Microelectronics Journal 45 (2014) 1335–1341

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

System fault-tolerance analysis of COTS-based satellite on-board computers

Dmitry Burlyaev *,[1], Rene van Leuken

EEMCS/ME/CAS, Delft University of Technology, Postbus 5, 2600 AA Delft, The Netherlands

ARTICLE INFO

Article history:
Received 14 February 2013
Received in revised form
3 October 2013
Accepted 7 January 2014
Available online 31 January 2014

Keywords:
Satellite dependability
SystemC modeling
Fault-tolerance techniques
System-on-chip
FMEA

ABSTRACT

Fault-tolerance analysis reveals possible system behavior under the influence of faults. Such analysis is essential for satellites where faults might be caused by space radiation and autonomous recovery is needed. In this paper we present a statistical simulation approach for fault-tolerance analysis of satellite On-Board Computers (OBCs) that are based on Commercial Off-The-Shelf (COTS) components. Since the logic level of COTS electronics is unknown to satellite designers, a new higher-level fault-tolerance analysis is required. We propose such technique that relies on OBC modeling and fault modeling, based on the modeling principle of Single-Event Upsets (SEUs). For the first time we can compare the efficiency of fault-tolerance techniques implemented in software and Field-Programmable Gate Array (FPGA). In addition, our approach enables to analyze system fault-tolerance at early development stages. In a case study the approach is applied to an OBC with a Microsemi SmartFusion SoC, that executes a satellite attitude control algorithm. The gained statistical simulation results enabled 50% reduction in the hardware overhead of the implemented memory scrubbing technique without loss in fault-tolerance. Our method revealed critical fault-tolerance drawbacks of the initial system design that could have lead to satellite mission failure.

1. Introduction

The main requirement for COTS-based satellite OBCs is their tolerance towards radiation-induced faults. Since COTS-based satellites are sensitive to radiation due to the low radiation tolerance of COTS components, the satellite designers have to use fault-tolerance techniques. However, the benefits and correctness of the applied fault-tolerance techniques are checked only under radiation tests at the latest development stage. If the system fails under a radiation test, expensive and time-consuming re-design is required. The comparative fault-tolerance analysis and system-level debugging are impossible with such system testing due to its high cost and the uncontrollable fault-injection procedure [1].

An alternative approach for the system-level fault-tolerance analysis is system simulation. In addition to the growing complexity of electronic components, the existing methods are limited by the fact that the logic level of COTS electronics is unknown to satellite designers. A new higher-level simulation-based approach for fault-tolerance analysis is needed. This paper proposes such a SystemC-based [2] approach and presents how it was applied to an OBC with a SmartFusion SoC [3] that executes the satellite attitude determination and control algorithm.

The remainder of the paper is organized as follows: Section 2 describes related work. Our proposed method is twofold: a system model and a fault model. Correspondingly, Section 3 explains the OBC modeling approach and Section 4 explains the chosen fault model. Section 5 introduces our statistical analysis that joins the OBC model and the fault model. Section 6 presents our case study. Section 7 summarizes the results.

2. Related work

A simulation-based approach for fault-tolerance analysis has already been used to study satellite sub-systems [4,5]. They investigate the fault tolerance through faults injection procedures. However, these approaches are based on the assumption that the logic circuit level of the used electronic components is known, which is not the case when COTS electronics are utilized. Moreover, low-level simulation is time-consuming and cannot be used for an extensive statistical analysis of complex heterogeneous systems.

The usage of high-level abstraction modeling language, such as SystemC [2], is imperative to simulate modern satellite sub-systems.

* Corresponding author. Present address: INRIA, 655 Avenue de l'Europe, 38334 Montbonnot, France.
E-mail addresses: burlyaev.dmitry@gmail.com (D. Burlyaev), t.g.r.m.vanleuken@tudelft.nl (R. van Leuken).

[1] The research was carried out under a contract with Innovative Solutions In Space B.V.

0026-2692/$ - see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.mejo.2014.01.007

A common approach for fault-tolerance analysis is based on the insertion of a Fault Injection Module (FIM) into the interconnections between functional blocks of a system model [6–8]. However, these approaches have been investigated without the consideration of hardware–software co-design and the comparative analysis of fault-tolerance techniques, which are covered in this paper.

