: Difference distribution of unequal bytes in the compressed image w/o mitigation techniques

versions. Since the SEU occurrence is randomly distributed in time, the copied pictures may have small or zero number of corrupted pixels. After copying, the FPGA scrubber continuously compares the source picture, with which the CPU is working, to the first backup copy. If the scrubber finds a difference, it compares it to the corresponding byte in the second backup copy and makes a decision about the most probable correct

value of this byte.

As a result, the number of corrupted bytes in the source image has been reduced by 80-87% (see Figure 8.14) and there is no visual corruption in the output picture(see Figure 8.15). Byte-by-byte comparison of the output picture with the faultless compressed version reveals the difference in a majority (more than 92%) of the files bytes. However, this difference cannot be seen since the JPEG compression is making a local average of the pixel color values. This way SEUs are filtered out.

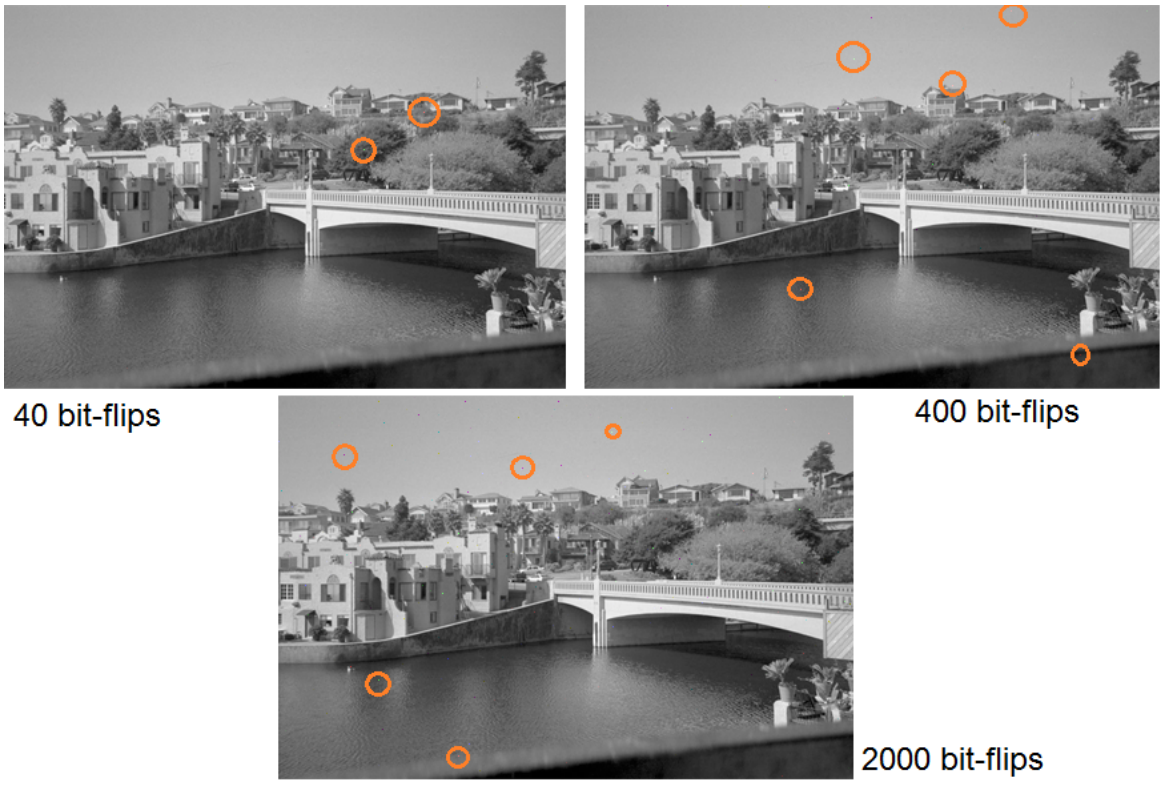

Figure 8.14: The source image SEU corruption - with Memory Scrubbing

When MCUs are introduced, the corruption becomes visible in the compressed picture (see Figure 8.16, no scrubbing implemented). But the probability of such long bit-flips is significantly lower.

The aforementioned memory scrubbing utilizes the TMR of the memory, which may be not supported by the limited OBC memory resources. Hence, more memory efficient approach should be used, e.g. memory scrubbing with Hamming encoding [123, 124]. While Hamming encoding requires memory only to save the syndromes, its implementation requires more FPGA logic due to a complicated control FSM. The encoding phase also requires an extensive number of multiplexers. RTL SystemC compilation to RTL has been conducted using CatapultC Synthesis[125]; the results are presented in Table 8.3 (frequency 50 MHz).

The co-processor with Hamming encoding is discussed in detail in Section 8.5.

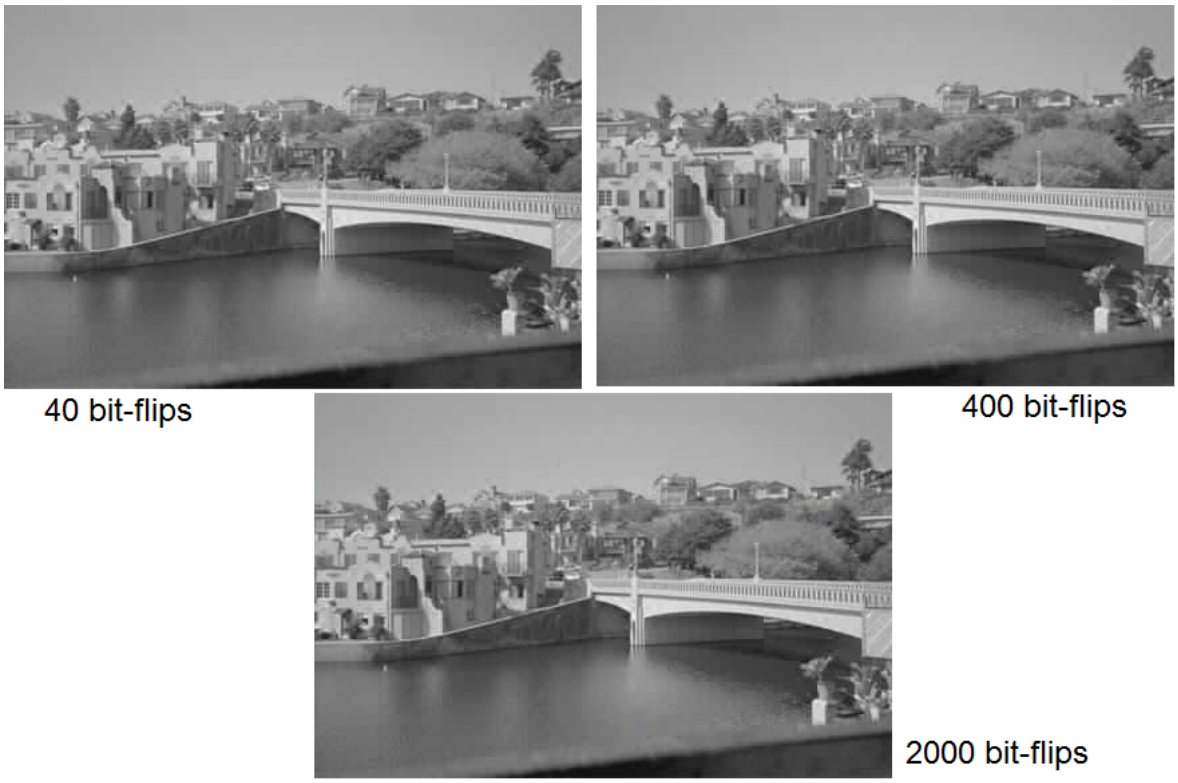

Figure 8.15: JPEG compression output - with Memory Scrubbing

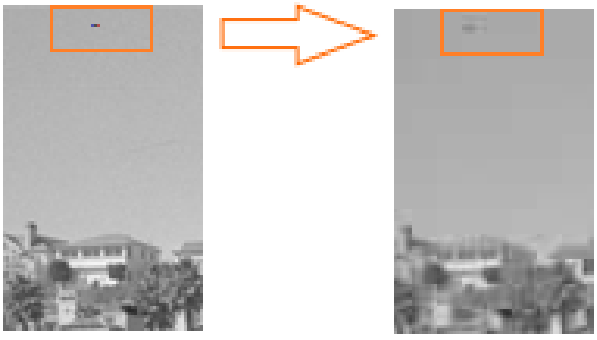

Figure 8.16: JPEG compression with introduced 56-bit long MCU; the source and the output

8.4 Case Study: Kalman filter of Attitude Determination and Control Algorithm

A satellites ADCS provides knowledge and control over its orientation in space. In the case of possible erroneous operation the orientation can be disturbed, which can have a detrimental effect on the satellite mission. To understand what kind of effects SRAM SEUs have on this type of algorithm, 1000 SEUs have been randomly introduced to the memory where the algorithms variables are stored (sections *.bss* and *.data*). The

Scrubbing type	Latency Cycles	Slack	Total Area
TMR	1500	12.25	29723.13
Hamming	177081	7.55	181677.10

Table 8.3: The output of CatapultC Synthesis for TMR and Hamming-based memory scrubbing FPGA co-processors

injected errors are uniformly distributed in time and physical area. The correct output of Kalman filter, which is the part of the ADCS, is presented in Figure 8.17 (three axes). The Kalman output with SEUs injection is presented in Figure 8.18.

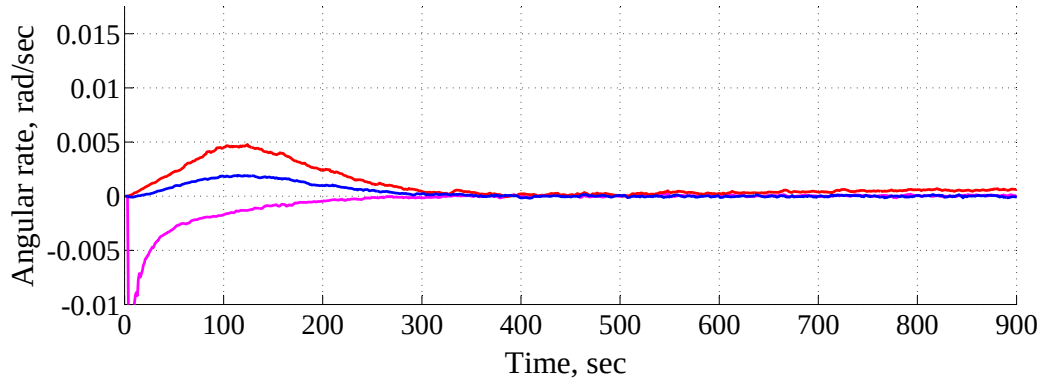

Figure 8.17: Correct Kalman filter output

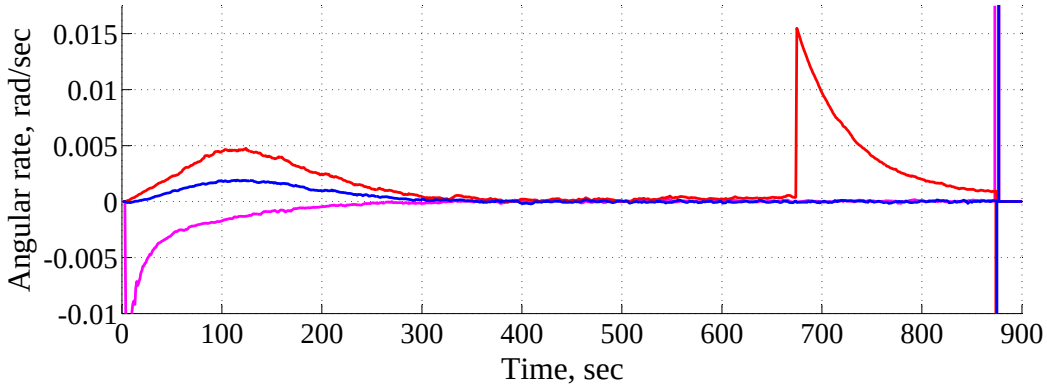

Figure 8.18: Kalman filter output with SEUs introduction

Comparing Figure 8.17 and Figure 8.18, it is easy to see the effect of the injected errors. The detection of such behavior plays an important role since, if Kalman filter losses stability (as it happened after 875 sec), it may not converge again until the external reset. The software fault detection mechanism has been implemented based on the detection of the output value jump bigger than 0.001 rad/sec. If the output value jump has been found, the current output value is ignored and the Kalman filter resets. As a result of the implemented technique, the output values became very close to the

correct result, Figure 8.17: the average difference between the correct filter output and the output with the described software fault-tolerance technique equals 1.710^{-7} rad/sec for the period 0-900 seconds. This difference is acceptable and can be considered as negligible since it is four orders of magnitude less than the scale of the output values.

8.5 Case Study: Multidimensional analysis of memory fault consequences in Adaptive filter

As an example of the multidimensional fault-tolerance analysis, one SEU is introduced each simulation iteration to the memory region where the program of an ADCS adaptive filter is saved (see Table 8.9).

Name	**Size**	**Virtual Memory Address (VMA)=Load Memory Address (LMA)**
.text	0x000101c	0x0000000
.rodata	0x0000044	0x000101c
.data	0x0000178	0x0001060
.bss	0x0000020	0x00011d8

Table 8.4: The Memory Sections where SEUs are Injected

The single SEU per iteration is again motivated by low possible fault-rates on altitudes less than 750 km in comparison with the short execution time of the algorithm.

8.5.1 Code execution without fault mitigation techniques

The CPU executes the code of an adaptive filter used in a ADCS algorithm for filtering the measurements of solar sensors and magnetometers (Table 8.5). The adaptive filter during one simulation iteration calculates 30 results for 3 axis, 90 double values in total. The chosen quality measurement is the average relative deviation of 90 values calculated with SEU introduction from 90 correctly calculated filter outputs. The result of the simulation is presented in Figure 8.19.

CPU type:	armm (Cortex-M3)
Nominal MIPS:	100
Simulated instructions:	105,515
User time:	0.79 seconds
Elapsed time:	0.83 seconds

Table 8.5: Simulation Time for the Version without Protection - One Iteration

Figure 8.19 shows that during the SEU introduction to particular parts of software (particular memory addresses) the calculated results differ from the correct ones significantly (these cases are marked with colored dots). At the same time, SEU introduction to other regions, e.g. the very beginning of the memory (address 0x0-0x19a), does not influence the calculation output. It can be explained by the interrupt vectors that are located at the beginning of the memory and not directly used during this simulation and during the real computation. The computation in main function starts particularly at address 0x19a where we can observe first error results(represented by colored dots).

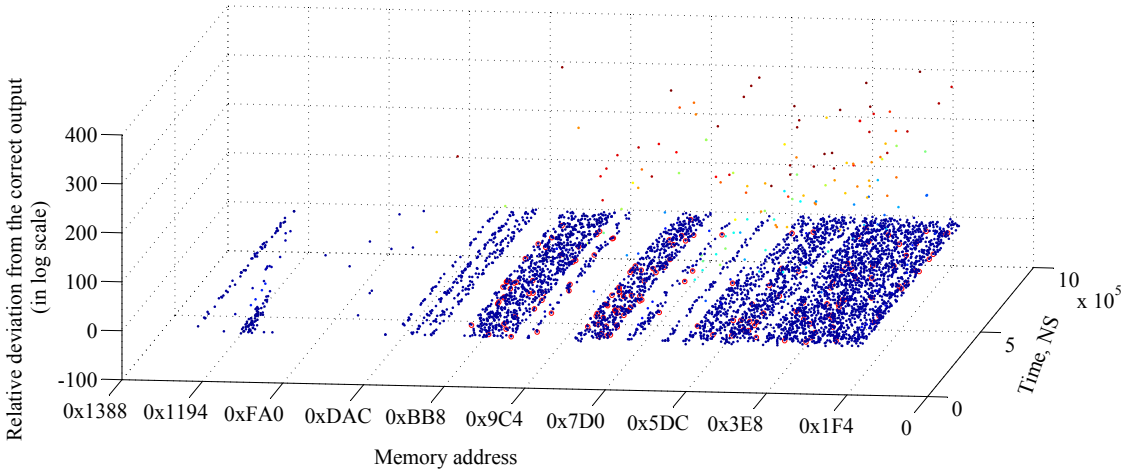

Figure 8.19: The simulation result of the adaptive filtering computation with one SEU introduction (20 000 iterations)

8.5.2 Code execution with fault mitigation

Since SmartFusion SoC includes FPGA fabric, some fault-tolerance techniques can be implemented in FPGA. Memory scrubbing based on Hamming encoding has been chosen as an example (see Table 8.6). Since *.text* and *.rodata* section of the program are static, they can be protected with the memory scrubbing technique.

Nominal MIPS:	100
Simulated instructions:	199,195
User time:	2.36 seconds
Elapsed time:	2.47 seconds

Table 8.6: Simulation Time for the Version with FPGA-based Protection - One Iteration

Before the algorithm execution the CPU sends to the FPGA co-processor the number and locations of memory ranges that should be protected by FPGA scrubbing. FPGA co-processor encodes the data saving the syndromes and one backup copy of the protected memory regions. After the syndromes generation, the FPGA co-processor sends to the CPU a message indicating that the CPU can continue algorithm execution. Since the syndromes are known, FPGA starts scrubbing the protected memory calculating syndromes again and comparing them with the saved ones that are supposed to be correct. However, the syndromes can also be damaged by SEU; so the backup version is used when the syndromes difference is found to clarify if the syndrome or the protected region is damaged.

Another used fault-mitigation technique is a software-based value limitation: the difference between consecutively calculated outputs are limited (see Table8.7). The limitation is based on the expected parameters range and the speed of their changes . In the case study, the limit on the output value change is set to 10^{-4} when the filter output values lie in the scale of 10^{-5} and 10^{-6}. If the limit is not met, the filter

recalculates the values.

Nominal MIPS:	100
Simulated instructions:	123, 442
User time:	1.26 seconds
Elapsed time:	1.28 seconds

Table 8.7: Simulation Time for the Version with Software Protection - One Iteration

The simulation results are shown in Figure 8.20 and Table 8.8.

Relative deviation of the values from the correct ones	Ratio of iterations with such deviation (20 000 iterations in total)		
	without protection	SW limitation	FPGA-based Hamming encoding
[0-10)%	*82.81%*	*84.58%*	*88.60%*
$[10-20)\%$	2.36%	1.76%	1.45%
$[20-30)\%$	0.57%	0.27%	0.21%
$[100-110)\%$	11.31%	11.75%	7.93%

Table 8.8: Simulation Results, System Fault-Tolerance with and without Fault-Mitigation Techniques

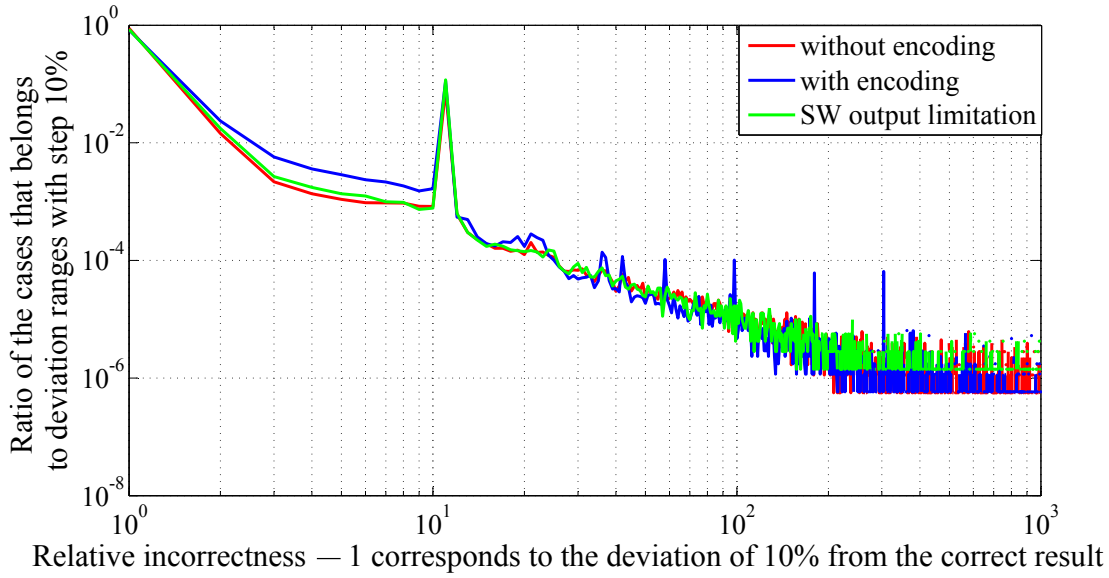

Figure 8.20: Histogram of system fault-tolerance with and without fault-mitigation techniques (20 000 iterations)

The jump when the relative incorrectness equals 11 (see Figure 8.20)can be explained by the next observation. In many cases after the fault injection, the CPU raises the exception. If the exception routine is not written (as in the presented case), the algorithm execution stops and the memory region, where the filter output values

should be written to, stays equal zero. When the output value is zero the relative deviation from the correct result equals one. The peak is located at 11 relative incorrectness since 11 corresponds to [100%, 110%) deviation from the correct result.

Figure 8.20 and Table 8.8 show the expected tendency: the ratio of the iterations with the correct results or results with less than 10% deviation from the correct values is increasing when fault-mitigation techniques are implemented. Additionally, Figure 8.20 shows that the ratio of rare detrimental faults (right bottom corner of the plot) is reduced by the memory scrubbing with Hamming encoding.

8.5.3 System-level behavior

As it was observed from Figure 8.19 , the corruption of some memory regions causes significant degradation in the correctness of the final result; while the corruption of other regions has less detrimental or none effects.

While 20 000 iterations (one SEU per iteration) cannot cover the whole space of possible SEU introduction options (200 000 executed instructions and 36 800 memory bits), some system characteristics converge as it is shown in the next experiment.

The same filtering algorithm was executed 1000, 5000, 10000, 20000, 30000, and 40000 iterations (one SEU per iteration). Figure 8.21 represents the histogram obtained during these simulations.

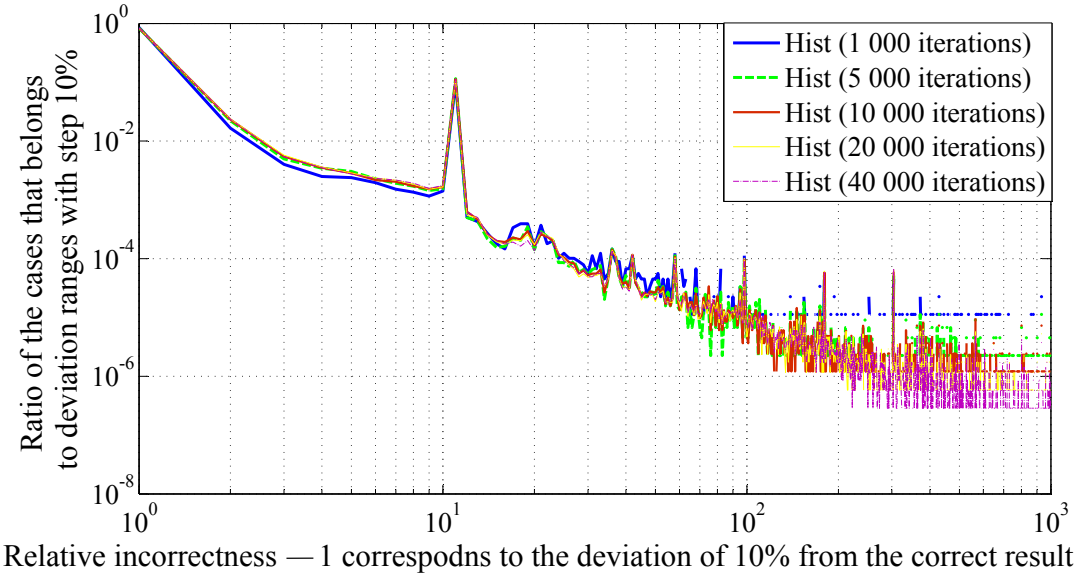

Figure 8.21: The dependency of system behavior on the number of simulation iterations

8.5.4 Clustering algorithm

Since the OBC has strict limitations on the used hardware resources, the optimization of fault-mitigation techniques is required.

According to the observation in Section 8.5.2, the memory regions have different influence on the whole program execution. However, the visual identification of these regions is not always an appropriate way to find the memory region with the biggest influence on the final execution results. The currently existing clustering algorithms are oriented to the division of the whole set into clusters. Meanwhile, in our case it is necessary to find only clusters that meet the requirements on the density and severity of the incorrect results. Consequently, it is highly probably that some memory regions should not be included in clusters since SEU introduction in this region rarely cause the damage of the computation result. Such regions that should not be included into clusters will be considered as outliers by existing clustering algorithms. Usually, such outliers are interpreted as the noise. In our case such regions cannot be considered as something additional, since they are highly expected to appear and should be just ignored from clustering.

The developed clustering algorithm consists of several steps, described hereafter:

1. Euclidian distance calculation with adjustment coefficients.

2. The building local clusters around each of the dots (rule: every other dot is included in the cluster of the current dot if the Euclidian distance for that dot is less than the given threshold).

3. Merging the local clusters if each pair of them have particular number of the common dots (we are starting merging the biggest clusters, the required number of the common dots is calculated as a pre-defined percentage of the all dots of the smaller cluster from the pair).Thus, the biggest clusters have an opportunity to include smaller ones, to form even more bigger clusters.

4. Additional: Overlap clearing. For more clear picture, it is possible to eliminate the overlaps between clusters. We are starting from the biggest ones according to the rule: if the bigger cluster contains the dots of the smaller cluster, then the smaller cluster looses the dots.

The algorithm contains several parameters to be tuned. For example, the distance to other dots that will be included in the local cluster; the percentage of the common dots of the smaller cluster that will be enough to merge two clusters; the coefficient for Euclidian distance calculation. Such parameters can be tuned for example by the genetic algorithm automatically.

Figure 8.19 is taken as an input data. The simulation result is presented in Figure 8.22 (500 simulation iterations/dots).

The presented clustering approach assists a designer to understand the most important memory regions from OBC fault-tolerance point of view and save hardware resources without significant loss in fault-tolerance. For example, it is not practical to scrub 45% of *.text* and *.rodata* regions since they do not have such detrimental effect as other regions (Figure 8.19). Thus, we can implement FPGA-based scrubbing only for other 55% which will reduce the scrubbing turnaround period and the amount of the allocated memory for syndromes and backup copy.

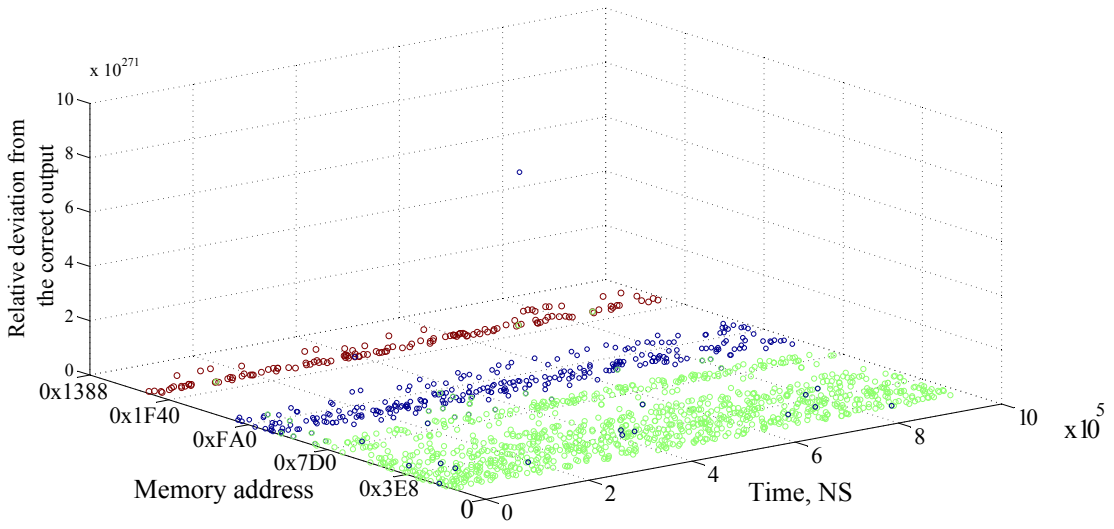

Figure 8.22: Clustering algorithm output

8.5.5 Conclusions

The chosen benchmark program is an adaptive filtering algorithm for the satellite attitude control. The FPGA-based memory protection with Hamming encoding is implemented, assessed, and optimized basing on the simulation results. The comparison between FPGA-based memory scrubbing and a software mitigation technique is presented. The importance of exception handling for satellite OBCs is explained: the proper exception handling may cover up to 11% of wrong computation results.

A shown optimization method based on the proposed clustering algorithm can assist satellite designers in system-level analysis, development, and optimization. Using the proposed method, the limited OBC hardware resources can be allocated with high efficiency. In the presented case study, the method helped to reduce the scrubbing turnaround period and the used memory resources almost by half.

Using the presented modeling and simulation methodology for the OBC fault-tolerance analysis, well-known fault mitigation techniques can be investigated and improved for different applications.

8.6 Simulation time of the case studies

This section provides the information about the simulation time of the presented case studies (see Table 8.9). All simulations have been conducted on a Personal Computer (PC) with an Intel Core i5-2430M and 4 GB of DDR3 memory. As was expected, SystemC TLM modeling provides fast simulation capabilities even for complex cases as the software JPEG image compression with the FPGA-based memory scrubbing.

Case study (one run)	**Number of Instructions**	**User Time (sec)**	**System Time(sec)**	**Elapsed Time(sec)**
Fibonacci sequence calculation(15 elements)	28,912	0.25	0.01	0.26
Fibonacci sequence calculation with watchdog	28,940	0.26	0.01	0.27
JPEG image compression (432x288 pixels)	22,079,255	185.93	0.09	186.89
JPEG image compression with memory scrubbing	22,079,255	225.90	0.10	226.99
Kalman filtering	120,504	1.56	0.04	1.67
Adaptive filtering	105,515	0.79	0.01	0.83
Adaptive filtering with memory scrubbing (Hamming encoding)	199,195	2.36	0.01	2.47
Adaptive filtering with software fault-tolerance technique	123,442	1.26	0.01	1.28

Table 8.9: The Simulation Time of the Case Studies

9 Conclusions and Future Work

9.1 Conclusions

The traditional fault-tolerance analysis of COTS-based satellites is limited due to the hardware unavailability at early development stages and low interpretability of radiation tests. The existing simulation techniques assume that IC-level of used electronic components is known or propose time-consuming complex analytical approaches which are unapplicable in small satellite industry. This work presents an innovative simulation approach for the statistical fault-tolerance analysis that assists satellite designers to develop fault-tolerant OBCs and OBC software in fast and cost-effective manner.

The proposed simulation methodology for the fault-tolerance analysis allows (Chapters 6,8):

- to understand the influence of each software and hardware component on the system fault-tolerance
- to compare the efficiency of fault-tolerance techniques
- to understand the consequences of diverse radiation effects
- to develop OBC software and conduct design exploration at early development stages

The presented methodology includes two main components:

- a SystemC-based OBC model (Chapter 4)
- a C++-based fault injection mechanism with a fault model library (Sections 4.5, 5)

The OBC model has been built based on an heterogeneous SoC, SmartFusion device from Microsemi Corporation (Section 4.3). Such OBC model covers the most general architecture and can be easily adapted for other OBC configurations. The work also shows how to build the stacked OBC model which is a typical representative of system-level redundant computers.

The radiation environment for typical CubeSat missions (with orbits lower 750 km) is estimated (Section 3.1):

- TID for 3-years satellite mission < 3.2 krad
- LET spectra can be considered as equal to zero when the LET is higher than 32.5 $MeV \cdot cm^2/mg$

The corresponding possible radiation effects for the assessed radiation environment include: SEU, MCU, and SEFI. Fault models for these radiation effects have been built (Chapter 5) and used by the object-oriented fault injection mechanism to simulate the consequences of radiation effects in the modeled OBC.

The work covers the next case studies: the fault-tolerance of adaptive and Kalman filters of ADCS, a recursive and compression algorithms. The next results have been obtained:

- the error of the ADCS Kalman filter is minimized to $1.7 \cdot 10^{-7} rad/sec$ using the SIFT technique (Section 8.4) .
- the number of accumulated memory bit-flips is reduced by 80-87% during JPEG image compression using the FPGA-based TMR memory protection technique (Section 8.3).
- the comparison between the FPGA-based memory protection with Hamming encoding and the software fault detection technique is conducted(Section 8.5.2).
- the scrubbing turnaround period and allocated memory resources are reduced almost by 50% with the proposed clustering algorithm (Section 8.5.4).
- the importance of CPU exception handling is highlighted since it may cover up to 11% of wrong computation results(Section 8.5.2).
- the importance of watchdog monitoring is explained (Section 8.1.2).

Three conference papers have been written based on the research results presented in this thesis. The work has been appreciated in Europe and USA where it was presented at ESA 4S Symposium[13] and North-Atlantic Testing Workshop[14] in 2012. The third written paper [15] is being under the review of XXVII Conference on Design of Circuits and Integrated Systems by the moment of this MSc thesis defence.

The proposed approach has been adopted for software development at the company ISIS B.V.

9.2 Future Work

This Section lists several items that can be considered as future work for the presented thesis research:

1. One of the required further steps to increase the usability of the proposed simulation framework is to add Graphical User Interface (GUI).

2. While the simulation time is short enough to conduct the presented statistical analyses, it is necessary to investigate the ways to reduce the total simulation time, e.g by parallelization. Although the object-oriented injection mechanism proposed in this work simplifies the OBC modeling, an additional comparison between the presented approach and well-known FIM is required especially for more complex system models.

3. Other satellite sub-systems can be modeled in the way similar to the OBC modeling. However, the modeling of analog sub-systems or mixed-signal sub-systems requires the utilization of SystemC-AMS library. The possible radiation effects on these sub-systems and the fault modeling can be considered as future work and may lead to SEL modeling.

4. The analysis of the typical radiation conditions presented in this work has shown that permanent effects like SHE will non be observed during a usual small satellite mission. Nevertheless, the presented simulation framework can be used for the modeling of such permanent effects and observing their consequences.

In general, the work can be considered as a basis for the creation of system-level debugging and validation environment. The unavailability of a satellite sub-system can be solved by its simulation on PC in the presented way. The PC interconnection with the satellite central bus can be done through modern digital analyzers. Such approach may have real-time limitations that should be additionally investigated.

Bibliography

[1] *Datasheet of NanoMind A702B OBC, GomSpace company.* http://www.gomspace.com/documents/GS-DS-NM702-2.1.pdf.

[2] Kurt Anderson. Low-cost, radiation-tolerant, on-board processing solution. *IEEE Aerospace Conference*, pages 1–8, March 2005.

[3] Sana Rezgui. RT ProASIC3: New RT Flash-Based FPGAs. CMOS Emerging Technologies Workshop, Whistler, Canada, May 2010.

[4] N. Battezzati, F. Decuzzi, L. Sterpone, and M. Violante. Soft Errors in Flash-based FPGAs: Analysis methodologies and first results. *International Conference on Field Programmable Logic and Applications*, pages 723 – 724, August-September 2009.

[5] J. L. Kaschmitter, D. L. Shaeffer, and N. J. Colella. Operation of commercial R3000 processors in the LEO space environment. *IEEE Transactions on Nuclear Science*, 38(6):1415–1420, December 1991.

[6] Microsemi Corporation. *SmartFusion Customizable System-on-Chip (cSoC)*, August 2011.

[7] Dolkart V. V., G. Kh. Novik, and I. S. Koltypin. *Miniaturized aerospace digital computers*. Soviet Radio, Moscow, 1967.

[8] California Polytechnic State University. The CubeSat Program. *CubeSat Design Specification*, 12 edition.

[9] Andrew S. Keys and Michael D. Watson. Radiation hardened electronics for extreme environments. *NASA Marshall Space Flight Center*.

[10] The official web-site of SERVIS project. http://www.usef.or.jp/english/f3_project/servis/f3_servis.html.

[11] Frank Ghenassia. *Transaction Level Modeling with SystemC TLM Concepts and Applications for Embedded Systems*. ISBN 10 0-387-26232-6. Springer, 2005.

[12] The official web-site of OVP project. http://www.ovpworld.org.

[13] D. Burlyaev and R. van Leuken. A simulator of on-board computers for evaluating fault-mitigation techniques. In *the ESA Small Satellites Systems and Services Symposium*, Grand Hotel Bernardin Convention Centre, Portoroz, Slovenia, June 2012.

[14] D. Burlyaev and R. van Leuken. SystemC-based on-board computer modeling for design fault-tolerance assessment. In *the 21st IEEE North Atlantic Test Workshop*, Woburn, MA(Greater Boston Area), USA, May 2012.

[15] D. Burlyaev and R. van Leuken. System fault-tolerance analysis of small satellite on-board computers. In *the XXVII Conference on Design of Circuits and Integrated Systems*, Avignon, France, November 2012.

[16] Rainer Sandau, Larry Paxton, and Jaime Esper. *Small Satellites for Earth Observation*. ISBN: 978-1-4020-6942-0. Springer, 2008.

[17] J. Puig-Suari, C. Turner, and R. J. Twiggs. Cubesat: The development and launch support infrastructure for eighteen different satellite customers on one launch. *15th Annual/USU Conference on Small Satellites*, (SSC01-VIIIb-5):1–5, 2001.

[18] J. Puig-Suari, C. Turner, and W. Ahlgren. Development of the standard CubeSat deployer and a CubeSat class picosatellite. *IEEE Aerospace Conference, year =*.

[19] The official web-site of Hiten project. http://www.isas.jaxa.jp/e/enterp/missions/hiten.shtml.

[20] The official web-site of TSUBASA project. http://www.jaxa.jp/projects/sat/mds1/index_e.html.

[21] The official web-site of QB50 project. https://www.qb50.eu/project.php.

[22] The official web-site of NASAs Cubesat Launch Initiative, August 2011. http://www.nasa.gov/directorates/heo/home/CubeSats_initiative.html.

[23] Tanya Vladimirova and Xiaofeng Wu. On-board partial run-time reconfiguration for pico-satellite constellations. *The First NASA/ESA Conference on Adaptive Hardware and Systems*, pages 262 – 269, June 2006.

[24] Lei Xing, Zhaowei Sun, and Guodong Xu. FPGA On-Board Computer design based on Hierarchical Fault tolerance. *The 2nd International Symposium on Systems and Control in Aerospace and Astronautics*, pages 1 – 5, December 2008.

[25] M.M.Ibrahim, A.M.Tobal, M.Y.E. Nahas, and M.K. Refai. FPGA-based On-Board Computer for LEO satellites. *IEEE International Conference on Space Science and Communication (IconSpace)*, pages 314 – 319, July 2011.

[26] James R. Wertz, Hans F.Meissinger, Lauri Hraft Newman, and Geoffrey N. Smit. *Mission Geometry; Orbit and Constellation Design and Management*. ISBN-10: 0792371488. Microcosm Press and Kluwer Academic Publisher, 1 edition, October 2001.

[27] Datasheet of Lithium UHF/VHF Radio LI-1. Astronautical development LLC company. http://www.astrodev.com/public_html2/downloads/datasheet/LithiumUserManual.pdf.

[28] Datasheet of Highly Integrated S-Band Transmitter for Pico and Nano Satellite. http://www.cubesatshop.com/index.php?page=shop.product_details&flypage=flypage.tpl&product_id=84&category_id=5&option=com_virtuemart&Itemid=67.

[29] TEKTRONIX company. *Guidelines for Lithium-Ion Battery Maintenance*, 001-1501-00 edition. http://www.newark.com/pdfs/techarticles/tektronix/LIBMG.pdf.

[30] PC/104 Consortium. *PC/104 Specification*, 2.3 edition, June 1996. http://versalogic.com/support/pdf/pc104-23.pdf.

[31] Pumpkin Incorporated, San Fransisco, CA, USA. *CubeSat Kit PCB Specification*.

[32] M. Duma, H. Urhan, O. Turhan, O. Kozal, and M. Gurun. A new generation On-Board Computer and Solid State Data Recorder suitable for SpaceWire Platforms. *The 3rd International Conference on Recent Advances in Space Technologies*, pages 429 – 432, June 2007.

[33] Hiroyuki Yashiro and Teruo Fujiwara. A high assurance on-line recovery technology for a Space On-board Computer. *The 5th International Symposium on Autonomous Decentralized Systems*, pages 47 – 56, August 2002.

[34] C.A. Hulme, H.H. Loomis, A.A.Ross, and Rong Yuan. Configurable Fault-Tolerant Processor (CFTP) for space based applications. *IEEE Aerospace Conference*, 4:2269 – 2276, March 2004.

[35] T.Takano, T. Yamada, K. Shutoh, and N. Kanekawa. In-orbit experiment on the fault-tolerant space computer aboard the satellite Hiten. *IEEE Transactions on Reliability*, 45:624 – 631, August 2002.

[36] Cedric V. W. Armstrong and Eli T. Fathi. A fault-tolerant multimicroprocessor-based computer system for space-based signal processing. *IEEE Micro*, 4:54 – 65, December 1984.

[37] GomSpace company. *Datasheet of NanoCam C1U camera, GS-DS-NANOCAM-1.1*, 1 edition, August 2011. http://www.gomspace.com/documents/GS-DS-NANOCAM-1.1.pdf.

[38] Atmel Corporation. *Passive InfraRed Reference Design (PIRRD) for SAM3S Motion Detector Camera*. http://atmel.com/dyn/resources/prod_documents/doc11091.pdf.

[39] ARM company. *Cortex-M3 Devices, Generic User Guide*, a edition, December 2010.

[40] Pumpkin, Inc. *CubeSat Kit Motherboard (MB)*, J edition, September 2009. http://cubesatkit.com/docs/datasheet/DS_CSK_MB_710-00484-D.pdf.