3. Proposed system model

The system model consists of the model of an electronic device/OBC and supporting mechanisms.

The model of OBC is based on a Transient-Level Modeling (TLM) methodology [2] (Fig. 1, the green section). The TLM approach simplifies the system model, shortens the simulation run time, and allows the system hardware–software co-design and co-simulation. TLM is used to interconnect functional blocks (CPU, FPGA, Memory blocks, Timers, Decoder/central bus, etc.) that have to be considered as 'black boxes' because their full configurations in COTS components are unknown to satellite designers.

Fig. 2 shows the interconnection between the Decoder/central SoC bus (e.g. AMBA bus) and the FPGA in details. Their interface is implemented through an intermediate layer to support the FPGA co-processor portability from the simulation environment to a real device. The layer is named Fabric Controller Model. It communicates with the co-processor using signals according to a particular protocol (e.g. AMBA protocol). On the opposite side, the Fabric Controller Model is connected to the Decoder/central bus through a TLM channel. The FPGA co-processor configuration (Fig. 2, blue section) is described in Register-Transfer Level (RTL) SystemC.

The supporting modules of the system model do not represent an electronic device but provide supporting functions for the radiation environment modeling and system state observability (Fig. 1, *Injector.obj* and *Observer.obj*). In particular, a simulation tracking mechanism (hereafter 'the Observer') and a fault-injection mechanism (hereafter 'the Injector') are two main supporting components interconnected with the OBC model (Fig. 1). The Observer provides information about the system state (memory content, bus transactions, etc.) at different moments of simulation to a user. The Injector controls the fault injection procedure according to the created time schedule and the pre-defined fault models.

The proposed fault injection technique mainly operates with object pointers. The Injector associates the fault models with functional blocks where particular faults can occur. It creates a

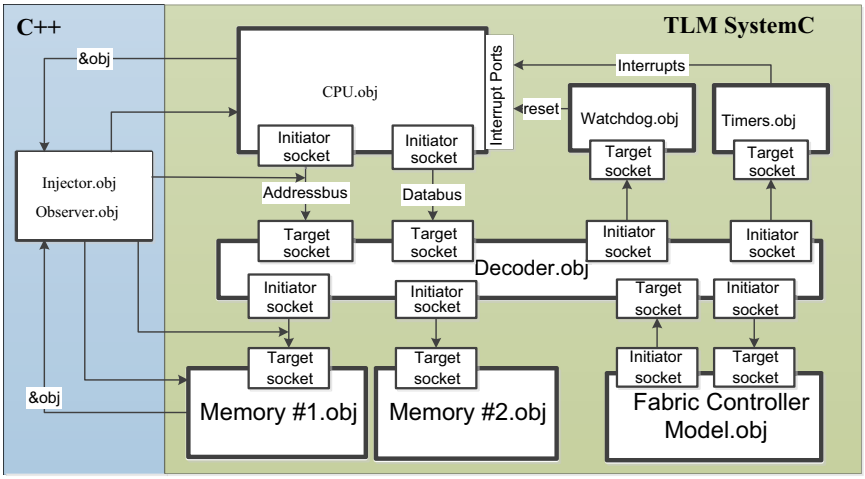

Fig. 1. The system model structure. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