[41] Day-Young Kim, Ki-Ho Kwon, Jong-Wook Choi, Jong-In Lee, and Hak-Jung Kim. Design of a new On-Board Computer for the new KOMPSAT bus. *IEEE Aerospace Conference*, pages 1 – 12, March 2005.

[42] Surrey Satellite Technology US LLC. *Space GPS Receiver SGR-05U*, July 2011. http://www.sst-us.com/getdoc/c408ce26-bd0b-4c43-88e0-6999372040a1.

[43] F. Bruhn, E. Lamoureux, G. Chosson, J. Bergman, K. Yoshida, T.George, R. Thorslund, and J. Kohler. Bridging the space technology Valley of Death: two spaceflights in 2009 to validate advanced MEMS/Microtechnology systems and subsystems. *CANEUS workshop NASA Ames*, March 2009. http://www.aaerospace.com/index.php/inovator-on-rubin-92-h2-2009/19.html.

[44] Franz Newland, Elliott Coleshill, Ian DSouza, and Jeff Cain. Nanosatellite tracking of ships - Review of the first year of operations. *The 7th Responsive Space Conference*, (AIAA-RS7-2009-6005):1–6, April 2009.

[45] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. ISBN-10 3-540-35652-5. Springer, 2 edition, 2006.

[46] Michael L. Bushnell and Vishwani D. Agrawal. *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Kluver Academic Publishers, 2002.

[47] C. Miller, R.Owen, M. Rose, P.M.Rutt, J. Schaefer, and I.A.Troxel. Trends in radiation susceptibility of commercial DRAMs for space systems. *IEEE Aerospace conference*, pages 1 – 12, March 2009.

[48] J. J. Wang. Radiation effects in FPGAs. *The 9th Workshop on Electronics for LHC Experiments*, pages 1–10, September-October 2003.

[49] The official web-site of SPENVIS project. http://www.spenvis.oma.be/.

[50] The official web-site of APEXRAD empirical model. https://creme.isde.vanderbilt.edu/CREME-MC/help/apexrad.

[51] P.J. McNulty. Charged particles cause microelectronics malfunction in space. *Physics Today*, 36(1):9, January 1983.

[52] Solar flare proton and heavy ion modeling for single event effects. *NASA Preferred Reliability Practices*, (Practice No. PD-EC-1105):1–3, June 1995.

[53] Toshinori Kuwahara. *FPGA-based Reconfigurable On-Board Computing System for Space Application*. PhD thesis, the University of Stuttgart, October 2009.

[54] G.Gasiot, S. Uznanski, and P. Roche. SEE test and modeling results on 45nm SRAMs with different well strategies. *IEEE International Reliability Physics Symposium*, pages 407 – 410, May 2010.

[55] S.Rezgui, J. J.Wang, Y.Sun, B.Cronquist, and J.McCollum. New reprogrammable and non-volatile radiation tolerant FPGA: Rta3p. *Proceedings of the IEEE Aerospace Conference*, 2008.

[56] R. Czajkowski, M. P. Pagey, P. K. Samudrala, M. Goksel, and M. J. Viehman. Low power, high-speed radiation hardened computer & flight experiment. *IEEE Aerospace Conference*, pages 1 – 10, March 2005.

[57] Gennady Ivanovich Zebrev, Igor Olegovich Ishutin, Rustem Galeyevich Useinov, and Vasily Sergeyevich Anashin. Methodology of soft error rate computation in modern microelectronics. *IEEE Transactions on Nuclear Science*, 57(6):3725–3733, December 2010.

[58] Robert Baumann. The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction. *Digest. International Electron Devices Meeting*, pages 329 – 332, 2002.

[59] Leif Scheick. SEE measurement for the SDRAM and the Blackjack Application-Specific Integrated Circuit (ASIC). The Jet Propulsion Laboratory, December 1999.

[60] Nobuyasu Kanekawa, Eishi H. Ibe, Ta kashi S uga, and Yu taka Uematsu. *Dependability in Electronic Systems. Mitigation of Hardware Failures, Soft Errors, and Electro-Magnetic Disturbances.* ISBN 978-1-4419-6714-5. Springer, 2011.

[61] R.K. Lawrence. Radiation characterization of 512 Mb SDRAMs. *IEEE Radiation Effects Data Workshop*, July 2007.

[62] K. Grurmann, D. Walter, M. Herrmann, and F. Gliem. Studies of radiation SEE effects in NAND-Flash and DDR types of memories. R&D Final Presentation Days, ESTEC, Nordwijk, March 2011.

[63] D.Giot, P. Roche, G.Gasiot, L.Autran, and R.Harboe-Sorensen. Heavy ion testing and 3D simulations of MCU in 65nm standard SRAMs. *The 9th European Conference on Radiation and Its Effects on Components and Systems*, pages 1 – 6, September 2007.

[64] M. Baba et al. Installation and application of an intense 7Li(p,n) neutron source for 2090 MeV region. *Radiation Protection Dosimetry*, 123(1-4):1317, 2007.

[65] F.X.Ruckerbauer and G. Georgakos. Soft error rates in 65nm SRAMs–analysis of new phenomena. *The 13th IEEE International On-Line Testing Symposium*, pages 203 – 204, July 2007.

[66] E.H. Cannon, M.Cabanas-Holmen, J.Wert, T.Amort, R.Brees, J.Koehn, B.Meaker, and E.Normand. Heavy ion, high-energy, and low-energy proton SEE sensitivity of 90-nm RHBD SRAMs. *IEEE Transactions on Nuclear Science*, 57(6):3493 – 3499, December 2010.

[67] Kathrin Peter. Data distribution algorithms for reliable parallel storage on Flash memories. Computer science research. *Zuse Institute Berlin.*

[68] D.N. Nguyen, S. M. Guertin, and J. D. Patterson. Radiation tests on 2Gb NAND Flash memories. *IEEE Radiation Effects Data Workshop*, pages 121 – 125, July 2006.

[69] D.N. Nguyen, S.M. Guertin, G.M. Swift, and A.H. Johnston. Radiation effects on advanced Flash memories. *IEEE Transactions on Nuclear Science*, 46(6):1744–1750, December 1999.

[70] G.Cellere et al. A model for TID effects on FG memory cells. *IEEE Transactions on Nuclear Science*, 51(6):3753 – 3758, December 2004.

[71] Farokh Irom and Duc N. Nguyen. Single event effect characterization of high density commercial nand and nor nonvolatile flash memories. *IEEE TRANSACTIONS ON NUCLEAR SCIENCE*, pages 2547–2553, December 2007.

[72] N. Battezzati, S. Gerardin, A. Manuzzato, A. Paccagnella, S. Rezgui, L. Sterpone, and M. Violante. On the evaluation of radiation-induced transient faults in Flash-based FPGAs. *The 14th IEEE International On-Line Testing Symposium*, pages 135 – 140, July 2008.

[73] Ramin Roosta. A comparison of radiation-hard and radiation-tolerant FPGAs for space applications. *NASA Electronic Parts and Packaging Program*, (JPL D-31228), December 2004.

[74] Dae-Soo Oh, Dong-Soo Kang, and Kyoung-Son Jhang. Design and implementation of a radiation tolerant On-Board Computer for science technology Satellite-3. *NASA/glsesa Conference on Adaptive Hardware and Systems*, pages 17 – 23, June 2010.

[75] F.Lima, L.Carro, and R. Reis. Designing fault tolerant systems into SRAM-based FPGAs. *Design Automation Conference*, pages 650 – 655, June 2003.

[76] Gregory R. Allen and Gary M. Swift. Single event effects test results for advanced field programmable gate arrays. *IEEE Radiation Effects Data Workshop*, pages 115 – 120, July 2006.

[77] N.Battezzati, L.Sterpone, M.Violante, and F.Decuzzi. A new software tool for static analysis of SET sensitiveness in Flash-based FPGAs. *The 18th IEEE/IFIP VLSI System on Chip Conference (VLSI-SoC)*, pages 79 – 84, September 2010.

[78] F.L. Kastensmidt, E.C.P. Fonseca, R.G. Vaz, O.L. Goncalez, R. Chipana, and G.I. Wirth. TID in Flash-based FPGA: Power supply-current rise and logic function mapping effects in propagation-delay degradation. *IEEE Transactions On Nuclear Science*, 58(4):1927 – 1934, August 2011.

[79] S. Rezgui et al. New methodologies for SET characterization and mitigation in Flash-based FPGAs. *IEEE Transactions on Nuclear Science*, 54(6):2512–2524, December 2007.

[80] Sana Rezgui. *Radiation-Tolerant ProASIC3 FPGAs Radiation Effects.* Actel Corporation, 2061,Stierlin Court, Mountain View, CA,USA, April 2010.

[81] Actel Corporation. *Radiation-Tolerant ProASIC3 Single-Event Latch-Up*, April 2010. Test Report.

[82] S. Rezgui, J. J. Wang, Y. Sun, B. Cronquist, and J. McCollum. TID characterization of 0.13-micrometer Flash-based FPGAs. *RADECS Workshop*, pages 1–6, 2008.

[83] S. Rezgui, J. J. Wang, Y. Sun, B. Cronquist, and J. McCollum. New reprogrammable and non-volatile radiation tolerant FPGA: RTA3P. *Proc. IEEE Aerospace Conference*, pages 1–11, March 2008.

[84] G. Allen, S. McClure, S. Rezgui, and J. J. Wang. TID characterization results of Actel ProASIC3, ProASIC3E, and IGLOO Flash-based Field Programmable Gate Arrays. *Proc. Military Aerospace Programmable Logic Devices (MAPLD)*, pages 1–11, 2008.

[85] J.W. Howard Jr., M.A. Carts, R. Stattel, C.E. Rogers, T.L. Irwin, C. Dunsmore, J.A. Sciarini, and K.A. LaBel. Total dose and single event effects testing of the Intel Pentium III and AMD K7 microprocessor. *IEEE Radiation Effects Data Workshop*, pages 38 – 47, July 2001.

[86] Actel Corporation. *Actel SmartFusion Microcontroller Subsystem Users Guide*, 50200250-1 edition, May 2010.

[87] Actel Corporation. *Actel SmartFusion FPGA Fabric Users Guide*, 50200249-0 edition, March 2010.

[88] Timothy R. Oldham, M. Friendlich, M. A. Cartsand C. M. Seidleck, and Kenneth A. LaBel. Effect of radiation exposure on the endurance of commercial NAND Flash memory. *IEEE Trnasactions on Nuclear Science*, pages 3280 – 3284, December 2009.

[89] T. R. Oldham, R. L. Ladbury, M. Friendlich, H. S. Kim, M. D. Berg, T. L. Irwin, C. Seidleck, and IEEE K. A. LaBel, Member. SEE and TID characterization of an advanced commercial 2Gbit NAND Flash nonvolatile memory. *IEEE Trnasactions on Nuclear Science*, 53(6):3217–3222, December 2006.

[90] Farokh Irom and Duc N. Nguyen. Single event effect characterization of high density commercial NAND and NOR Nonvolatile Flash memories. *IEEE Transactions on Nuclear Science*, pages 2547 – 2553, December 2007.

[91] F.Irom, D.N.Nguyen, R.Harboe-Sorensen, and A.Virtanen. Comparison of TID response and SEE characterization of single- and multi-level high density NAND Flash memories. *European Conference on Radiation and Its Effects on Components and Systems*, pages 606–608, September 2009.

[92] Arbi V. Karapetian, Raphael R. Some, and John J. Beahan. Radiation fault modeling and fault estimation for a COTS based space-borne supercomputer. *Proceeding IEEE Aerospace Conference*, 2002.

[93] M.S. Reorda, M. Violante, C. Meinhardt, and R. Reis. An On-Board Data-Handling Computer for deep-space exploration built using Commercial-Off-the-Shelf SRAM-based FPGAs. *The 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 254 – 262, October 2009.

[94] Xinping Zhu and Wei Qin. Prototyping a fault-tolerant multiprocessor SoC with run-time fault recovery. *The 43rd ACM/IEEE Design Automation Conference*, pages 53 – 56, September 2006.

[95] The official web-site of Green Hills Software company. http://www.ghs.com/.

[96] C. C. Li and W. K. Fuchs. CATCH - compiler-assisted techniques for checkpointing. *FTCS-20,IEEE Computer Society*, page 7481, 1990.

[97] P.P. Shirvani, N. Oh, E.J. McCluskey, D.L. Wood M.N.and Lovellette, and K.S. Wood. Software-implemented hardware fault tolerance experiments COTS in space.

[98] L. Chen and A. Avizienis. N-version programming: A fault tolerance approach to reliability of software operation. *FTCS-8*, page 39, 1978.

[99] B. Randell. System structure for software fault tolerance. *Processing International Conference on Reliable software*, page 437449, 1975. New York, NY, USA: ACM Press.

[100] Demid Borodin. *Performance-Oriented Fault Tolerance in Computing Systems.* PhD thesis, TU Delft, Delft, the Netherlands, 2010.

[101] M. Namjoo and E. J. McCluskey. Watchdog processors and capability checking. *FTCS-12. Washington, DC, USA: IEEE Computer Society*, page 245248, 1982.

[102] A.T. Tai, K.S. Tso, L. Alkalai, S.N. Chau, and W.H. Sanders. Low-cost error containment and recovery for onboard guarded software upgrading and beyond. *IEEE Transactions on Computers*, 51:121 – 137, February 2002.

[103] Krzysztof Iniewski. *Radiation Effects In Semiconductors.* Taylor and Francis Group, 2011.

[104] C.C. Yui et al. SEU mitigation testing of Xilinx Virtex II FPGAs. *IEEE Radiation Effects Data Workshop*, pages 92 – 97, July 2003.

[105] K. J. Chang and Y. Y. Chen. System-level fault injection in SystemC design platform. *Proc. 8th Int. Symposium on Advanced Intelligent Systems*, pages 354–359, 2007.

[106] T. Grtker, S.Liao, G.Martin, and S.Swan. *System Design with SystemC.* ISBN 978-1-4020-7072-3. Kluwer Academic Publishers, 2002.

[107] Leena Singh, Leonard, Drucker, and Neyaz Khan. *Advanced Verification Techniques: A SystemC Based Approach for Successful Tapeout.* ISBN: 1-4020-7672-X. Kluwer Academic Publishers, 2004.

[108] J.-C.Ruiz, P. Yuste, P. Gil, and L. Lemus. On benchmarking the dependability of automotive engine control applications. *International Conference on Dependable Systems and Networks*, pages 857 – 866, June-July 2004.

[109] R.Mariani, G.Boschi, and F.Colucci. Using an innovative SoC-level FMEA methodology to design in complience with IEC61508. *Proc. Design, Automation & Test in Europe Conf. &Exhibition*, pages 492–497, 2007.

[110] Y. Y. Chen, Chung-Hsien Hsu, and Kuen-Long Leu. SoC-level risk assessment using FMEA approach in system design with SystemC. *IEEE International Symposium on Industrial Embedded Systems*, pages 82–89, July 2009.

[111] G. Beltrame, C. Bolchini, L. Fossati, A. Miele, and D. Sciuto. A framework for reliability assessment and enhancement in multi-processor Systems-On-Chip. *The 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, pages 132 – 142, 2007.

[112] O. Ruano, J.A. Maestro, P.Reyes, and P. Reviriego. A simulation platform for the study of soft errors on signal processing circuits through software fault injection. *IEEE International Symposium on Industrial Electronics*, pages 3316–3321, 2007.

[113] K.Rothbart and et al. High level fault injection for attack simulation in smart cards. *The 13th Asian Test Symposium*, pages 118 – 121, November 2004.

[114] K.Rothbart and et al. A smart card test environment using multi-level fault injection in SystemC. *The 6th IEEE Latin-American Test Workshop*, pages 103–108, March-April 2005.

[115] Kun-Jun Chang, Yung-Yuan Chen, and Jian-Min Peng. SoC-level fault injection methodology in SystemC design platform. *Asia Simulation Conference - the 7th International Conference on System Simulation and Scientific Computing*, pages 680 – 687, October 2008.

[116] Yung-Yuan Chen, Chung-Hsien Hsu, and Kuen-Long Leu. Analysis of system bus transaction vulnerability in SystemC TLM design platform. *Proc. the 3rd WSEAS International Conference on Computer Engineering and Applications*, (ISBN:978-960-474-41-3):284–289, 2009.

[117] Microsemi Corporation. *SmartFusion: System Power Optimization Using Low Power Modes*, application note AC364 edition, March 2011.

[118] G. Allen, S. McClure, S. Rezgui, and J.J. Wang. Total ionizing dose characterization results of Actel ProAsic3, ProAsic3L, and IGLOO Flash-based FPGA. The Jet Propulsion Laboratory, California Institute of Technology and Actel Corporation, September 2008. https://nepp.nasa.gov/mapld_2008/presentations/w/10%20-%20Allen_Gregory_mapld08_pres_1.pdf.

[119] Imperas Software Limited. *Using OVP Models in SystemC TLM2.0 Platforms*, 1.10 edition, September 2011.

[120] Imperas Software Limited. *OVPsim and Imperas CpuManager User Guide*, 2.0.43 edition, November 2011.

[121] Charles F. Van Loan. *Computational Frameworks for the Fast Fourier Transform.* ISBN 0-89871-285-8. the Society for Industrial and Applied Mathematics, Cornell University, Ithaca, New York, USA, 1992.

[122] Joris van Emden. Embedded JPEG codec library. http://sourceforge.net/projects/mb-jpeg.

[123] Actel Corporation. *CoreEDAC Handbook*, January 2006.

[124] Actel Corporation. *Using EDAC RAM for RadTolerant RTAX-S FPGAs and Axcelerator FPGAs*, application note AC273 edition, July 2006.

[125] The official web-site of CatapultC Synthesis tool from Mentor Graphics company. http://www.mentor.com/esl/catapult/overview.

Appendix A

A.1 SPENVIS settings for Radiation Environment Estimation

Mission overview
Orbit around: Earth
Number of mission segments: 1
Mission start: 01/01/2011 00:00:00
Mission end: 31/12/2013 00:00:00
Mission duration: 1095.00 days (3.00 years)
Satellite axis: velocity vector

Orbit type: general
Apogee: 750.00 km
Perigee: 750.00 km
Inclination: 98.00
R. A. Ascending Node: 0.00
Argument of Perigee: 0.00
True Anomaly: 0.00
Period: 1.66 hrs
Number of orbits: 14.45
Duration: 1.00 days
Orbit start: 01/01/2011 00:00: 0.0
Orbit end: 02/01/2011 00:00: 0.0
Segment end: 31/12/2013 00:00: 0.0
Segment length: 1095.00 days

Trapped proton model: CRRESPRO Quiet
Internal magnetic field model: DGRF 1990 updated to 1990.0
External magnetic field model: Olson-Pfitzer 1977

Solar flare model: CREME-96 (peak 5 min-1)
Magnetic shielding: eccentric dipole/quiet magnetosphere/unchanged magn. mom./all arrival directions
Ion range: H - Ni

Overview input
Ion range: H - Ni
GCR model: ISO 15390
Solar activity data: May 1996

Magnetic shielding: eccentric dipole/quiet magnetosphere/unchanged magn. mom./all arrival directions

Test 1. Environment assessment

Overview input

Particle spectra : trapped protons + solar particles (H - Ni) + GCR particles (H - Ni)
Spacecraft shielding thickness (Al equivalent) : 0.15 cm

Appendix B

B.1 MATLAB Fault rate estimation algorithm

```
\% Fault rate calculation
clear all;
\% Algorithm
flux_data = xlsread('spectra.xls','flux_data');
LET_flux=flux_data(:,1); \%MeV*cm^2/g
Integral_flux=flux_data(:,2);\%1/m^2/sr/sec

\%experiment
exper_data = xlsread('spectra.xls','sram2_fig2_65');
LET_exper=exper_data(:,1);\%MeV*cm^2/mg!
Cros_section_exper=exper_data(:,2);\%cm^2/bit

\%conversion everything to the same units
LET_flux_uni=LET_flux/10000; \% MeV*m^2/g
Integral_flux_uni=Integral_flux;\%1/m^2/sr/sec

LET_exper_uni=LET_exper*1000/10000;\%MeV*m^2/g
Cros_section_exper_uni=Cros_section_exper/10000;\%m^2/bit

\% integration
sum=0;
previous=0;
for i=1:1:1000

   for j=1:1:length(Cros_section_exper_uni)
       if (LET_flux_uni(i)<LET_exper_uni(j))
          j=j-1;

          if j<1
              j=1
          end;

          break;
       end;
   end;

   sum=sum+Integral_flux_uni(i)*Cros_section_exper_uni(j)*(
      LET_flux_uni(i)-previous) % events/bit/sec
   previous=LET_flux_uni(i);
end;
```

Appendix C

The checking of the model work has been done by sending the message from OBC #1 to OBC #2 through modeled I2C bus. Some steps during the communication included *printf* operation to see the correctness of the event sequence (see the next Section). After sending the data from OBC #1 to OBC #2,OBC #2 checks the received data:

Reg checking::23
Reg checking::45
Reg checking::ab
Reg checking::cd
Reg checking::ef

These 4 bytes are exactly the same that were sent by OBC #1. The next lines are derived from the *C* code executed by OBC #1 :

$write\_data[0] = 0xab;$
$write\_data[1] = 0xcd;$
$write\_data[2] = 0xef;$
$uint16\_tstart\_address = 0x2345;$

First two bytes 0x23 and 0x45 is the serial address of the receiver-OBC. The receiver was found, bytes were transfered, and the communication have been finished (see the communication printout in the next Section).Thus, the communication happened in correct way, the driver software is working, and FSM for I2C.obj has been created without mistakes.

C.1 Printout of Two OBCs communication through I2C bus

```
1
  ||PLATFORM||::Starting the timing for simulation
  ||COMPUTER_1|| before start
                            ||COMPUTER_2||before start
  The I2C has been enabled
6                          The I2C has been enabled
  ||COMPUTER_1|| this_i2c->target_addr::e::e
  Line is not occupied-> we now occupy the line::COMPUTER_I2C_0_1
  ||COMPUTER::I2C_0_1|| Execution of 0x08
  ||I2C_0_1||the 1st interrupt is sent
11 ||COMPUTER_1|| -interrupt_ I2C0_IRQHandler triggered--
  ||COMPUTER_1|| isr::ST_START::0x08
  ||COMPUTER_1|| isr::ST_RESTART::0x10
  ||I2C_0_1|| I am sending the next target address::e
```

```
                ||I2C Decoder||:: I am sending out the message to every
                   target that I have::1::Socket id::0
16                  ||I2C_0_2||:: just received the request from the I2C
                       decoder
                        ||I2C decoder::data||::len::adr|e|1|0|
                    ||I2C_0_2|| received address equals my serial number::::e
                    ||I2C_0_2||I am calling the 1st interrupt at the receiver
                        side
    ||COMPUTER_1|| main1...
21                  ||COMPUTER_2||  interrupt_ I2C0_IRQHandler triggered--
                    ||COMPUTER_2||  isr::ST_SLAVE_SLAW
                    ||COMPUTER::I2C_0_2|| I am sending ACK to the source to
                       say                                    that target is
                        found
                ||I2C Decoder||:: I am sending out the message to every
                   target that I have::0::Socket id::1
    ||I2C_0_1||:: just received the request from the I2C decoder
26      ||I2C decoder::data||::len::adr|1|1|1|
    ||I2C_0_1||::ACK has been received about the proper target address->
       going to send the 1 byte
    ||COMPUTER_1|| -interrupt_ I2C0_IRQHandler triggered--
    ||COMPUTER_1|| isr::ST_SLAW_ACK::0x18
    ||I2C_0_1||I am sending the DATA to I2C Decoder in state 0x18
31              ||I2C Decoder||:: I am sending out the message to every
                   target that I have::1::Socket id::0
                ||I2C_0_2||:: just received the request from the I2C decoder
                    ||I2C decoder::data||::len::adr|23|1|0|
                ||I2C_0_2||I have received new DATA going to call the
                   interrupt and send ACK    back
                    ||COMPUTER_2||  --interrupt_ I2C0_IRQHandler triggered--
36                  ||COMPUTER_2||  isr::ST_RDATA
                     ||COMPUTER_2||     I am saving::23::0
                    ||COMPUTER::I2C_0_2|| I am sending ACK to the source to
                       say that target is found
                ||I2C Decoder||:: I am sending out the message to every
                   target that I have::0::Socket id::1
    ||I2C_0_1||:: just received the request from the I2C decoder
41  ||I2C_0_1||::ACK has been received about the proper target address->
       going to send the 1 byte
    ||COMPUTER_1|| interrupt_ I2C0_IRQHandler triggered--
     ||COMPUTER_1|| isr::ST_TX_DATA_ACK::0x28
    ||I2C_0_1||I am sending the DATA to I2C Decoder in state 0x18
                ||I2C Decoder||:: I am sending out the message to every
                   target that I have::1::Socket id::0
46                  ||I2C_0_2||:: just received the request from the I2C
                       decoder
                    |I2C_0_2||I have received new DATA going to call the
                       interrupt and send ACK
                                                 back
                    ||COMPUTER_2|| interrupt_ I2C0_IRQHandler   triggered--
                    ||COMPUTER_2||  isr::ST_RDATA
                    ||COMPUTER::I2C_0_2|| I am sending ACK to the source to
                       say that target is found
```

```
51              ||I2C Decoder||:: I am sending out the message to every
                   target that I
    ||I2C_0_1||:: just received the request from the I2C decoder
    ||I2C_0_1||::ACK has been received about the proper target address->
       going to send the 1 byte
    ||COMPUTER_1|| interrupt_ I2C0_IRQHandler triggered--
    ||COMPUTER_1|| isr::ST_TX_DATA_ACK::0x28
56  ||I2C_0_1||I am sending the DATA to I2C Decoder in state 0x18
                ||I2C Decoder||:: I am sending out the message to every
                   target that I have::1::Socket id::0
                    ||I2C_0_2||:: just received the request from the I2C
                       decoder
                    ||I2C_0_2||I have received new DATA going to call the
                       interrupt and send ACK back
                    ||COMPUTER_2|| interrupt_ I2C0_IRQHandler triggered--
61                  ||COMPUTER_2|| isr::ST_RDATA
                    ||COMPUTER::I2C_0_2|| I am sending ACK to the source to
                       say that target is found
                ||I2C Decoder||:: I am sending out the message to every
                   target that I have::0::Socket id::1
    ||I2C_0_1||:: just received the request from the I2C decoder
       I2C decoder::data::len::adr|1|1|1|
66  ||I2C_0_1||::ACK has been received about the proper target address->
       going to send the 1 byte
    ||COMPUTER_1|| interrupt_ I2C0_IRQHandler triggered--
    ||COMPUTER_1|| isr::ST_TX_DATA_ACK::0x28
    ||I2C_0_1||I am sending the DATA to I2C Decoder in state 0x18
    ||I2C_0_1||I am sending the DATA to I2C Decoder::cd
71              ||I2C Decoder||:: I am sending out the message to every
                   target that I have::1::Socket id::0
                    ||I2C_0_2||:: just received the request from the I2C
                       decoder
                        I2C decoder::data::len::adr|cd|1|0|
                    ||I2C_0_2||I have received new DATA going to call the
                       interrupt and send ACK back
                    ||COMPUTER_2|| interrupt_ I2C0_IRQHandler triggered--
76                  ||COMPUTER_2|| isr::ST_RDATA
                    ||COMPUTER_2|| I am saving::cd::3
                    ||COMPUTER_2|| isr::Nothing has been executed
                    ||COMPUTER::I2C_0_2|| I am sending ACK to the source to
                       say that target is found
                ||I2C Decoder||:: I am sending out the message to every
                   target that I have::0::Socket id::1
81  ||I2C_0_1||:: just received the request from the I2C decoder
       I2C decoder::data::len::adr|1|1|1|
    ||I2C_0_1||::ACK has been received about the proper target address->
       going to send the 1 byte
    ||COMPUTER_1|| interrupt_ I2C0_IRQHandler triggered--
    ||COMPUTER_1|| isr::ST_TX_DATA_ACK::0x28
86  ||I2C_0_1||I am sending the DATA to I2C Decoder::ef
                ||I2C Decoder||:: I am sending out the message to every
                   target that I have::1::Socket id::0
                    ||I2C_0_2||:: just received the request from the I2C
```

```
                    decoder
                     I2C decoder::data::len::adr|ef|1|0|
                  ||I2C_0_2||I have received new DATA going to call the
                    interrupt and send ACK back
91                ||COMPUTER_2||  interrupt_ I2C0_IRQHandler triggered--
                  ||COMPUTER_2||  isr::ST_RDATA
                  ||COMPUTER_2|| I am saving::ef::4
                  ||COMPUTER_2||  isr::Nothing has been executed
                  ||COMPUTER::I2C_0_2|| I am sending ACK to the source to
                    say that target is found
96             ||I2C Decoder||:: I am sending out the message to every
                  target that I have::0::Socket id::1
   ||I2C_0_1||:: just received the request from the I2C decoder
     I2C decoder::data::len::adr|1|1|1|
   ||I2C_0_1||::ACK has been received about the proper target address->
     going to send the 1 byte
   ||COMPUTER_1|| interrupt_ I2C0_IRQHandler triggered--
101||COMPUTER_1|| isr::ST_TX_DATA_ACK::0x28
   ||COMPUTER_1|| 0x28:: 3rd option
   ||COMPUTER::I2C_0_1|| I am sending STOP signal for everybody to indicate
     the end of sending
             ||I2C Decoder||:: I am sending out the message to every
                target that I have::1::Socket id::0
             ||I2C_0_2||:: just received the request from the I2C decoder
106             I2C decoder::data::len::adr|2|1|1|
             ||I2C_0_2||I have received new DATA going to call the
                interrupt and send ACK back

    ||COMPUTER_1|| main2...
             ||COMPUTER_2||  interrupt_ I2C0_IRQHandler triggered--
111           ||COMPUTER_2|| isr::ST_RSTOP
             ||COMPUTER_2||  c4...
             Reg checking::23
             Reg checking::45
             Reg checking::ab
116          Reg checking::cd
             Reg checking::ef
             ||COMPUTER_2||  finishing...1
    ||COMPUTER_1|| finishing...1
  SystemC: simulation stopped by user.
121 ||PLATFORM||::Finished sc_main.
```

Appendix D

D.1 Top object of OBC model - code explanation

This Section covers some important parts of the model code to clarify the modeling approach. Listing D.1 shows the definition of model components. *I2CDecoder* is the I2C Decoder that provides the interconnection between the computers. Parameters $< 2, 2 >$ means that the I2C Decoder has 2 initiator sockets and 2 target sockets, which is expected since each OBC model has one target socket and one initiator socket towards I2C Decoder. *reset_wire* and *reset_wire1* are signals from watchdog towards the CPU to simulate the reset procedure of the CPU. Any I2C controller has configuration registers that are modeled with the memory regions *PerifBBReg* and *PerifBBReg_2* (bit-bang regions).

Listing D.1: The definition of model components

```
 I2Cdecoder  <2,2> I2CDecoder1;
///*** OBC1 (PCB 1)
decoder  <2,6> Decoder1;///Internal
Memory_module eFlash;
Memory_module eSRAM;
Memory_module DRAM;
armm  cpu1;
watchdog  wdog1;
icmNetObject reset_wire; ///reset wire

///---Peripherals


Memory_module PerifBBReg;
I2C  I2C_0_1;


 ///***OBC2 (PCB 2)
decoder  <2,6> Decoder2;///Internal
Memory_module eFlash2;
Memory_module eSRAM2;
Memory_module DRAM2;
armm  cpu2;
watchdog  wdog2;
icmNetObject reset_wire2;///reset wire
///---Peripherals
Memory_module PerifBBReg_2;
I2C  I2C_0_2;
```

Different memory storages at different positions of the SmartFusion memory map are modeled with different objects of *Memory_model* type. During the object creation the memory parameters are defined: volume, name, etc. (see Listing D.2). Memory ranges are defined in the constructor (see Listing D.3).

Listing D.2: The Model components initialization

```
    , Decoder1("Decoder1")
30  , eFlash ("eFlash", "sp1", 0x100000,4,"flash",0,0)
    , eSRAM ("eSRAM", "sp1", 0x100000,4,"sram", 36,12)
    , DRAM ("DRAM", "sp1", 0x2000000,4,"flash", 36,12)
    , cpu1 ( "cpu1", 0, ICM_ATTR_DEFAULT|ICM_ATTR_SIMEX, attrsForcpu(variant))
    , wdog1("wdog1","sp1",0x1000,4)
35  , reset_wire("resetNet")

    ///--- Peripherals
    ,I2CDecoder1("I2CDecoder1")
    ,PerifBBReg("PerifBBReg", "sp1", 0x1000000,4,"sram", 36,12)
40  ,I2C_0_1 ("I2C_0_1", "sp1", 0x1000,4)

    ///**** OBC2 (PCB 2)
    , Decoder2("Decoder2")
    , eFlash2 ("eFlash2", "sp1", 0x100000,4,"flash",0,0)
45  , eSRAM2 ("eSRAM2", "sp1", 0x100000,4,"sram", 36,12)
    , DRAM2 ("DRAM2", "sp1", 0x2000000,4,"flash", 36,12)
    , cpu2 ( "cpu2", 1, ICM_ATTR_DEFAULT|ICM_ATTR_SIMEX, attrsForcpu(variant))
    , wdog2("wdog2","sp1",0x1000,4)
    , reset_wire2("resetNet2")
50  , PerifBBReg_2 ("PerifBBReg_2", "sp1", 0x10000000,4,"sram",0,0)
    , I2C_0_2 ("I2C_0_2", "sp1", 0x1000,4)
```

Listing D.3 also shows the structure of the interconnections, e.g. *cpu1* is connected to *Decoder1*. *cpu1* uses 2 target sockets of Decoder1 #0 and #1 (first 3 line of Listing D.3). These two interconnections correspond to Data and Address buses.

Listing D.3: The Model components' constructor

```
    cpu1.INSTRUCTION.socket(Decoder1.target_socket[0]); /// CPU
    cpu1.DATA.socket(Decoder1.target_socket[1]);

55  /// Decoder1 slaves
    ///-- embedded SRAM memory
    Decoder1.initiator_socket[0](eSRAM.sp1); /// Memory >> Embedded SRAM
    Decoder1.setDecode(0, 0x0, 0xfffff);
    ///-- embedded Flash memory
60  Decoder1.initiator_socket[1](eFlash.sp1); /// Memory >> Embedded Flash
    Decoder1.setDecode(1, 0x00100000, 0x001fffff);
    ///-- Decoder1 connect to Watchdog slave
    Decoder1.initiator_socket[2](wdog1.sp1);/// Watchdog
    Decoder1.setDecode(2,0x40006000,0x40006fff);
65  ///-- DRAM memory
    Decoder1.initiator_socket[3](DRAM.sp1); /// Memory >> External DRAM
    Decoder1.setDecode(3, 0x60100100, 0x621000ff);
```

```
    /// -***- Peripherals -***-
70  Decoder1.initiator_socket[4](PerifBBReg.sp1); /// Memory of Peripheral
                        ///Bit-Band (BB) Region
    Decoder1.setDecode(4, 0x52000000,0x52ffffff);

    ///-- I2C_0_1 address space and registers
75  Decoder1.initiator_socket[5](I2C_0_1.sp1); /// Memory >> I2C
    Decoder1.setDecode(5, 0x40002000,0x44002fff);
    I2C_net.connect(&I2C_0_1,PerifBBReg.getMemory()->get_mem_ptr()+0x40000);
    /// I2C connection with the I2C net through TLM
    I2C_0_1.socket_send.bind(I2CDecoder1.target_socket[0]);
80  I2C_0_1.socket_receive.bind(I2CDecoder1.initiator_socket[0]);
    I2C_0_1.register_decoder(&(I2CDecoder1.BUSY),&(cpu1)); /// registration of
              ///the decoder and the corresponding processor
```

Listing D.3 represents how memory ranges for different memory storages can be specified. We also can see how the sockets of *I2C.obj* are connected to *I2CDecoder*.

TLM methodology simplifies the communication procedures and the model components: the data sending is based on a package sending. The high observability of the model structure allows the tuning of TLM model to a particular interconnection architecture. It is also possible to define the latency of sending the particular amount of data.

SystemC-based On-board Computer Modeling for Design Fault-Tolerance Assessment

Burlyaev Dmitry (the presenter, a student)
EEMCS/ME/CAS
Delft University of Technology
Delft, the Netherlands
burlyaev.dmitry@gmail.com
Tel.: +31634767736

Rene van Leuken
EEMCS/ME/CAS
Delft University of Technology
Delft, the Netherlands
t.g.r.m.vanleuken@tudelft.nl

***Abstract*— Space radiation can induce catastrophic faults in electronic systems. Since CubeSat satellites utilize Commercial Off-The-Shelf (COTS) components that are radiation-sensitive by nature, the CubeSat On-Board Computers (OBCs) have to use fault mitigation techniques to prevent mission failures. The traditional analysis of the OBC fault-tolerance is limited due to the hardware unavailability at early development stages and low interpretability of radiation tests. This paper presents a SystemC-based modeling methodology which allows the evaluation of the fault mitigation techniques and the observation of fault consequences. The methodology includes the use of fault models of possible radiation effects in different electronic components that are based on published empirical observations. A dedicated fault-injection mechanism controls and tracks the fault introduction into the system model during real CubeSat control algorithm execution. Consequently, we can trace the injected faults in the different components and observe the resulting errors. This allows us to assess fault mitigation techniques as will be shown in several examples, an attitude control and a JPEG compression algorithm. The work shows how the error of the attitude control algorithm can be minimized to $1.7 \cdot 10^{-7}$rad/sec and the number of accumulated memory bit-flips can be reduced by 80-87% during the image compression.**

Keywords- CubeSat, SystemC modeling, fault mitigation techniques, system-on-chip, FMEA.

I. INTRODUCTION

Modern electronic systems are becoming more and more complex incorporating a large number of transistors whose sizes are shrinking from one generation to another. When chip fabrication entered the deep submicron technology, the reliability issues arose on the component level and on the system level consequently [1-3]. One of such existing issues is radiation-induced soft errors that are common not only in the open space but in terrestrial conditions too [4]. Thus, fault mitigation techniques became the integral part of any system level design that prevents system failures and disastrous consequences in such dependable applications as satellites, avionics, medical, and automotive devices.

An important question in the design of dependable systems is how to assess the efficiency of the applied fault mitigation techniques for particular application. Such approach as testing for radiation-induced faults does not necessarily guarantee specific errors, nor explain their nature or behavior [5]. Another problem is to estimate the influence of the radiation sensitivity of each component on the system level behavior taking into account that modern components incorporate sub-systems of different technologies. However, both of these issues can be resolved by system and fault modeling that can reveal the most vulnerable parts of the design. Due to the high complexity of each component and the system as a whole, the utilization of high level abstraction modeling language, such as SystemC, is imperative.

SystemC is a system-level modeling language that allows utilizing a mixture of various abstraction levels [6]. SystemC-based design platforms take advantage of Intellectual Property (IP) reuse to reduce design complexity, development cost and time.

Since modern electronic devices can incorporate both a processor and Field-Programmable Gate Array (FPGA) (or their variations, e.g. a processor realized in FPGA), the requirement on the hardware-software co-modeling and co-simulation is essential for early development stages. This requirement can be satisfied with SystemC Transaction-Level Modeling (TLM) methodology [7]. TLM cleanly dissociates functions and communications in SystemC models reducing the modeling complexity and the simulation time.

System vulnerability can be revealed by Failure Mode and Effect Analysis (FMEA) methodology [8] that is usually accompanied with the fault injection techniques to analyze the impact of system/component failures and to measure the risks of the system [9]. A common SystemC fault injection approach is the insertion of Fault Injection Module (FIM) into the interconnections of the functional blocks, where FIM is the controller of a fault injection process that can be centralized [11] or distributed [9].

The research in this paper was carried out under a contract with Innovative Solutions In Space B.V.

However, until now the SystemC TLM design level modeling with faults injection has been applied only to SoC-scale systems [9, 10] without consideration of software-hardware co-design, fault mitigation techniques implementation, and a comprehensive fault model library.

This work is focused particularly on the software-hardware co-modeling that provides an opportunity to understand the system behavior when diverse faults are introduced into different components. The introduced approach makes it possible to assess the efficiency of the applied fault mitigation techniques on the system level.

This work has been done in the context of satellite applications, in particular for CubeSat [12] On-Board Computers (OBCs). This category of applications is naturally exposed to the high level of radiation and, consequently, to diverse radiation-induced faults. However, the presented approach has a generic character, which makes it applicable for any electronic system.

The rest of the paper is organized as follows. Section II presents the modeling approach of the OBC. Section III observes how the fault injection technique has been integrated to the OBC system model. Section IV demonstrates several case studies (Fibonacci sequence calculation, JPEG image compression, and Kalman Filtering for Satellite Attitude Control) that show how mitigation techniques can be assessed and hardware-software co-modeling implemented.

II. SYSTEM MODEL

System-on-Chip (SoC) with CPU and FPGA is chosen as a core component of the OBC model to cover the most general OBC architecture. The SmartFusion SoC [13] is one of the most appropriate candidates for the satellite OBC that has to work in the space radiation environment at altitudes lower 750 km. Thus, it is chosen as a reference architecture for this work.

SmartFusion SoC includes Flash-based FPGA that is more immune to soft errors [14] than SRAM-based ones. SmartFusion SoC incorporates the majority of on-chip subsystems that can be met in the advanced Commercial Of-The-Shelf (COTS) electronics: a hard-core Cortex-M3, Flash-based FPGA fabric, Embedded Flash and SRAM memories, Analog subsystem, AMBA bus, etc.

A. OBC architecture modeling

The model structure of the OBC is presented in Fig.1; it incorporates the above-mentioned Cortex-M3 CPU, memories, the central bus (the Decoder plays the role of the central AMBA bus), FPGA fabric, a watchdog, and timers.