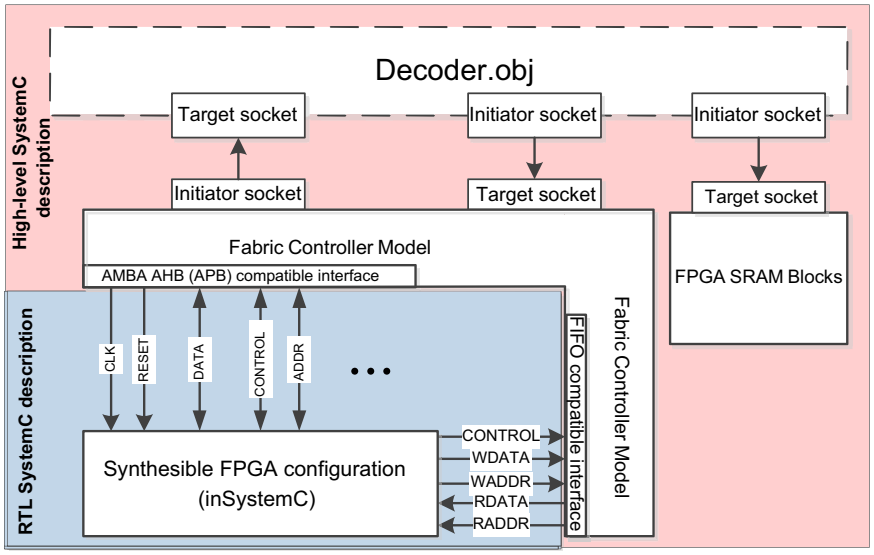

Fig. 2. The FPGA fabric model as a part of the system model. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

fault-injection time schedule (e.g. Fig. 3) according to the predefined fault rates either uses the schedule written by a user.

The presented system model supports software–hardware co-design and is applicable for a system performance analysis. However, a proper fault-model has to be chosen to assess the influence of radiation environment on COTS electronic components. The fault modeling relies on the radiation environment estimation for a satellite mission and is explained in the next section.

4. Radiation environment and fault modeling

The inside of COTS components is unknown to satellite designers. Consequently, the only way to simulate radiation-induced errors is to change the outputs or/and the states of its functional blocks in the proposed high-level OBC model (Section 3).

```
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)
```

Fig. 3. An example of the fault time schedule.

The accuracy of the fault-tolerance analysis depends on the accuracy of the fault models with regard to the real system behavior. Since the utilization of COTS-based components is popular in the small satellite industry, the fault models and fault rates have been built particularly for satellites with 3-years mission at orbits lower than 750 km. The total amount of radiation absorbed during the mission time (Total Ionised Dose, TID [9–11]) is presented in Fig. 4 and was estimated by SPENVIS [12]. SPENVIS also provided the Linear Energy Transfer (LET) spectrum [13] for space radiation particles, see Fig. 5.

For the small satellites, the following fault models are valid [14–17]:

1. *SEU – Single Event Upset or bit-flip in a memory cell* (*CPU registers*, *memory arrays*, *block outputs*, *etc.*).
2. *MCU – Multiple-Cells Upset or multiple SEU in adjacent memory cells* (*CPU registers*, *memory arrays*, *block outputs*, *etc.*).
3. *SEFI – Single Event Functional Interrupt*, *the functional block/electronic component is put into an unknown state or frozen. SEFI*

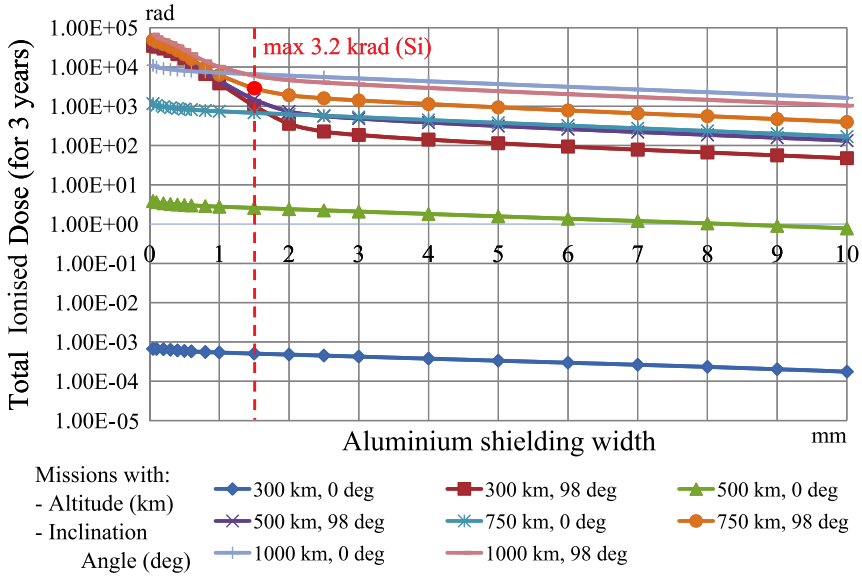

Fig. 4. Simulation results: TID vs aluminium shielding width for satellite missions characterized with their altitudes (km) and inclination angles (deg).