The existing instruction-accurate model of the processor Cortex-M3 has been obtained from the Open Virtual Platforms (OVP) project [14]. OVP provides a SystemC TLM wrapper for the Cortex-M3 model which makes the integration of the processor into the OBC model easier and faster.

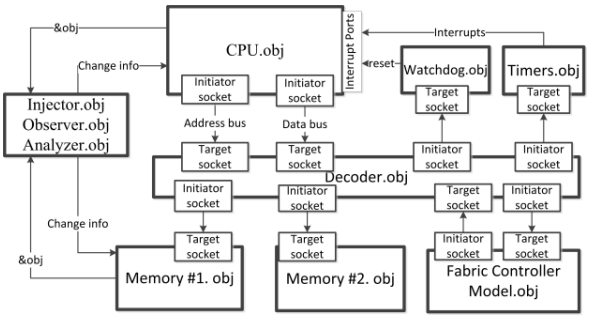

Figure 1. OBC model structure

The Watchdog and the Timer objects correspond to Watchdog and System Timers of the SmartFusion device [15]. The Watchdog object is connected to the reset port of the CPU and clocked at 100 MHz. When the Watchdog timeouts, the reset signal to CPU simulates the power-cycling (or reset) procedure. The Watchdog object also signalizes to volatile memories (the volatility is set during memory object creation) when it is to be erased because of the simulated power cycle. The Timer objects are connected to CPU interrupt ports and provide interrupt capabilities.

Although Fig.1 mainly represents the SmartFusion architecture, off-chip components (e.g. external SDRAM) can be also simulated by their connection to the Decoder. However, the communication delay of such component should be higher than on-chip components have.

B. FPGA fabric model incorporation

This work introduces the software-hardware co-design approach when the software executed by CPU and FPGA configuration described in RTL SystemC are simulated together. This approach provides an opportunity to verify and research the behavior of the system when both CPU and FPGA participate in OBC work. In order to introduce the FPGA fabric configuration to the OBC model, the intermediate FPGA wrapper has been created.

The Fabric Controller Model, represented in Fig.1 and Fig.2, is the wrapper of the synthesizable FPGA RTL SystemC configuration. The portability of the FPGA configuration to the real hardware is guaranteed by the Fabric Controller Model as it operates with the signals according to the protocol standards (AMBA AHB, FIFO, etc.).

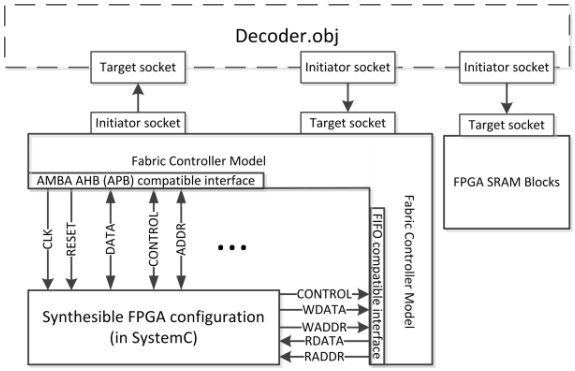

Figure 2. The FPGA fabric model as a part of the OBC model

The FPGA fabric in the SmartFusion device has a direct access to on-chip SRAM blocks through a FIFO controller that is modeled again by the Fabric Controller Model (see Fig.2).

The Flash-based FPGA fabric significantly increases the flexibility of OBC design and widens the design space of fault-mitigation techniques. One of the prospective approaches is to outsource the fault-mitigation functionality to the FPGA as it will be shown in Section IV. For instance, the FPGA can make copies of the data in radiation vulnerable SRAM/DRAM memories and continuously compare them. Thus, the FPGA can incorporate the memory scrubber functionality. Another example is the implementation of Error Detection and Correction (EDAC) techniques in the FPGA fabric, e.g. EDAC based on Hamming code.

III. Fault Injection Framework

A. Fault Injection Supporting Objects

The fault injection technique implemented in the OBC model mainly operates with the object pointers. When any object is created (e.g. CPU, memory storage), it sends the identification information and the object pointer to the Injector object (see Fig.1). Consequently, when the OBC model is initialized, the Injector is aware of the OBC's hardware configuration and it associates the fault models and fault occurrences with each component according to the predefined fault rates. Thus, it generates the fault list that is used during the simulation to inject faults at a predefined moment in time (controlled by the Analyzer object).

The faults inside the user memory of Flash-based FPGA fabric can be modeled by utilizing the Fabric Controller Model as a usual FIM.

The Observer object saves necessary tracking information about the simulation process (e.g. sequence of the executed instructions per time, transactions on the central bus).

B. System failure modes

The OBC fault tolerance estimation is based on the FMEA method [9]. The potential failure modes of the OBC can be found from the fault injection procedures during each simulation cycle. The system failure modes can be classified as:

1. *Incorrect data/Correct time (ID/CT)*: a benchmark execution is finished normally (on time) but the final execution results are wrong
2. *Correct data/Incorrect time (CD/IT)*: execution results are correct, but the process takes a longer time than originally allocated
3. *Incorrect data/Incorrect time (ID/IT)*: execution results are not correct, the simulated execution is not finished as expected by the program flow and was interrupted externally
4. *Processor Deadlock (PD)*
5. *Read Align Exception (RAE)*
6. *Write Align Exception (WAE)*
7. *Read Privilege Exception (RPE)*

PD, RAE, WAE, and RPE may belong to ID/IT or CD/IT, so these failure modes are researched separately to see the final picture more clearly.

IV. Case Studies

This section presents several case studies that show how the created OBC model can be used for software-hardware co-design and system vulnerability estimation.

A. Fibonachi sequence calculation

The calculation of Fibonacci algorithm is chosen as a benchmark algorithm. The computation of the algorithm is performed up to the 15^{th} element. Each element is calculated in a recursive way which guarantees that the calculation of the n^{th} and $(n+1)^{th}$ element are data independent and the calculation of each element takes longer than the preceding one. Each simulation is repeated for 300 times to obtain a statistical overview of possible influences of bit-flips induced in CPU registers (or Single Event Upsets - SEU). The simulation results show the dependence of the incorrect result ratio on the fault rate (see Fig. 3) as well as the dependence between the fault modes and the fault rate (see Fig. 4).

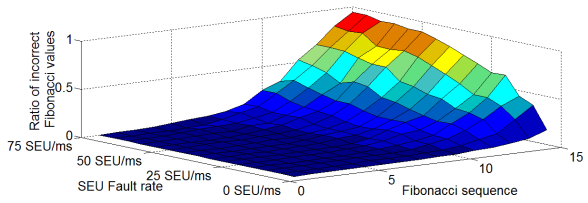

Figure 3. Influence of SEU rate in CPU on incorrect result ratio

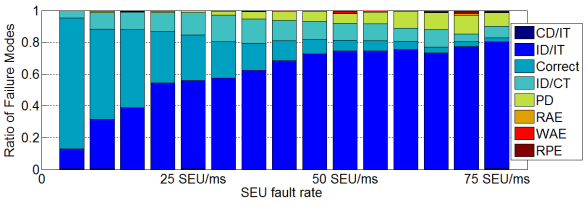

Figure 4. Influence of SEU rate in CPU on Failure Mode ratio

Both Fig. 4 and Fig. 5 show expected results: the incorrect results ratio grows with higher fault rate. The success of the calculation also depends on the execution time: the faster the calculation is performed, the lower the probability that the fault occurs during this short period of time. The percentage of ID/CT (or Silent Data Corruption [9]) stays approximately the same (see Fig.4). This observation means that for this particular program the ratio of faults that do not cause the CPU hanging, nor exceptions but do corrupt the final results stays approximately the same. This simulation result can be explained by the small number of General Purpose Registers (GPR) used for the execution of the algorithm and the increasing probability that the CPU will hang at higher fault rates, which increases PD ratio and masks ID/CT faults.

In the next experiment, the same Fibonacci sequence calculation is performed but after each member computation, the CPU resets the watchdog. Single Event Functional Interrupts (SEFI, or CPU hanging) are randomly introduced to the CPU during execution. Consequently, if the CPU freezes, the watchdog resets it which restarts the program.

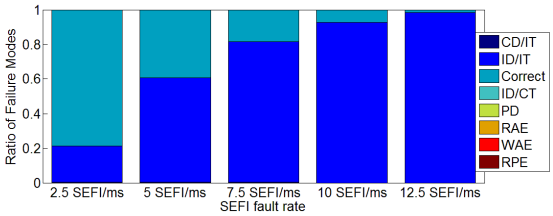

Fig.5 Influence of SEFI rate on Failure Modes

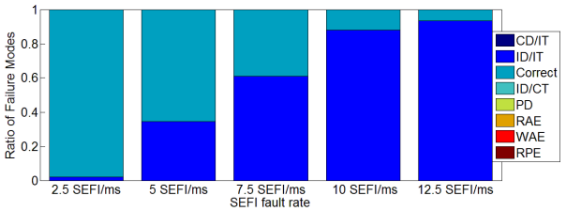

Fig.6 Influence of SEFI rate on Failure Modes with the tuned watchdog

Fig.5 presents the case where a watchdog allows a longer period than the calculation requires. As a result, CPU waits for an extended period of time as opposed to the case of Fig.6. Fig. 6 presents the case where the watchdog timeout period has been reduced by approximately four times, to 102.3 µs. This watchdog period is more tuned to the algorithm; hence in the case of SEFI the CPU is halted for a shorter period of time before the watchdog times out and resets the CPU. Consequently, the allocated time becomes sufficient to execute the whole algorithm at low fault rates; the allocated time equals tree execution time without the faults injection.

B. JPEG image compression

Typically, the data rate of Cubesat downlinks is very limited (lower 1 Mbit/sec). Therefore, the JPEG compression scheme plays an important role in on-board data processing.

Fig.7 shows a black-and-white source image with 2000 SEUs (several SEUs marked with orange). The JPEG compression output (30 times compressed) is presented in Fig. 8. The corruption of the output pictures is visible and marked with orange. While 2000 SEUs is an exaggerated value, they do allow the visualization of the effect, whereas 40 or 400 bit-flips did not show any visible change in the JPEG output.

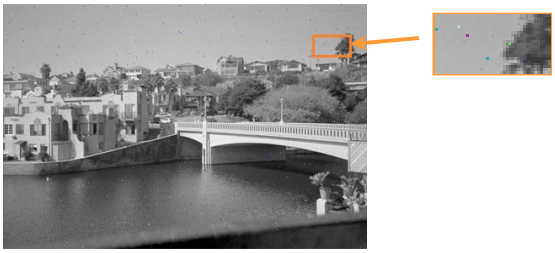

Fig. 7 Source image with 4000 SEUs

Fig. 8 JPEG output, the source is corrupted

To prevent corruption visible in Fig.8, the FPGA scrubbing technique mentioned in Section II.B is implemented. After the ARM core starts the JPEG compression execution, the scrubbing unit implemented in the FPGA, begins to copy the source picture to two backup versions. Since the SEU occurrence is randomly distributed in time, the copied pictures may have small or zero number of corrupted pixels. After copying, the FPGA scrubber continuously compares the source picture, with which the CPU is working, to the first backup copy. If the scrubber finds a difference, it compares it with the corresponding byte in the second backup copy and makes a decision about the most probable correct value of this byte.

As a result, the number of corrupted bytes in the source image has been reduced by 80-87% and there is no visual corruption in the output picture (see Fig.9). Byte-by-byte comparison of the output picture with the faultless compressed version reveals the difference in a majority (> 92%) of the file's bytes. However, this difference cannot be seen since the JPEG compression is making a local average of the pixel color values. This way SEUs are filtered out.

Fig.9 JPEG output, source is corrupted, FPGA scrubbing implemented

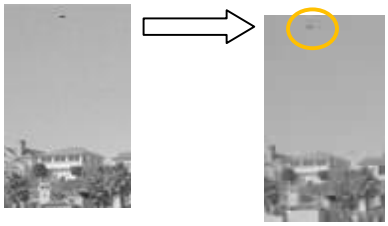

Fig. 10 JPEG compression with introduced 56-bit long multiple bit flip

Only when multiple bit flips are introduced, the corruption in the compressed picture becomes visible (see Fig. 10, no scrubbing implemented). Although, the probability of long bit-flips like this is significantly lower.

C. Kalman filter of Attitude Control Algorithm

A satellite's determination and attitude control algorithm provides knowledge and control over a satellite orientation in space. In the case of possible erroneous operation the orientation can be disturbed, which can have a detrimental effect on the satellite's mission. To understand what effects SRAM SEUs have on this type of algorithm, 1000 SEUs have been randomly introduced to the memory where the algorithm's variables are stored (sections *.bss* and *.data*). The injected errors are uniformly distributed in time and physical area. The correct output of Kalman filter is presented in Fig.11(three axes). The Kalman output with SEU introduction is presented in Fig.12.

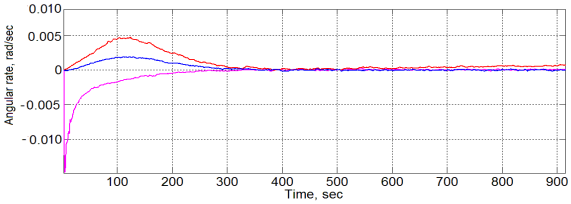

Fig.11 Correct Kalman filter output

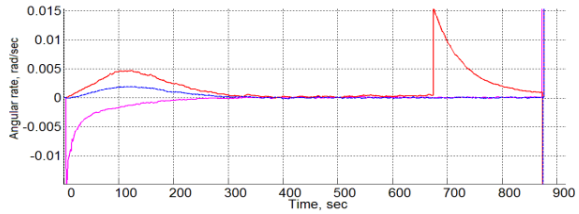

Fig.12 Kalman filter output with SEUs introduction

Comparing Fig. 11 and Fig.12, it is easy to see the effect of the injected errors. This can be easily detected by software. The detection of such behavior plays an important role since if Kalman losses stability (as it happened after 875 sec), it may not converges again until the external reset. We have implemented the software fault detection based on the detection of the output value jump bigger than 0.001 rad/sec. If such fault has been identified, the current output value is ignored and the Kalman filter resets. As a result of the implemented technique, the output values became very close to Fig. 11: the average difference between the correct filter output and the output with the software mitigation technique equals $1.7 \cdot 10^{-7}$ rad/sec for the period 0-900 seconds. This difference is acceptable and can be considered as negligible since it is four orders of magnitude less than the scale of the output values.

V. CONCLUSIONS

In this study, a novel methodology for fault-tolerance assessment of CubeSat OBCs is presented. It solved the problem of software-hardware co-design when the hardware may be not available. The methodology allows the early assessment of system vulnerability, which may help to improve the overall satellite reliability.

The required FPGA modeling approach is presented and the comprehensive OBC model is built. Consequently, the fault mitigation techniques that are implemented in FPGA or CPU can be verified and their

efficiency can be evaluated. The fault injection methodology is described and the case studies are given.

The case studies include the investigation of the system behavior under faults injection conditions. The chosen benchmark programs include Fibonacci sequence calculation, JPEG compression algorithm, and Kalman filtering for the satellite attitude control algorithm. The work shows how to reduce the number of accumulated memory bit-flips by 80-87% using a FPGA-based memory scrubbing technique. The paper presents a solution how to minimize the malfunction of a Kalman filter due to soft errors to guarantee the correct CubeSat orientation: the error was minimized to $1.7 \cdot 10^{-7}$ rad/sec on average on axis. The benefits of the presented methodology are also shown in the JPEG picture compression case study.

The exaggerated fault rates have been used to explicitly show the trends in the system behavior. Hence, the comparative analysis of radiation effects and the mitigation techniques is possible and was demonstrated in this paper.

The Actel SmartFusion platform was chosen for the realization of OBC implementation. This SoC includes a Cortex-M3 ARM core, FPGA fabric, and an Analogue Compute Engine, allowing the creation of a highly flexible design. The created model can be easily adapted for other OBC configurations.

REFERENCES

[1] C. Constantinescu, "Impact of Deep Submicron Technology on Dependability of VLSI Circuits", IEEE Intl. Conf. On Dependable Systems and Networks(DSN), pp. 205-209, 2002.

[2] P. Shivakumar, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic,"DSN, pp. 389-398,2002.

[3] T. Karnik, P. Hazucha, and J. Patel, "Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes," IEEE Trans. On Dependable and Secure Computing, Vol. 1, No. 2, pp. 128-143, April-June 2004.

[4] N. Kanekawa, Eishi H. Ibe, Ta kashi Suga, and Yu taka Uematsu,"Dependability in Electronic Systems. Mitigation of Hardware Failures, Soft Errors, and Electro-Magnetic Disturbances." ISBN 978-1-4419-6714-5. Springer, 2011.

[5] Andrew S. Keys and Michael D. Watson, "Radiation Hardened Electronics for Extreme Environments," Source of Acquisition NASA Marshall Space Flight Center.

[6] T. Grtker, S.Liao, G.Martin, and S.Swan, "System Design with SystemC," ISBN 978-1-4020-7072-3. Kluwer Academic Publishers, 2002.

[7] Frank Ghenassia, "Transaction Level Modeling with SystemC TLM Concepts and Applic-ations for Embedded Systems," ISBN 10 0-387-26232-6. Springer, 2005.

[8] J.C.Ruiz, P. Yuste, P. Gil, and L. Lemus, "On Benchmarking the Dependability of Automotive Engine Control Applications," International Conference on Dependable Systems and Networks, pages 857-866, June-July 2004.

[9] Y. Y. Chen, Chung-Hsien Hsu, and Kuen-Long Leu, "SoC-level Risk Assessment using FMEA Approach in System Design with SystemC," IEEE International Symposium on Industrial Embedded Systems, pp. 82-89, July 2009.

[10] Yung-Yuan Chen, Chung-Hsien Hsu, and Kuen-Long Leu, "Analysis of System Bus Transaction Vulnerability in SystemC TLM Design Platform," Proc. the 3rd WSEAS International Conference on Computer Engineering and Applications, pp.284-289, 2009.

[11] K.Rothbart, et al, "A Smart Card Test Environment using Multi-level Fault Injection in SystemC," 6th IEEE Latin-American Test Workshop, pp. 103-108, March-April 2005.

[12] S. Lee , A. Hutputanasin, A. Toorian, W. Lam, R. Munakata, "The CubeSat Program. CubeSat Design Specification," 12 edition. California Polytechnic State University. Available from:<http://www.cubesat.org/>.

[13] Microsemi Corporation. "Smartfusion Customizable System-on-Chip (CSoC)," Revision 7, August 2011.

[14] The official web-site of Open Virtual Platforms (OVP) project. <http://www.ovpworld.org/> .

[15] Actel Corporation, "Actel SmartFusion Microcontroller Subsystem Users Guide," 50200250-1 edition, pp.163-171, pp.302-304, May 2010.

A Simulator of On-Board Computers for Evaluating Fault-Mitigation Techniques

Burlyaev Dmitry[(1)], Rene van Leuken[(2)]

[(1)] Delft University of Technology (EEMCS/ME/CAS)/Innovative Solutions In Space B.V., Molengraaffsingel 12-14, 2629 JD, Delft, the Netherland, (+31) 634 767 736, d.burlyaev@isispace.nl

[(2)] Delft University of Technology (EEMCS/ME/CAS), Postbus 5, 2600 AA, Delft, the Netherland, t.g.r.m.vanleuken@tudelft.nl

ABSTRACT

This paper presents a SystemC-based methodology for CubeSat On-Board Computers (OBCs) modeling and simulation which allows the evaluation of the software and hardware fault mitigation techniques and the observation of radiation-induced fault consequences. The traditional analysis of the OBC fault-tolerance is limited due to the hardware unavailability at early development stages and low interpretability of radiation tests. Using the presented approach, the OBC model has been built and used for the comparative analysis of the mitigation techniques. The radiation environment for typical CubeSat missions are estimated, fault rates are calculated, and the corresponding fault models are created based on published empirical observations. The radiation-effects influence on the output of a CubeSat control algorithm was observed through a fault injection procedure into the OBC model. The methodology allows us to assess fault mitigation techniques as it will be shown in several examples, such as an attitude control and a JPEG compression algorithms. The work shows how the error of the attitude control algorithm can be minimized to $1.7 \bullet 10^{-7}$rad/sec and the number of accumulated memory bit-flips can be reduced by 80-87% during the image compression. The short simulation time allows the extensive statistical analysis for such modeling and simulation approach.

1 INTRODUCTION

Modern electronic systems are becoming more and more complex incorporating a large number of transistors whose sizes are shrinking from one generation to another. With the introduction of the deep submicron technology, reliability issues significantly increased on the component and system levels [1-3]. One of such issues is radiation-induced soft errors. To circumvent the effect of the soft errors, fault mitigation techniques are used in critical applications as satellites, avionics, medical and automotive devices.