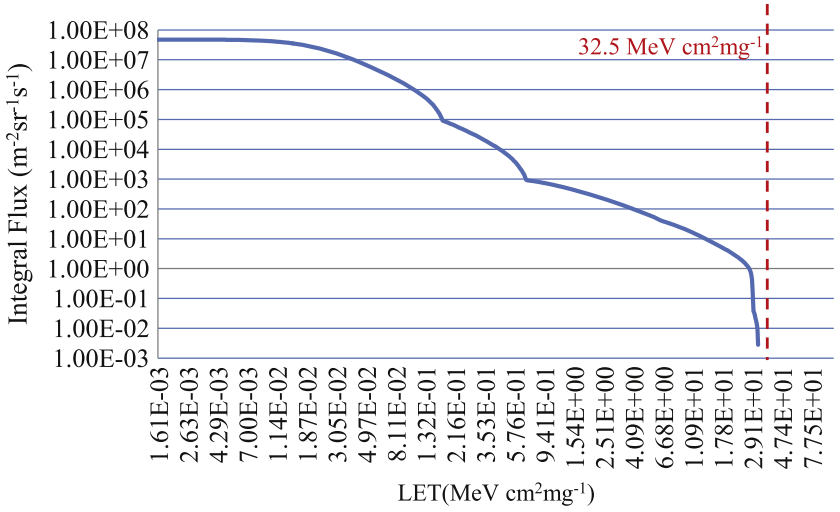

Fig. 5. Simulation results: spacecraft shielded linear energy transfer-LET(Si) spectrum.

Table 1
Radiation sensitivity of COTS components, the worst case.

Component type	Malfunction at TID (krad(Si))	bit-flip rate (upset/bit/day)	SEFI rate (event/device/day)
DRAM	>20	1.45×10^{-8}	0.26
SRAM	20	1.27×10^{-4}	Not observed
Flash NAND	15	1.38×10^{-9}	0.013
Flash NOR	10–20	Tolerant	0.0013

corresponds to a SEU/MCU error in the control circuit of an electronic component.

Using the LET spectrum (Fig. 5), the fault rates have been calculated [18] and presented in Table 1.

According to Table 1, only a TID of 15 krad or more may cause the permanent functionality loss of heterogenous SoCs. But by using an affordable 1.5 mm aluminium shielding, a small satellite will not reach such high TID levels during its typical 3-years mission (Fig. 4). As a result, only temporal effects, such as SEFIs and bit-flips (SEU/MCU), should be taken into account.

According to the observations in [19], SEFI in CPU can be modeled by CPU freezing; SEFI in memory can be simulated by the memory block that stopped responding. However, the knowledge about all possible SEFI consequences is limited since the logic level of the control logic is unknown.

Single-Event Transient (SET) faults are also expected in COTS components in space radiation environment. However, a SET fault or its propagation may cause the change of memory cells' content which also corresponds to SEU/MCU fault models. Thus, additional SET fault models are not required on the high abstraction level we discuss.

Both SEU and MCU can be modeled by flipping the content of memory cells (e.g. in registers, Memory blocks) or by changing 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. Due to the spatial nature of a MCU and limited knowledge about the logic-level implementation, MCU fault models are simplified and correspond to the adjacent multiple bit-flips in the same register or memory array. Such MCU simplification may lead to incorrect results since simultaneous multiple upsets of different but physically adjacent registers (memory arrays) are not taken into account by the model. In this case, additional information is needed to assure the negligible probability of such multiple upset or to simulate possible combinations of bit-flips in physically adjacent registers (memory arrays) to obtain more accurate simulation results.

The presented fault modeling is complicated when not all used memory resources are known. The limited information about the logic level of COTS components may lead to such absence of knowledge. In such cases, the fault injection can be applied to the outputs of the corresponding functional blocks (in particular, corrupting the content of TLM transaction objects). We model radiation-induced errors inside the user memory of Flash-based or Anti-fuse FPGAs through the use of the Fabric Controller Model as a FIM. SRAM-based FPGA type is not considered in this work.