An important question in the design of dependable systems is how to assess the efficiency of the applied fault mitigation techniques for a particular application. Testing for radiation-induced faults using radioactive isotopes does not guarantee the injection of specific errors, nor explain their nature or behavior [4]. Another problem is to estimate the influence of the radiation sensitivity of each component on the system level behavior taking into account that modern components incorporate sub-systems of different technologies. Both of these issues can be resolved by system and fault modeling. Such modeling provides the controlled fault-injection procedure and reveals the most vulnerable parts of the design. Due to the high complexity of each component and the system as a whole, the utilization of high level abstraction modeling language, such as SystemC, is imperative. The requirement on the hardware-software co-design and co-simulation is met with Transaction-Level Modeling (TLM) methodology [6].

This work has been done in the context of satellite applications, in particular for CubeSat [11] On-

Board Computers (OBCs). This category of applications is naturally exposed to the high level of radiation and, consequently, to diverse radiation-induced faults. The radiation environment assessment for CubeSat satellites is also presented in this paper.

The reminder of the paper is organized as follows. Section 2 observes the related works about the system modeling and fault-tolerance analysis. Section 3 presents the modeling approach of the OBC. Section 4 observes the ESA SPENVIS simulation results for the typical CubeSat radiation conditions. Section 5 shows how the fault injection technique has been integrated into the OBC system model. Section 6 demonstrates several case studies that show how mitigation techniques can be analyzed and hardware-software co-simulation implemented. Section 7 concludes the paper and summarizes the presented work.

2 RELATED WORK

A common existing SystemC approach to analyze the fault effects is based on the insertion of Fault Injection Module (FIM) into the interconnections of the functional blocks. FIM plays a role of the controller of a fault injection process that can be centralized [10] or distributed [8, 9]. However, such approach has been applied without consideration of hardware-software co-design, fault mitigation techniques implementation, and a comprehensive fault model library.

A general framework for reliability assessment in Multi-Processor SoC has been presented in the work [17]. The framework was created for the design space exploration and is not concentrated on radiation-related faults in COTS components, e.g. Single-Event Functional Interrupts (SEFI). The space-related works, like [18 and included], mainly discuss the RTL simulation with the assumption that the IC configuration of the used components are known. However, since CubeSats utilize COTS parts, all electronic components can be considered as black boxes.

When working with the limited knowledge about the internal components configuration, extensive fault-model library (based mainly on previous empirical observations) and statistical approaches are required to explore possible system behavior. The presentation of such approach is the main contribution of this paper.

3 SYSTEM MODEL

System-on-Chip (SoC) with CPU and FPGA is chosen as a core component of the OBC model to cover the most general OBC architecture. The SmartFusion SoC [12] is one of the most appropriate candidates for the CubeSat OBC that has to work in the space radiation environment at altitudes lower 750 km. It includes Flash-based FPGA that is more immune to soft errors [13] than SRAM-based FPGAs. The SmartFusion SoC incorporates the majority of on-chip subsystems that can be met in the advanced COTS electronics: a hard-core Cortex-M3, Flash-based FPGA fabric, Embedded Flash and SRAM memories, Analog subsystem, AMBA bus, etc.

3.1 OBC architecture modeling

The model structure of the OBC is presented in Fig.1; it incorporates the above-mentioned Cortex-M3 CPU, memories, the central bus (the Decoder plays the role of the central AMBA bus), FPGA fabric, a watchdog, and timers [14].

The existing instruction-accurate model of the processor Cortex-M3 has been obtained from the Open Virtual Platforms (OVP) project [13]. OVP provides a SystemC TLM wrapper for the Cortex-M3 model which makes the integration of the processor into the OBC model easier and faster.

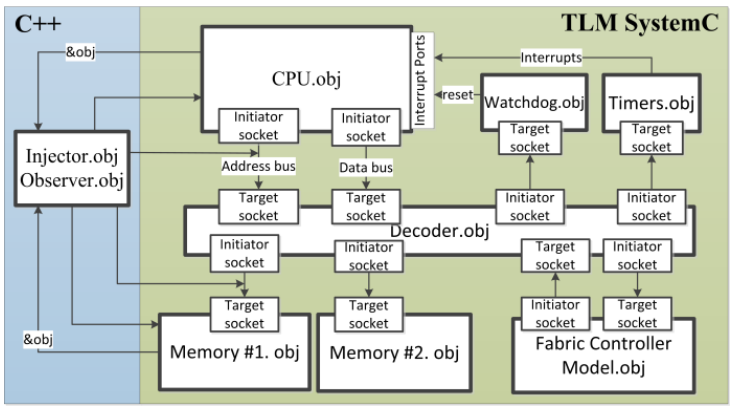

Figure 1. OBC model structure

The Watchdog object is connected to the CPU reset port and clocked at 100 MHz. When the Watchdog timeouts, the reset signal to CPU simulates the power-cycling (or reset) procedure. The Watchdog object also signalizes to volatile memory (the volatility is set during memory object creation) when it is to be erased because of the simulated power-cycle. The Timer objects are connected to CPU interrupt ports and provide interrupt capabilities.

Although Fig.1 mainly represents the SmartFusion architecture, off-chip components (e.g. external SDRAM) can be also simulated by their connection to the Decoder. However, the communication delay of such components should be higher than on-chip components have.

3.2 FPGA fabric model incorporation

We introduce the hardware-software co-design approach where the software, executed by CPU, and FPGA configuration, described in RTL SystemC, are simulated together. This approach provides an opportunity to verify and study the behavior of the system when both CPU and FPGA are used concurrently. An intermediate FPGA TLM wrapper has been created in order to introduce the FPGA fabric configuration to the OBC model.

The Fabric Controller Model, represented in Fig.1 and Fig.2 (red zone), is the TLM wrapper of the synthesizable RTL SystemC FPGA configuration (blue zone). The portability of the RTL FPGA configuration to the real hardware is guaranteed by the Fabric Controller Model as it operates with the signals according to the protocol standards (AMBA AHB, FIFO, etc.).

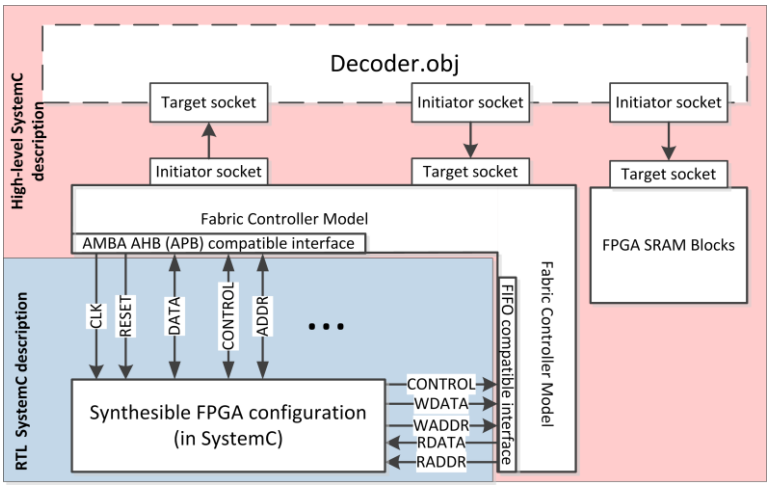

Figure 2. The FPGA fabric model as a part of the OBC model

The FPGA fabric in the SmartFusion device has a direct access to on-chip SRAM blocks through a FIFO controller. This communication is again modeled by the Fabric Controller Model (see Fig.2).

The Flash-based FPGA fabric significantly increases the flexibility of OBC design and widens the

design space of fault mitigation techniques (see Section V).

4 RADIATION ENVIRONMENT ASSESSMENT FOR CUBESATS

The radiation environment for CubeSat missions has been estimated using European Space Agency (ESA) SPENVIS system [15]. The estimated total mission dose with 1.5 mm aluminum shielding is never higher than 3.2 krad at 750 km regardless the inclination for the typical CubeSat operational lifetime (three years) (see Fig.3). This is also supported by the empirical APEXRAD model [16]. The particle spectra can be considered as equal to zero when the Linear Energy Transfer (LET) is higher than 32.5 MeV·cm^2/mg (see Fig.4).

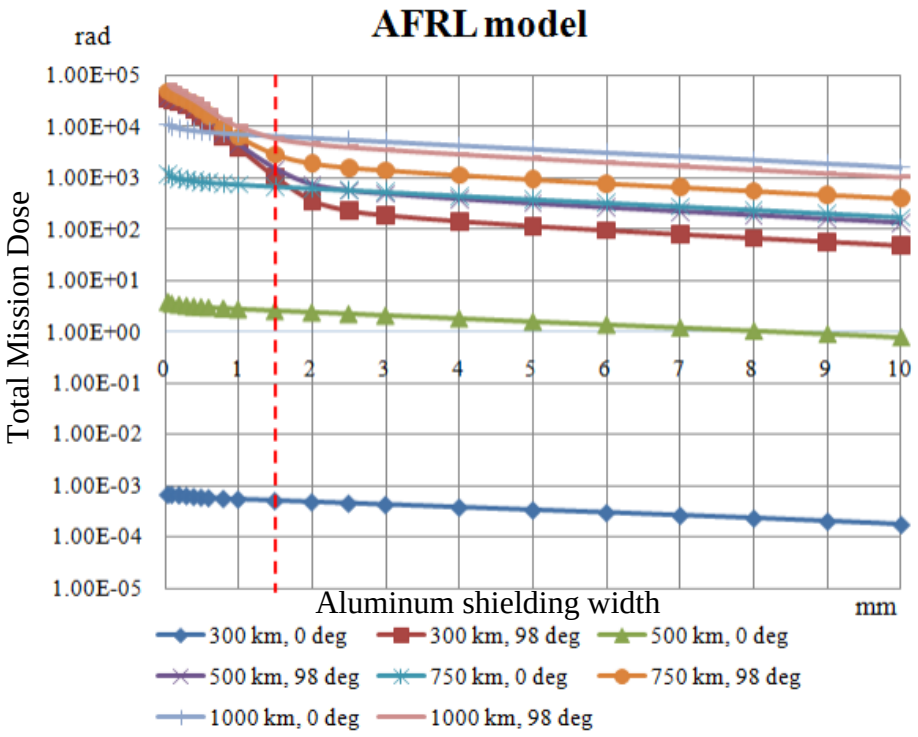

Figure 3. Simulation results: Total mission dose vs aluminium shielding width

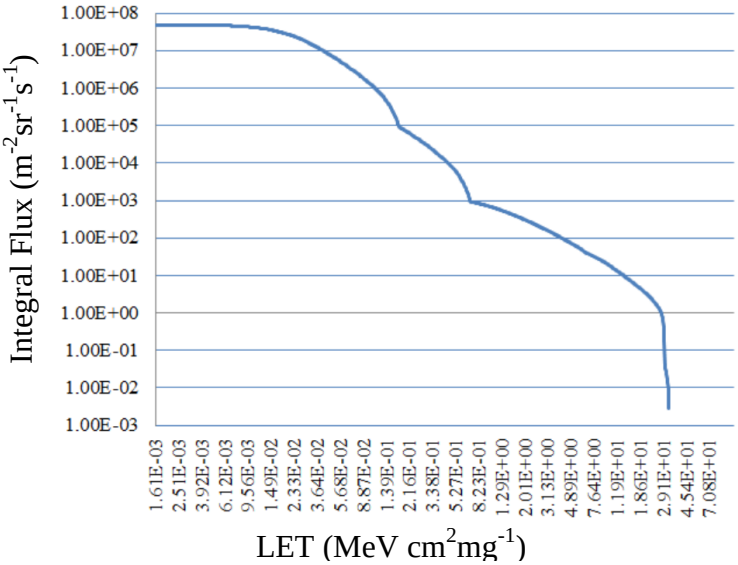

Figure 4. Simulation results: Spacecraft shielded LET(Si) spectrum

According to the investigation of published radiation test results (see Table I), such levels of Total Ionizing Dose (TID) and the width of LET spectrum cannot lead to destructive consequences in COTS components, like Single-Event Latchup (SEL) or Single Hard Error (SHE). Thus, the presented fault models are focused on soft errors and Single-Event Functional Interrupts (SEFI). The fault-rates (Table I) are calculated using the output LET(Si) spectrum from SPENVIS system and cross-sections from empirical observations according to the method described in [19].

TABLE I. Radiation sensitivity of COTS components, the worst case

Component type	**Malfunction at TID (krad)**	**SEL at LET ($MeV\text{-}cm^2/mg$)**	**SEU rate (upset/bit/day)**	**SEFI rate (event/device/day)**
DRAM [21]	> 20	51	$1.45 \cdot 10^{-8}$	0.26
SRAM [22]	20[20]	117	$1.27 \cdot 10^{-4}$	not observed
Flash NAND [23]	15 [24]	-	$1.38 \cdot 10^{-9}$	0.013
Flash NOR [23]	10-20	-	tolerant	0.0013

While the empirical observations show low fault rates, this work also includes the case studies with exaggerated fault rates to show the trend of the system behaviour.

5 FAULT INJECTION FRAMEWORK

5.1 Fault Injection Supporting Objects

The fault injection technique implemented in the OBC model mainly operates with object pointers. When any object is created (e.g. CPU, a memory storage), it sends the identification information and the object pointer to the Injector object (see Fig.1). Consequently, when the OBC model is initialized, the Injector associates the fault models and fault occurrences with each component according to the predefined fault rates and the fault model library.

The faults inside the user memory of Flash-based FPGA fabric can be modeled by utilizing the Fabric Controller Model as a usual FIM.

The Observer object saves the necessary tracking information about the simulation process (e.g. sequence of the executed instructions, transactions on the central bus).

5.2 System failure modes

The OBC fault tolerance analysis is based on the Failure Mode and Effect Analysis (FMEA) method [7-8]. The potential failure modes of the OBC can be found from the fault injection procedures during each simulation cycle. The system failure modes can be classified as:

1. *Incorrect data/Correct time* (ID/CT): a benchmark execution is finished normally (on time) but the final execution results are wrong.
2. *Correct data/Incorrect time* (CD/IT): execution results are correct, but the process takes a longer time than originally allocated.
3. *Incorrect data/Incorrect time* (ID/IT): execution results are not correct, the simulated execution is not finished as expected by the program flow and was interrupted externally.
4. *Processor Deadlock* (PD)
5. *Read Align Exception* (RAE)
6. *Write Align Exception* (WAE)
7. *Arithmetic Exception* (AE)

8. *Read, Write, or Fetch Privilege Exception* (RPE, WPE, or FPE consequently)

PD, RAE, WAE, RPE, WPE, FPE and AE may belong to ID/IT or CD/IT [28], but researched separately to see the final picture more clearly.

6 CASE SATUDIES

This section presents several case studies that show how the created OBC model can be used for hardware-software co-design and radiation effect analysis.

6.1 Fibonacci sequence calculation

The calculation of a Fibonacci sequence is chosen as a benchmark algorithm because it represents the family of recursive algorithms. The computation of the algorithm is performed up to the 15th element; each element is calculated in a recursive way. Each simulation is repeated for 300 times to obtain a statistical overview of possible influence of bit-flips induced in CPU registers (or Single Event Upsets - SEU). The simulation results show the dependence of the incorrect result ratio on the fault rate (see Fig. 5) as well as the dependence between the fault modes and the fault rate (see Fig. 6).

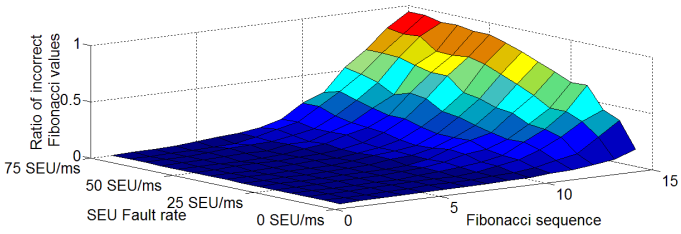

Figure 5. Influence of SEU rate in CPU on incorrect result ratio

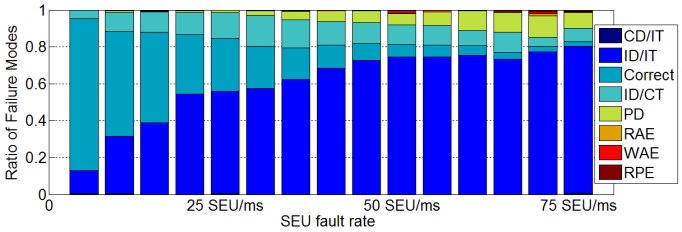

Figure 6. Influence of SEU rate in CPU on Failure Mode ratio

Both Fig. 5 and Fig. 6 show expected results: the incorrect results ratio grows with higher fault rate. The percentage of ID/CT (or Silent Data Corruption [8]) stays approximately the same (see Fig.6). This observation means that for this particular program the ratio of faults that do not cause the CPU deadlock, nor CPU exceptions but do corrupt the final results stays approximately the same. This simulation result can be explained by the small number of General Purpose Registers (GPR) used for the execution of the algorithm and the increasing probability that the CPU will freeze at higher fault rates, which increases PD ratio and masks ID/CT faults.

In the next experiment, the same Fibonacci sequence calculation is executed but the CPU resets the watchdog after each member computation. Single Event Functional Interrupts (SEFI, or CPU hanging) are randomly introduced into the CPU during execution. Consequently, if the CPU freezes, the watchdog resets it which restarts the program.

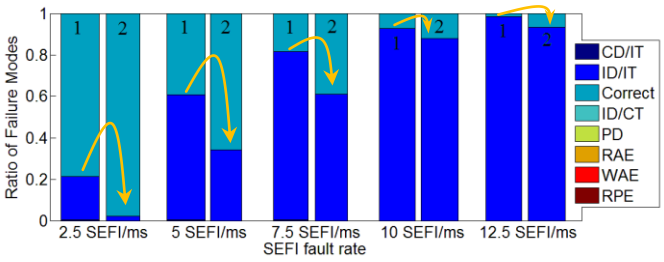

Figure 7. Influence of SEFI rate on Failure Modes with the unturned (marked with "1") and tuned watchdog (marked with "2")

Fig.7 presents the case (marked with "1") where the watchdog has a longer timeout period than the calculation requires. As a result, CPU waits for an extended period of time as opposed to the second case, marked with "2". In the 2nd case the watchdog timeout period has been reduced by approximately four times, to 102.3 µs. This watchdog period is more tuned to the algorithm; hence in the case of SEFI the CPU is halted for a shorter period of time before the watchdog times out and resets the CPU. Consequently, the allocated time becomes sufficient to execute the whole algorithm at low fault rates; the allocated time equals tree times the execution time without the faults injection.

6.2 JPEG image compression

Typically, the data rate of CubeSat downlink is very limited (lower 1 Mbit/sec). Therefore, the JPEG compression scheme plays an important role in on-board data processing.

Fig.8 shows a black-and-white source image with 2000 SEUs (several SEUs marked with orange). The JPEG compression output (30 times compressed) is presented in Fig. 9. The corruption of the output pictures is visible and marked with orange. While 2000 SEUs is an exaggerated value, they do allow the visualization of the effect, whereas 40 or 400 bit-flips might not show any visible change in the JPEG output.

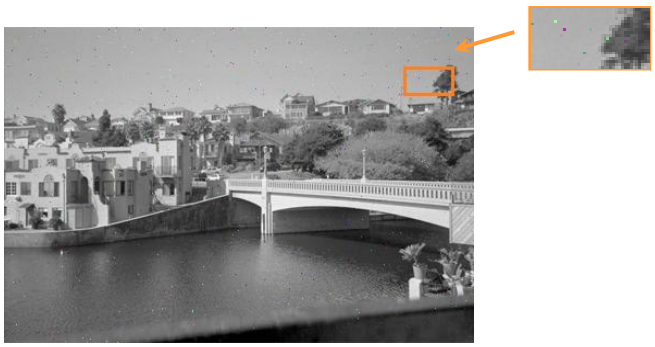

Figure 8. Source image with 2000 SEUs

Figure 9. JPEG output, the source is corrupted with 2000 SEUs

To prevent corruption that is visible in Fig.9, the FPGA scrubbing technique is implemented. After the CPU starts the JPEG compression execution, the scrubbing unit, implemented in the FPGA,

begins to copy the source picture to two backup versions. Since the SEU occurrence is randomly distributed in time, the copied pictures may have small or zero number of corrupted pixels. After copying, the FPGA scrubber continuously compares the source picture, with which the CPU is working, to the first backup copy. If the scrubber finds a difference, it compares it to the corresponding byte in the second backup copy and makes a decision about the most probable correct value of this byte.

As a result, the number of corrupted bytes in the source image has been reduced by 80-87% and there is no visual corruption in the output picture. Byte-by-byte comparison of the output picture with the faultless compressed version reveals the difference in a majority (more than 92%) of the file's bytes. However, this difference cannot be seen since the JPEG compression is making a local average of the pixel color values. This way SEUs are filtered out.

When multiple bit-flips are introduced, the corruption becomes visible in the compressed picture (see Fig. 10, no scrubbing implemented). But the probability of such long bit-flips is significantly lower.