In Section 5 we propose the generalized simulation approach for the fault-tolerance analysis with SEU fault-model.

5. Proposed statistical fault-tolerance analysis

Our fault-tolerance analysis is built upon two discussed components: an OBC model (Section 3) and a fault model (Section 4). This section describes how to use them in a single simulation procedure to investigate the fault-tolerance properties of an OBC.

During the instantiation of the OBC model we identify and collect OBC memory regions used in an application (e.g. memory ranges, list of registers, and TLM objects). They are used as the memory address axis (Fig. 6). The second horizontal axis corresponds to the time of the memory error injection. Thus, any error injection can be characterized by its memory location and the time of occurrence. Other dimensions are named as *quality measurements* and represent the correctness of the OBC computation output. For different programs the quality measurements can be different (introduced by OBC designers): a total execution time, the deviation of an output value from the correct result, the final computation fault mode [20,7], etc. The quality measurements allow the fault-tolerance comparison 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 (see Table 1). As a result, one fault injection per algorithm execution is a justified assumption that we made in this paper, in particular one fault per an attitude control algorithm iteration.

OBC software can be divided into sub-programs that are executed sequentially by a CPU. The sub-programs occupy memory resources (e.g. registers *R*0-*R*1, stack, RAM 0*x*…−0*x*…− see Memory axis, Fig. 6) and the CPU execution time (see Time axis, Fig. 6).

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 a stack, Fig. 6), the influence of a fault that happened during the execution of f_1 on the f_2 sub-program can be investigated by the fault injection to the commonly used memory locations just before f_2 is being executed. It makes the approach modular, scalable, and parallelizable.

The iterative simulation of the OBC is performed with one fault injection procedure per iteration. The simulation result of each iteration is mapped to the multidimensional space to see the overall picture of faults' effects. A case study of the presented methodology is given in the next section.

6. SmartFusion SoC: case study

A model of an OBC with the SmartFusion device [3] has been created to apply the proposed fault-tolerance analysis. The instruction-accurate model of a Cortex-M3 CPU is obtained from the Open-Virtual Platforms project [21] and used as a CPU functional block. One SEU (bit-flip) per simulation iteration is injected to the memory region where the program of an adaptive filter is located.

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. 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 output values (calculated with a bit-flip introduction) from 90 correct values. The result of the simulation is presented in Fig. 7. This figure is an experimentally gained example of Fig. 6 as discussed in the previous section.

Fig. 7 shows that a bit-flip injection to only particular memory locations leads to incorrect output values; colored dots represent these cases. At the same time, a bit-flip injection to other memory regions, e.g. the address range 0*x*0–0*x*19*a*, did not corrupt the calculation output. It can be explained by the interrupt vectors that are located there and not directly used during the performed CPU computation. The *main* function code is located from the address 0*x*19*a* where the first erroneous outputs can be observed after SEU injection (represented by colored dots).

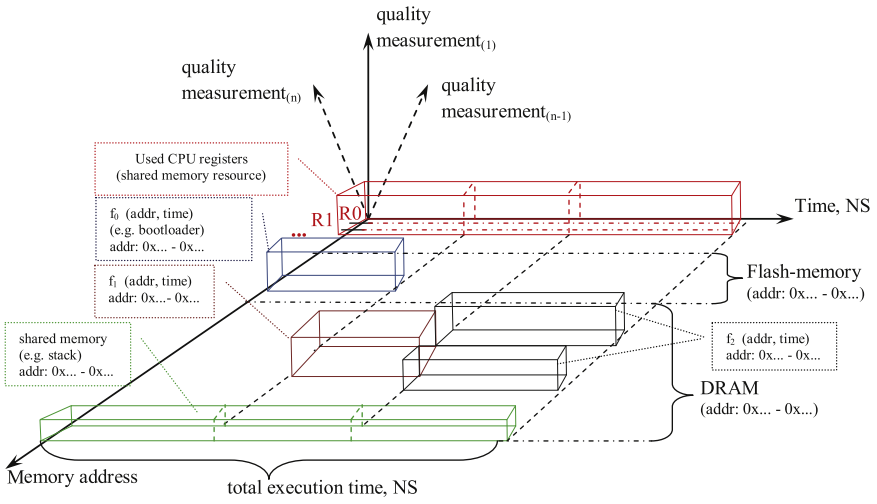

Fig. 6. The representation of multidimensional fault-tolerance analysis.