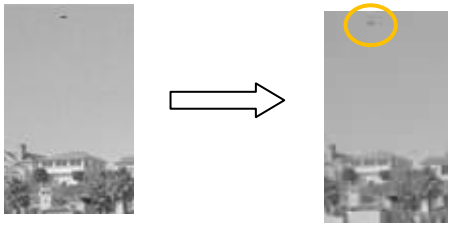

Figure 10. JPEG compression with introduced 56-bit long multiple bit-flip; the source and the output

The aforementioned memory scrubbing utilizes the Triple-Modular Redundancy(TMR) of the memory, which may be not supported by the limited OBC memory resources. Hence, more memory efficient approach should be used, e.g. memory scrubbing with Hamming encoding [25, 26]. While Hamming encoding requires memory only to save the syndromes, its implementation requires more FPGA logic due to a complicated control Finite State Machine (FSM). The encoding phase also requires an extensive number of multiplexers. RTL SystemC compilation to RTL has been conducted using CatapultC Synthesis[27]; the results are presented in Table II (frequency 50 MHz).

TABLE II. The output of CatapultC Synthesis for TMR and Hamming-based memory scrubbing FPGA co-processors

Scrubbing type	**Latency Cycles**	**Slack**	**Total Area**
TMR	1500	12.25	29723.13
Hamming	177081	7.55	181677.10

6.3 Kalman filter of Attitude Control Algorithm

A satellite's attitude determination and control (ADC) algorithm provides knowledge and control over its orientation in space. In the case of possible erroneous operation the orientation can be disturbed, which can have a detrimental effect on the satellite's mission. To understand what kind of effects SRAM SEUs have on this type of algorithm, 1000 SEUs have been randomly introduced to the memory where the algorithm's variables are stored (sections *.bss* and *.data*). The injected errors are uniformly distributed in time and physical area. The correct output of Kalman filter, which is the part of the ADC algorithm, is presented in Fig.11 (three axes). The Kalman output with SEUs injection is presented in Fig.12.

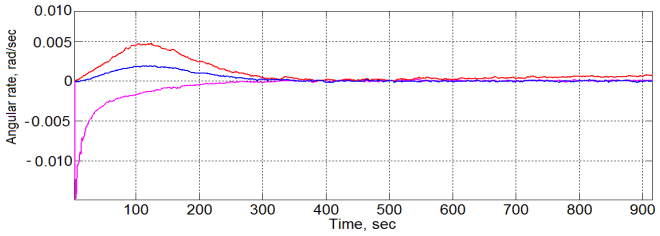

Figure 11. Correct Kalman filter output

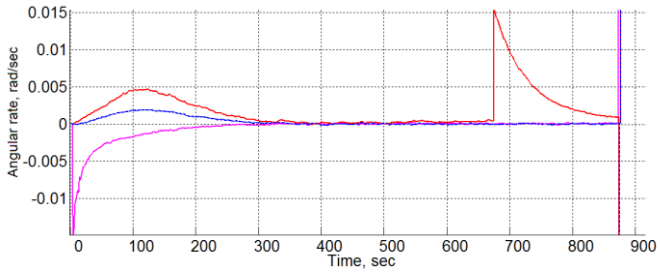

Figure 12. Kalman filter output with SEUs introduction

Comparing Fig. 11 and Fig. 12, it is easy to see the effect of the injected errors. This can be easily detected by software. The detection of such behavior plays an important role since, if Kalman filter losses stability (as it happened after 875 sec), it may not converge again until the external reset. We have implemented the software fault detection based on the detection of the output value jump bigger than 0.001 rad/sec. If such fault has been identified, the current output value is ignored and the Kalman filter resets. As a result of the implemented technique, the output values became very close to Fig. 11: the average difference between the correct filter output and the output with the software mitigation technique equals $1.7 \cdot 10^{-7}$ rad/sec for the period 0-900 seconds. This difference is acceptable and can be considered as negligible since it is four orders of magnitude less than the scale of the output values.

6.4 Simulation time of the case studies

This section provides the information about the simulation time of the presented case studies (see Table III). All simulations have been conducted on a PC with an Intel Core i5-2430M and 4 GB of DDR3 memory. As was expected, SystemC TLM modeling provides fast simulation capabilities even for complex cases as the software JPEG image compression with the FPGA-based memory scrubbing.

TABLE III. The Simulation Time of the Case Studies

Case study (one iteration)	**Number of Instructions**	**User Time (sec)**	**System Time (sec)**	**Elapsed Time (sec)**
Fibonacci sequence calculation (15 elements)	28,912	0.25	0.01	0.26
Fibonacci sequence calculation with watchdog	28,940	0.26	0.01	0.27
JPEG image compression (432x288 pixels)	22,079,255	185.93	0.09	186.89
JPEG image compression with memory scrubbing	22,079,255	225.90	0.10	226.99
Kalman filtering	120,504	1.56	0.04	1.67

7 CONCLUSIONS

This paper proposes the simulation approach for fault-tolerant analysis of the CubeSat OBC based on Commercial Off-The-Shelf (COTS) components. The presented method allows to understand the influence of each software and hardware component on the OBC fault tolerance at early development stages. The paper presents the system and fault modeling approaches that in conjunction indentifies the satellite sub-system behavior in radiation environment.

The worst-case radiation conditions for CubeSat mission at 750 km altitude has been assessed and the correspondent fault-models of possible radiation effects are created. This faults have been introduced to the CubeSat OBC model according by the dedicated fault-injection mechanism to assess the radiation effect influence on final computation results of benchmark programs.

The chosen benchmark programs include Fibonacci sequence calculation, JPEG compression algorithm, and Kalman filtering for the satellite attitude control algorithm. The work shows how to reduce the number of accumulated memory bit-flips by 80-87% using a FPGA-based memory protection technique. The paper presents a solution how to minimize the malfunction of a Kalman filter due to soft errors to guarantee the correct CubeSat orientation: the error was minimized to $1.7 \bullet 10^{-7}$ rad/sec on average on axis. The benefits of the presented methodology are also shown in the JPEG picture compression case study.

Using the presented modeling and simulation methodology for the fault-tolerance assessment of an OBC, well-known fault mitigation techniques can be investigated and improved for different applications. This allows for the early assessment of system vulnerability, improving the overall satellite reliability. The presented methodology can be easily adapted for other OBC configurations.

8 REFERENCES

[1] C. Constantinescu, *Impact of Deep Submicron Technology on Dependability of VLSI Circuits*, IEEE Intl. Conf. On Dependable Systems and Networks (DSN), pp. 205-209, 2002.

[2] P. Shivakumar, *Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic*, DSN, pp. 389-398, 2002.

[3] T. Karnik, P. Hazucha, and J. Patel, *Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes*, IEEE Trans. On Dependable and Secure Computing, Vol. 1, No. 2, pp. 128-143, April-June 2004.

[4] Andrew S. Keys and Michael D. Watson, *Radiation Hardened Electronics for Extreme Environments*, NASA Marshall Space Flight Center, December 15, 2007.

[5] T. Grtker, S.Liao, G.Martin, and S.Swan, *System Design with SystemC*, ISBN 978-1-4020-7072-3. Kluwer Academic Publishers, 2002.

[6] Frank Ghenassia, *Transaction Level Modeling with SystemC TLM Concepts and Applications for Embedded Systems*, ISBN 10 0-387-26232-6. Springer, 2005.

[7] J.-C.Ruiz, P. Yuste, P. Gil, and L. Lemus, *On Benchmarking the Dependability of Automotive Engine Control Applications*, International Conference on Dependable Systems and Networks, pages 857-866, June-July 2004.

[8] Y. Y. Chen, Chung-Hsien Hsu, and Kuen-Long Leu, *SoC-level Risk Assessment using FMEA Approach in System Design with SystemC*, IEEE International Symposium on Industrial Embedded Systems, pp. 82-89, July 2009.

[9] Y. Y. Chen, Chung-Hsien Hsu, and Kuen-Long Leu, *Analysis of System Bus Transaction Vulnerability in SystemC TLM Design Platform*, Proc. the 3rd WSEAS International Conference on Computer Engineering and Applications, pp.284-289, 2009.

[10] K.Rothbart, et al, *A Smart Card Test Environment using Multi-level Fault Injection in SystemC*, 6th IEEE Latin-American Test Workshop, pp. 103-108, March-April 2005.

[11] S. Lee , A. Hutputanasin, A. Toorian, W. Lam, R. Munakata, *The CubeSat Program. CubeSat Design Specification*, 12 edition. California Polytechnic State University. Available from: <http://www.cubesat.org/>.
[12] Microsemi Corporation. *SmartFusion Customizable System-on-Chip (CSoC)*, Revision 7, August 2011.
[13] The official web-site of Open Virtual Platforms project. <http://www.ovpworld.org/>.
[14] Actel Corporation, *Actel SmartFusion Microcontroller Subsystem Users Guide*, 50200250-1 edition, pp.163-171, pp.302-304, May 2010.
[15] The offcial web-site of SPENVIS system: <http://www.spenvis.oma.be/>.
[16] The official web-site of APEXRAD model. <https://creme.isde.vanderbilt.edu/CREME-MC/help/apexrad>.
[17] G. Beltrame, C. Bolchini, L. Fossati, A. Miele, D. Sciuto, *A Framework for Reliability Assessment and Enhancement in Multi-Processor Systems-On-Chip*, 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, pp. 132 - 142, 2007.
[18] O. Ruano, J.A. Maestro, P.Reyes, and P. Reviriego, *A Simulation Platform for the Study of Soft Errors on Signal Processing Circuits through Software Fault Injection*, IEEE International Symposium on Industrial Electronics, pp. 3316-3321, 2007.
[19] G.I. Zebrev, I.O. Ishutin, R.G.Useinov, and V.S.Anashin, *Methodology of soft error rate computation in modern microelectronics*, IEEE Transactions On Nuclear Science, 57(6): 3725-3733,December 2010.
[20] R. Koga, P. Yu, K. Crawford, J. George, and M. Zakrzewski, *Synergistic Effects of Total Ionizing Dose on SEU Sensitive SRAMs*, IEEE Radiation Effects Data Workshop, pp. 127 - 132,2009.
[21] R.K. Lawrence, *Radiation characterization of 512Mb SDRAM*, IEEE Radiation Effects Data Workshop, July, 2007.
[22] E.H. Cannon, et al., *Heavy ion, high-energy, and low-energy proton SEE sensitivity of 90-nm RHBF SRAMs*, IEEE Transactions on Nuclear Science, 57(6)L3493-3499, December 2010.
[23] Farokh Irom and Duc N. Nguyen, *Single Event Effect Characterization of High Density Commercial NAND and NOR Nonvolatile Flash memories*, IEEE Transactions on Nuclear Science, pp. 2547-2553, December, 2007.
[24] Kurt Anderson, *Low-cost, radiation-tolerant, on-board processing solution*, IEEE Aerospace Conference, pp. 1-8, March, 2005.
[25] Actel Corporation, *Using EDAC RAM for RadTolerant RTAX-S FPGAs and Axcelerator FPGAs*, Application Note AC273,July 2006.
[26] Actel Corporation, *CoreEDAC Handbook*, January 2006.
[27] The official web-site of CatapultC Synthesis tool from Mentor Graphics company:
< http://www.mentor.com/esl/catapult/overview>.
[28] Imperas Software Limited, *OVPsim and Imperas CpuManager User Guide*, Version 2.0.43, November 2011, p. 29-30.

System Fault-tolerance Analysis of Small Satellite On-board Computers

Abstract—**This paper proposes the statistical simulation approach for fault-tolerance analysis and software-hardware co-design of Small Satellite On-board Computers (OBCs) based on Commercial Off-The-Shelf (COTS) components. The presented method allows to understand the influence of each software and hardware component on the system fault tolerance at early development stages and when the IC-level of COTS parts are unknown. The proposed approach allows the comparison between the efficiency of fault-mitigation techniques as it is presented in the paper. As a case study, the work presents the model of SmartFusion System-On-Chip (SoC) and the fault-tolerance analysis of a satellite attitude determination and control algorithm. The FPGA-based memory protection with Hamming encoding is implemented, assessed, and optimized basing on the simulation results. The comparison between FPGA-based memory protection and a software mitigation technique is conducted. The importance of CPU exception handling for satellite OBCs is explained: the proper exception handling may cover up to 11% of wrong computation results. A presented optimization method based on a clustering algorithm helped to reduce the scrubbing turnaround period and allocated memory resources almost by half.**

Keywords- SystemC; fault-tolerance analysis; on-board computer; Single-Event Effects; fault-mitigation techniques

I. INTRODUCTION

The main requirement for small satellite OBCs is their tolerance towards faults caused by radiation. While the complexity of each component and whole systems are growing, the existing analytical methods for fault-tolerance analysis become infeasible for complex heterogeneous systems and for COTS components whose IC-level is unknown for satellite designers. Since COTS-based small satellites are sensitive to radiation due to low radiation-tolerance of COTS components, the satellite designers use well-known fault-mitigation techniques. The benefits and correctness of implemented fault-mitigation techniques can be assessed under the radiation tests at the final development stage. However, if the radiation test is failed, the expensive and time-consuming re-design is required. The comparative analysis of the design space and system-level debugging for fault-tolerance are impossible due to high cost of radiation tests. Moreover, testing for radiation-induced faults using radioactive isotopes does not guarantee the injection of specific errors, nor explain their nature or the system behavior [10].

An alternative approach for the system-level fault-tolerance analysis is through simulation, which is examined in this work.

The rest of the paper is organized as follows: Section II observes the related works. Section III presents the proposed SystemC-based simulation approach with the explanation of the implemented system modeling (Section III.a) and radiation fault modeling (Section III.b). Section IV gives an example of the approach use for SmartFusion SoC [11] that executes the parts of satellite attitude determination and control algorithm.

II. RELATED WORKS

The simulation approach for fault-tolerance analysis has already been used for satellite sub-systems in works [1, 2]. These works investigate the fault tolerance through the fault injection into the system model. However, the works are based on the assumption that we know the circuit level of the components we use, which is not the case when COTS parts are used. Moreover, the low-level simulation is time consuming; hence, it cannot be used for statistical analysis.

Due to the high complexity of each component and the system as a whole, the utilization of high level abstraction modeling language, such as SystemC, is imperative. A common existing SystemC approach to analyze fault effects is based on the insertion of Fault Injection Module (FIM) into the interconnections of the functional blocks. FIM plays a role of the controller of a fault injection process that can be centralized [4] or distributed [5, 6]. However, such approach has been applied without consideration of hardware-software co-design, fault mitigation techniques implementation, and a comprehensive fault model library that are covered in this work.

This work proposes the innovative method to analyze the fault-tolerance of COTS-based satellite on-board computers and compare hardware or/and software mitigation techniques' efficiency at early development stages.

III. PROPOSED METHOD

The proposed method is based on two main components: the system model of an electronic device/computer and the library of fault models. The supporting modules such as a dedicated fault-injection mechanism (hereafter 'the Injector') and a dedicated data tracking mechanism (hereafter 'the Observer') are also required (Fig. 1). The Injector controls the fault injection procedure according to the created schedule and the created fault-models; the Observer provides information about the system state (memory content, bus transactions, etc.) at different moment of simulation for a user.

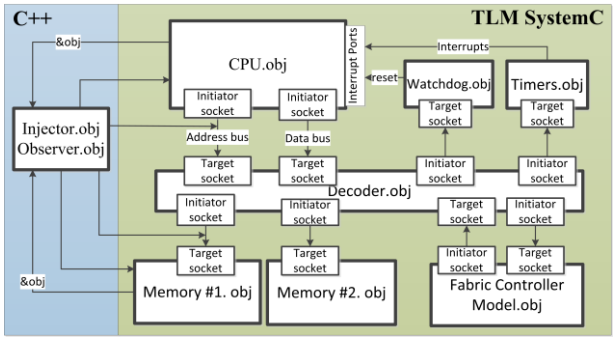

Figure 1. System model structure

A. *System Modeling*

The requirement on the hardware-software co-design and co-simulation is met with Transaction-Level Modeling (TLM) methodology [3] (Fig.1, the green section). The TLM approach significantly simplifies the system model and shortens the simulation time. TLM is used for interconnection (with target and initiator sockets) between functional blocks (CPU, FPGA, Memory blocks, Timers, Decoder as a central bus, etc.) that can be considered as "black boxes" since their full configuration in COTS parts are unknown for satellite designers. However, the standardized interconnections should be simulated not with high-level TLM approach but with Register-Transfer-Level (RTL) in some cases, e.g. the interconnection between FPGA fabric and the central bus (see Fig. 2).

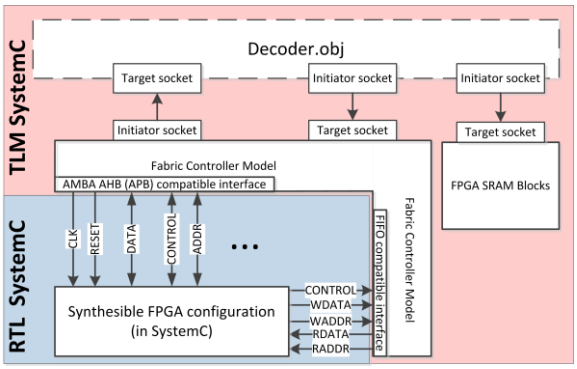

Figure 2. The FPGA fabric model as a part of the system model

Fig. 2 shows how the interconnection between the central SoC bus (e.g. AMBA bus) and the FPGA co-processor unit can be created using an intermediate level to support the portability of the co-processor unit from the simulation environment to a real device. The Fabric Controller Model is an intermediate level that communicates with the co-processor according to the particular protocol using signals (e.g. AMBA protocol) but connected to the Decoder through a TLM channel. The co-processor configuration (Fig.2, blue section) is described in RTL SystemC.

Consequently, the presented system model supports software-hardware co-design and applicable for system performance analysis.

B. *Fault-Modeling and Tolerance Analysis*

The proposed fault injection technique mainly operates with object pointers. When any object is created (e.g. CPU, Memory blocks), it sends the identification information and the object pointer to the Injector (see Fig.1). Consequently, when the system model is initialized, the Injector associates the fault models and components where particular faults can happen. The Injector creates a fault-injection schedule (see Fig.3) according to the predefined fault rates or uses the schedule written by a user.

```
36346:  NS: SEFI:   CPU:    cpu1:              (time:err_type:Module_name)
52232:  NS: SEFI:   CPU:    cpu1:              (time:err_type:Module_name)
163147: NS: SEU:    CPU:    cpu1:   9:      14: (time:err_type:Module_name:register:bit)
494789: NS: SEU:    MEM:    eSRAM:  3948:   4:  (time:err_type:Module_name:address:bit)
```

Figure 3. The example of the fault schedule created automatically or defined by a user

Since the internal structure of COTS parts are unknown, the effects of different faults can be simulated by changing the outputs or/and the state of simulated functional blocks. The change of block outputs can be done at TLM initiator sockets (Fig.1, 2). The change of the block state depends on the knowledge about the block, e.g. it corresponds to the register change in the CPU block and the memory array change in the Memory block. Thus, the fault-models are high-level as well.

The accuracy of fault-tolerance analysis in the proposed method depends on how well the high-level fault models correspond to the real system behavior. For small satellites at typical orbits (lower 750 km), the next fault models are valid based on the published empirical observations [12-15] :

1) Single-Event Upset (SEU) - or bit-flip in a memory cell (CPU registers, memory arrays, block outputs, etc.)

2) Multiple-Cell Upset (MCU) - or multiple SEUs in adjacent memory cells (CPU registers, memory arrays, block outputs, etc.)

3) Single-Event Functional Interrupt (SEFI)- the functional block/electronic component is put into an unknown state or frozen. SEFI corresponds to the radiation-induced fault in the control circiut of a component

According to the published observations [16], SEFI in CPU can be modeled by the CPU freezing, and SEFI in memory can be simulated by the memory block that stopped responding. The common, if not only, SEFI mitigation techniques are the component reset or power-cycle. The knowledge about possible SEFI consequences is limited since the IC-level of control logic is unknown; thus, it is necessary to be sure that at least power-cycling will work in the majority cases of system behavior. According to the observations [16], the component reset cannot assure SEFI mitigation as opposed to power-cycling.

Single-Event Transient (SET) faults are also expected in COTS components in space; however, unless SET is latched in a memory cell, it is not causing any changes in the OBC's working process. If SET is latched, it may change the memory cell content. The memory content change corresponds also to SEU/MCU fault models and SEFI fault models if the changed memory cell belongs to control logic; thus, additional SET fault models are not required.

Both SEU and MCU can be modeled by changing the content of memory cells (e.g. in registers, memory blocks) or TLM transaction objects (e.g. of *tlm_generic_payload* type). TLM transaction objects also correspond to the data saved in memory cells in real hardware. Hereafter, we propose the generalized simulation approach for system fault-tolerance analysis in the case of single memory-based faults (SEU and MCU).

C. *Multidimentional analysis of memory fault consequences*

Building the OBC model, it is possible to indentify used memory locations (e.g. memory ranges, list of registers). For some cases, e.g. FPGA co-processors before mapping, we cannot identify used memory resources. In such cases the fault models can be implemented only in the modeled

communication channels of these functional blocks (particularly corrupting the content of TLM transaction objects). For instance, faults inside the user memory of Flash-based and Anti-fuse FPGA fabric can be modeled by utilizing the Fabric Controller Model as a usual FIM. Flash-based and Anti-fuse FPGAs are more immune to soft errors [17] than SRAM-based FPGA type, that is not considered in this work.