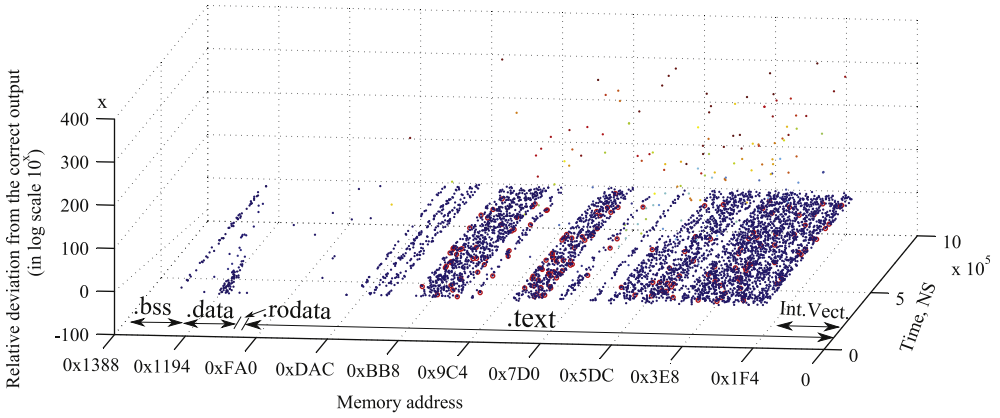

Fig. 7. The simulation result of the adaptive filtering computation with one SEU introduction (20,000 iterations). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

6.1. Hardware-based fault-tolerance technique

Since a SmartFusion SoC includes a FPGA fabric, some fault-tolerance techniques can be implemented in the FPGA. Memory scrubbing, based on Hamming encoding, is chosen as an example. Since *.text* and *.rodata* sections of the filtering program are static, they can be protected with the chosen memory scrubbing technique. In total, 4.1 Kilobytes of memory are protected.

Before the program execution, the CPU sends to the FPGA co-processor the beginning and end addresses of memory ranges that should be protected. The co-processor encodes the data of these ranges saving Hamming syndrome vectors. When it finishes, the co-processor sends to the CPU a message indicating that the CPU can continue the algorithm execution. Since the syndrome vectors are known now, the co-processor starts scrubbing the protected memory checking its correctness. However, the syndrome vectors can also be damaged by SEU; so their additional backup version is used to check if the syndrome or the protected region is damaged. The size of the syndrome vectors (as well as their backup version) is 245 bytes; the codeword length for such Hamming encoding equals 127 bits.

Since an OBC has limited hardware resources, the optimization of the fault-tolerance technique is required. As mentioned before, the corruptions of different memory regions have different influence on the whole program execution. A clustering algorithm can find the most influential memory regions taking Fig. 7 as an input data. We created the clustering algorithm that is based on the Euclidian distances in the multidimensional space. Fig. 8 shows the output of the clustering algorithm applied to 500 simulation iterations (500 dots). The output presents three memory clusters: red, blue, and green.

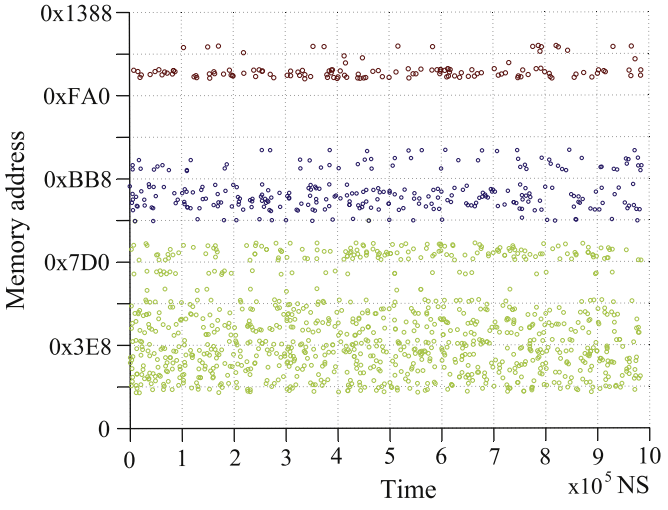

Fig. 8. Clustering algorithm output applied to 500 test cases. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