So, we can identify memory resources used for different purposes. Collecting the corresponding memory locations, we can form the memory address axis(see Fig.4). The second axis corresponds to the time of the memory fault introduction. Other

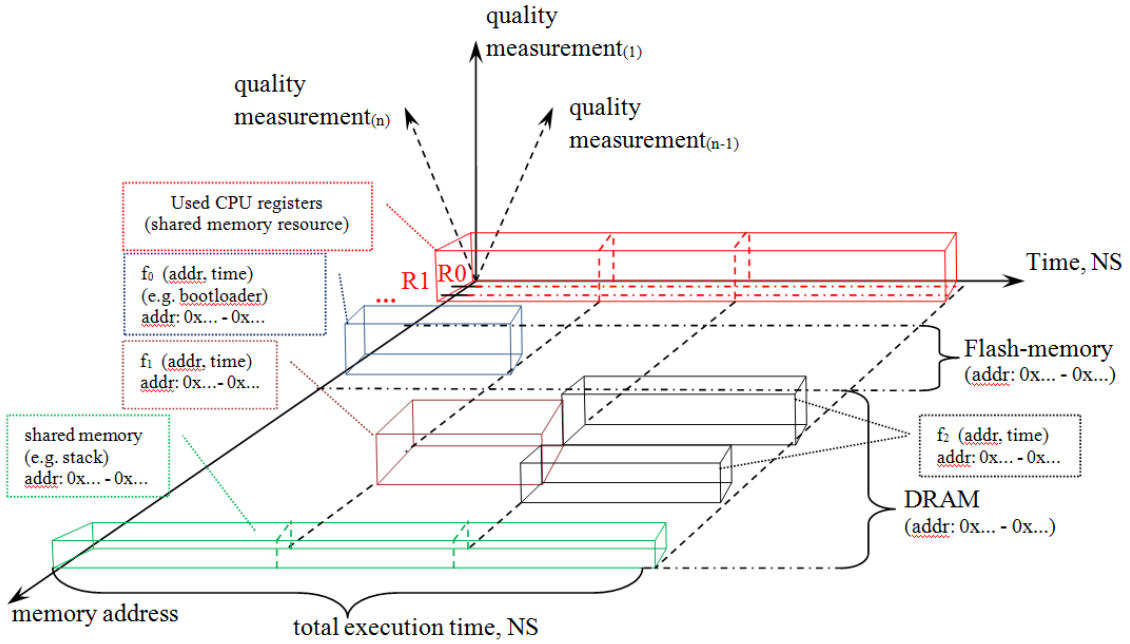

Figure 4. The representation of multidimensional analysis

dimensions (the third and higher) correspond to the computational quality measurements (introduced by OBC designers) that represent the correctness of the computation output. For different programs the quality measurements can be different: a total execution time, an output value deviation from the correct result, the final computation fault mode [18-19], etc. The introduction of one or several quality measurements allow the comparison between fault-tolerance of different OBC implementations.

For small satellites at typical orbits less than 750 km, the fault rates are negligible in comparison with CPU execution speed. The maximum estimated fault rate is not more than one fault/sec for modern COTS components. Consequently, the one fault injection per algorithm execution is a justified approach.

The whole OBC software can be divided into sub-programs that are executed sequentially by a CPU. If there is a program flow dependency between the execution of the previous sub-program f_1 and the next sub-program f_2 (e.g. the shared data in stack, see Fig. 4), the influence of a fault that happened during the execution of f_1 on the output of f_2 sub-program can be investigated by fault introduction to the commonly used memory locations just before f_2 starts execution. Thus, the fault injection into used memory ranges by different sub-programs can be limited to the injection when the sub-program (f_0, f_1, or f_2) is being executed.

IV. SmartFusion SoC: Case Study

As a case study, the model of the SmartFusion device [11] has been created and the proposed fault-tolerance analysis is applied. The existing instruction-accurate model of the processor Cortex-M3 from the Open Virtual Platforms (OVP) project [9] is used as the CPU block. Each simulation iteration one SEU is introduced to the memory region where the program of an adaptive filter is saved (Table I). The single SEU per iteration is motivated by low possible fault-rates on altitudes less than 750 km in comparison with the short execution time of the algorithm.

A. *Code execution without fault mitigation techniques*

The CPU executes the code of an adaptive filter used in a satellite attitude determination and control algorithm for filtering the measurements of solar sensors and magnetometers (Table II). The adaptive filter during one simulation iteration calculates 30 results for 3 axis, 90 double values in total. The chosen quality measurement is the average relative deviation of 90 values calculated with SEU introduction from 90 correctly calculated filter outputs. The result of the simulation is presented in Fig.5.

TABLE I. The Memory Sections where SEU are injected

Name	Size	VMA=LMA
.text	0x000101c	0x0000000
.rodata	0x0000044	0x000101c
.data	0x0000178	0x0001060
.bss	0x0000020	0x00011d8

TABLE II. SIMULATION TIME FOR THE VERSION WITHOUT PROTECTION- ONE ITERATION

CPU type:	armm (Cortex-M3)
Nominal MIPS:	100
Simulated instructions:	105,515
User time:	0.79 seconds
Elapsed time:	0.83 seconds

memory calculating syndromes again and comparing them with the saved ones that are supposed to be correct. However, the syndromes are also can be damaged by SEU; so the backup version is used when the syndromes difference is found to clarify if the syndrome or the protected region is damaged.

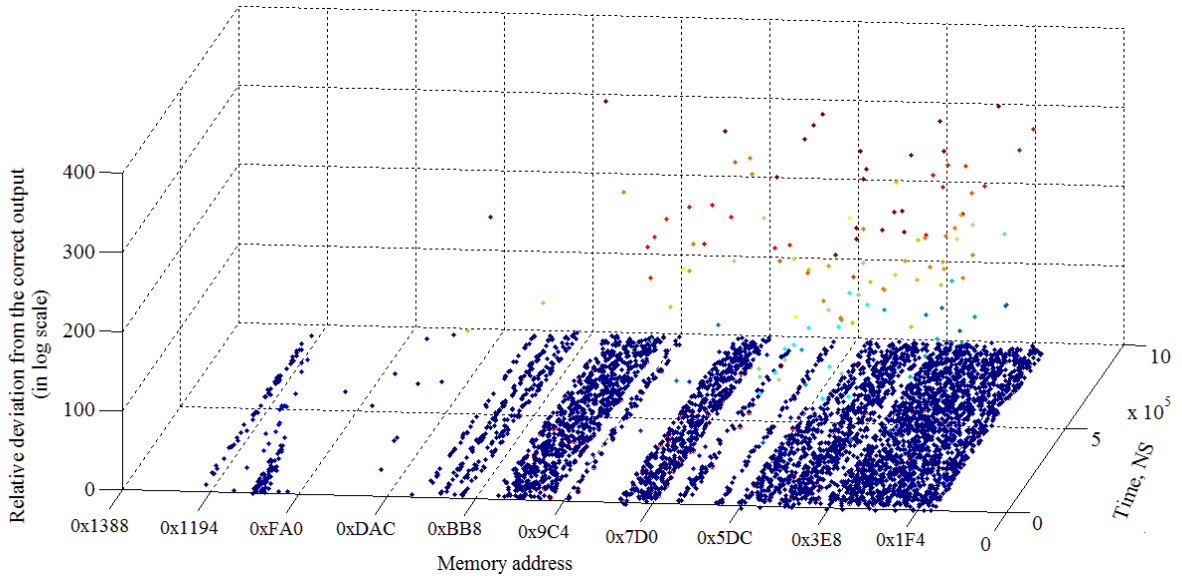

Figure 5. The simulation result of the adaptive filtering computation with one SEU introduction (20 000 iterations)

Fig.5 shows that during the SEU introduction to particular parts of software (particular memory addresses) the calculated results differ from the correct ones significantly (these cases are marked with colored dots). At the same time, SEU introduction to other regions, e.g. the very beginning of the memory (address 0x0-0x19a), does not influence the calculation output. It can be explained by the interrupt vectors that are located at the beginning of the memory and not directly used during this simulation and during the real computation. The computation in main function starts particularly at address 0x19a where we can observe first error results(represented by colored dots).

B. Code execution with fault-mitigation

Since SmartFusion SoC includes FPGA fabric, some fault-mitigation techniques can be implemented in FPGA. Memory scrubbing based on Hamming encoding has been chosen as an example (Table III). Since *.text* and *.rodata* section of the program are static, they can be protected with the memory scrubbing technique.

Before the algorithm execution the CPU sends to FPGA co-processor the number and locations of memory ranges that should be protected by FPGA scrubbing. FPGA co-processor encodes the data saving the syndromes and one backup copy of the protected memory regions. After the syndromes generation, the FPGA co-processor sends to the CPU a message indicating that the CPU can continue algorithm execution. Since the syndromes are known, FPGA starts scrubbing the protected

TABLE III. SIMULATION TIME FOR THE VERSION WITH FPGA-BASED PROTECTION- ONE ITERATION

Nominal MIPS:	100
Simulated instructions:	199,195
User time:	2.36 seconds
Elapsed time:	2.47 seconds

Another used fault-mitigation technique is a software-based value limitation: the difference between consecutively calculated outputs are limited (Table IV). The limitation is based on the expected parameters range and the speed of their changes . In the case study, the limit is E-4 when the filter output values lie in the scale of E-5 and E-6. If the limit is not met, the filter recalculates the values.

TABLE IV. SIMULATION TIME FOR THE VERSION WITH SOFTWARE PROTECTION- ONE ITERATION

Nominal MIPS:	100
Simulated instructions:	123, 442
User time:	1.26 seconds
Elapsed time:	1.28 seconds

The simulation results are shown in Fig.6 and Table V.

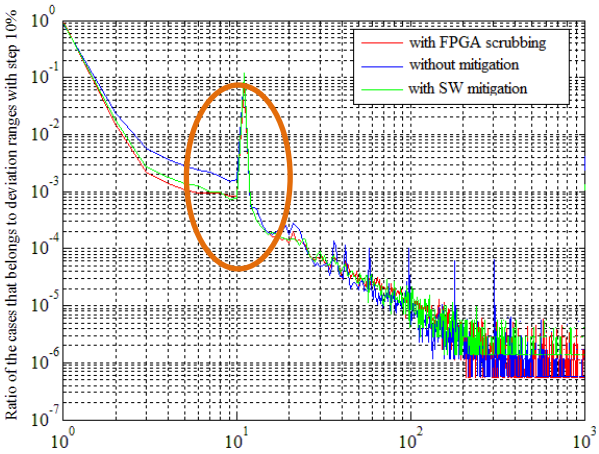

Figure 6. Histogram of system fault-tolerance with and without fault-mitigation techniques (20 000 iterations)

The jump when the relative deviation histogram equal 11 (marked with orange oval) can be explained by the next observation. In many cases after the fault injection, the CPU rises the exception. If the exception routine is not written (as in the presented case), the algorithm execution stops and the memory region, where the filter output values should be written to, stays equal zero. When the output value is zero the relative deviation from the correct result equals one. The peak is located at 11 relative incorrectness since 11 corresponds to [100%, 110%) deviation from the correct result.

TABLE V. SIMULATION RESUTLS, SYSTEM FAULT-TOLERANCE WITH AND WITHOUT FAULT-MITIGATION TECHNIQUES

Relative deviation of the out values from the correct resutl	Ratio of iterations with such deviation (20 000 iterations in total)		
	without protection	SW limitation	FPGA-based Hamming encoding
[0-10)%	82.81%	84.58%	88.60%
[10-20)%	2.36%	1.76%	1.45%
[20-30)%	0.57%	0.27%	0.21%
[100-110)%	11.31%	11.75%	7.93%

Table 5 shows the expected tendency: the ratio of the iterations with the correct results or results with less than 10% deviation from the correct values is increasing when fault-mitigation techniques are implemented. Additionally, Fig.6 shows that the ratio of rare detrimental faults (right bottom corner of the plot) is reduced by the memory scrubbing with Hamming encoding.

C. *System-level behavior*

As it was observed from Fig.5 , the modeled system has shown the system behavior: the corruption of some memory regions causes significant degradation in the correctness of the final result; when the corruption of other regions has less detrimental or none effects.

While 20 000 iterations (one SEU per iteration) cannot cover the whole space of possible SEU introduction options (~ 200 000 executed instructions and 36 800 memory bits), some system characteristics converge as it is shown in the next experiment.

The same filtering algorithm was executed 1000, 5000, 10000, 20000, 30000, and 40000 iterations. Fig. 7 represents the histogram obtained during these simulations:

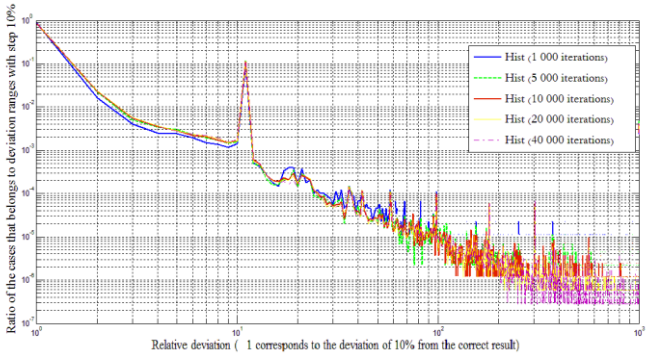

Figure 7. The dependency of system behavior on the number of simulation iterations

Consequently, it is not necessity to survey the whole space of fault introduction to compare the efficiency of fault-mitigation techniques if the convergence is observed.

D. *Optimization of fault-mitigation techniques*

Since the OBC has strict limitations on used hardware resources, the optimization of fault-mitigation techniques is required.

According to the observations from the previous sub-sections, the memory regions have different influence on the whole program execution. The simple clustering algorithm can be used to find out the most influential memory regions taking Fig.5 as an input data.

We created the clustering algorithms that is based on the Euclidian distances in the multidimensional space discussed in Section III.C. The simulation result is presented by Fig.8 (500 iterations).

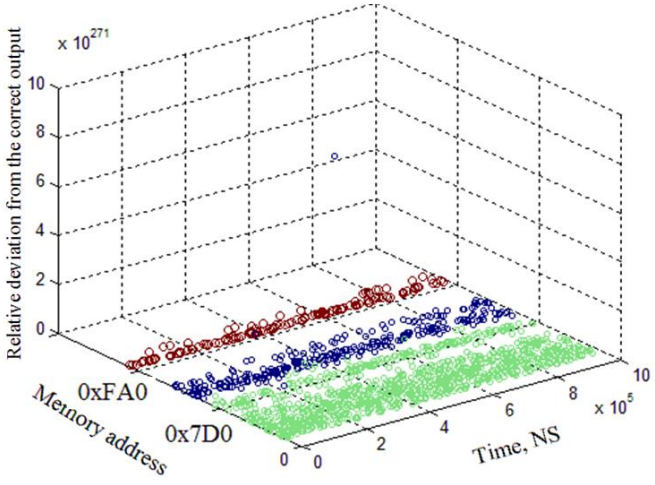

Figure 8. Clustering algorithm output

The presented clustering approach assists the designer to understand the most important memory regions from OBC fault-tolerance point of view and save hardware resources without significant loss in fault-tolerance. For example, it is not

practical to scrub 45% of *.text* and *.rodata* regions since they do not have such detrimental effect as other regions (Fig.5). Thus, we can implement FPGA-based scrubbing only for other 55% which will reduce the scrubbing turnaround period and the amount of allocated memory for syndromes and backup copy.

V. CONCLUSIONS

This paper proposes the statistical simulation approach for fault-tolerant analysis of Small Satellite OBCs based on COTS components. The paper presents the system and fault modeling approaches that in conjunction can be used to understand the satellite sub-system behaviour in radiation environment and compare the efficiency of different software/hardware fault-mitigation techniques.

The work presents how to build the high-level system model for COTS-based OBCs, incorporate the dedicated fault injection and tracking mechanisms, and keep software and FPGA configuration portability to real hardware. The papers clarifies the choice of fault models for Small Satellite with orbits lower 750 km.

As a case study, the work presents the model of Microsemi SmartFusion System-On-Chip (SoC) that incorporates Cortex-M3 CPU, Flash-based FPGA fabric, SRAM and Flash memory storages, Watchdog, Timers, etc. The chosen benchmark program is an adaptive filtering algorithm for the satellite attitude control. The FPGA-based memory protection with Hamming encoding is implemented, assessed, and optimized basing on the simulation results. The comparison between FPGA-based memory scrubbing and a software mitigation technique is presented. The importance of exception handling for satellite OBCs is explained: the proper exception handling may cover up to 11% of wrong computation results.

A shown optimization method based on a clustering algorithm can assist satellite designers in system-level analysis, development, and optimization. Using the proposed method, the limited OBC hardware resources can be allocated with high efficiency. In the presented case study, the method helped to reduce the scrubbing turnaround period and used memory resources almost by half.

Using the presented modeling and simulation methodology for the fault-tolerance assessment of an OBC, well-known fault mitigation techniques can be investigated and improved for different applications. This allows for the early assessment of system vulnerability, improving the overall satellite reliability. The presented methodology can be easily adapted for other OBC configurations.

REFERENCES

[1] G. Beltrame, C. Bolchini, L. Fossati, A. Miele, D. Sciuto, "A Framework for Reliability Assessment and Enhancement in Multi-Processor Systems-On-Chip," 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, pp. 132 - 142, 2007.

[2] O. Ruano, J.A. Maestro, P.Reyes, and P. Reviriego,"A Simulation Platform for the Study of Soft Errors on Signal Processing Circuits through Software Fault Injection," IEEE International Symposium on Industrial Electronics, pp. 3316-3321, 2007.

[3] Frank Ghenassia, "Transaction Level Modeling with SystemC TLM Concepts and Applic-ations for Embedded Systems," ISBN 10 0-387-26232-6. Springer, 2005.

[4] K.Rothbart, et al, "A Smart Card Test Environment using Multi-level Fault Injection in SystemC," 6th IEEE Latin-American Test Workshop, pp. 103-108, March-April 2005.

[5] Y. Y. Chen, Chung-Hsien Hsu, and Kuen-Long Leu, "SoC-level Risk Assessment using FMEA Approach in System Design with SystemC," IEEE International Symposium on Industrial Embedded Systems, pp. 82-89, July 2009.

[6] Y. Y. Chen, Chung-Hsien Hsu, and Kuen-Long Leu, "Analysis of System Bus Transaction Vulnerability in SystemC TLM Design Platform," Proc. the 3rd WSEAS International Conference on Computer Engineering and Applications, pp.284-289, 2009.

[7] J.-C.Ruiz, P. Yuste, P. Gil, and L. Lemus, "On Benchmarking the Dependability of Automotive Engine Control Applications," International Conference on Dependable Systems and Networks, pages 857-866, June-July 2004.

[8] Y. Y. Chen, Chung-Hsien Hsu, and Kuen-Long Leu, "SoC-level Risk Assessment using FMEA Approach in System Design with SystemC," IEEE International Symposium on Industrial Embedded Systems, pp. 82-89, July 2009.

[9] The official web-site of Open Virtual Platforms (OVP) project. <http://www.ovpworld.org/> .

[10] Andrew S. Keys and Michael D. Watson, "Radiation Hardened Electronics for Extreme Environments", NASA Marshall Space Flight Center, December 15, 2007.

[11] Microsemi Corporation, "SmartFusion Customizable System-on-Chip (CSoC) ", Revision 7, August 2011.

[12] R.K. Lawrence,"Radiation characterization of 512Mb SDRAM, IEEE Radiation Effects Data Workshop", July, 2007.

[13] E.H. Cannon, et al., "Heavy ion, high-energy, and low-energy proton SEE sensitivity of 90-nm RHBF SRAMs, IEEE Transactions on Nuclear Science", 57(6)L3493-3499, December 2010.

[14] Farokh Irom and Duc N. Nguyen, "Single Event Effect Characterization of High Density Commercial NAND and NOR Nonvolatile Flash memories", IEEE Transactions on Nuclear Science, pp. 2547-2553, December, 2007.

[15] Kurt Anderson, "Low-cost, radiation-tolerant, on-board processing solution", IEEE Aerospace Conference, pp. 1-8, March, 2005.

[16] T.Takano, T.Yamada, K.Shutoh, and N.Kanekawa, "In-orbit experiment on the fault-tolerant space computer aboard the satellite HITEN", IEEE Transactionson Reliability, 45:624-631, August 2002.

[17] GregoryR.Allen and Gary M. Swift, "Single event effects test results for advanced field programmable gate arrays", IEEE Radiation Effects DataWorkshop,pages 115-120, July2006.

[18] J.-C.Ruiz, P. Yuste, P. Gil, and L. Lemus, "On Benchmarking the Dependability of Automotive Engine Control Applications", International Conference on Dependable Systems and Networks, pages 857-866, June-July 2004.

[19] Y. Y. Chen, Chung-Hsien Hsu, and Kuen-Long Leu, "SoC-level Risk Assessment using FMEA Approach in System Design with SystemC", IEEE International Symposium on Industrial Embedded Systems, pp. 82-89, July 2009.