The approach helps to group important memory regions into clusters to be protected as continuous memory ranges with memory encoding techniques. In the observed case study 45% of *.text* and *.rodata* regions can be kept unprotected since their corruption does not have such detrimental effect as other regions. Thus, we can implement the FPGA-based scrubbing only for the remaining 55% (Fig. 8: three clusters marked with red, blue, and green) which also reduces the scrubbing turnaround period and the required memory for the syndromes by half.

6.2. Software-based fault-tolerance technique

Another used fault-mitigation technique is software-based value limitation: the absolute difference between consecutively calculated filter outputs is restricted. Such technique catches the abnormal behavior (spikes) as shown in Fig. 9.

As the picture presents, Kalman filter calculates three values of satellite angular rates. When a satellite stabilizes its orientation towards the Earth, its angular rates converges to zero as it is happening during 0–675 s. SEU injections into the used memory locations cause the spikes in the filter outputs, e.g. the spikes at ~676 s and ~875 s as given in Fig. 9. Such spikes can be caught if the expected output value range and the speed of its change are known. In the case study, the output is considered to be abnormal if its change is bigger than 10^{-4} rad/s relative to its value during the previous filter iteration. If such abnormal behavior is observed, the output value is ignored and the filter is reset. The time-redundancy (or re-calculation) is not used here because the filter converges fast enough to prevent the satellite mission loss.

6.3. Experimental results

The OBC has been modeled with three modifications: with the observed FPGA-based memory scrubbing, the software-based fault-tolerance technique, and without any fault-tolerance protection. One SEU has been injected for each OBC simulation run, 20,000 simulation iterations in total. The simulation results are presented in Table 2. First, the results show the expected tendency: the ratio of the simulation results with [0–10)% deviation from the correct filter values is increasing when the fault-mitigation techniques are used. Table A1 (see the Appendix) presents the simulation time for different case studies.

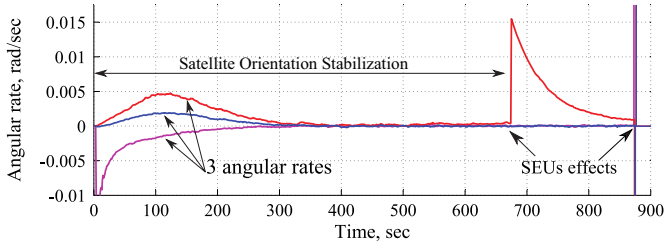

Fig. 9. Kalman filter behavior with two bit-flip introductions at ~675 and ~875 s.

Table 2
System fault-tolerance with and without fault-mitigation techniques.

Output relative deviation from the correct results (%)	Ratio of iterations with such deviation (20,000 iterations)		
	Without protection (%)	SW limitation (%)	FPGA-based scrubbing (%)
[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

The large number of cases where the relative deviation equals [100–110)% can be explained by the next observation. The CPU often raises exceptions after SEU injection, e.g. when it cannot decode a corrupted instruction. If the exception routine is not defined (as in the presented case), the algorithm execution stops and the output values stay equal to zero. When the output value is zero, its relative deviation from the correct result equals 100%. The figures explicitly show the importance to keep the exception handlers in the program to prevent CPU freezing.

In addition, the modeled OBC showed a strong general behavior under a bit-flip introduction. While we cannot cover all possible SEU injection combinations in general case (e.g. $\approx 200,000$ executed instructions on Time axis and 36,800 memory bits on Memory axis), system fault-tolerance characteristics converge. We conducted the same fault-injection procedure 1000, 5000, 10,000, 20,000, 30,000, and 40,000 times (one bit-flip per time). The obtained results (such as in Table 2) converge fast and the obtained tendency with 5000 iterations is accurate enough to compare fault-tolerance techniques and to observe that the absence of exception handlers negatively influences the OBC fault-tolerance.

7. Conclusions

The paper proposes a high-level simulation approach for statistical fault-tolerance analysis. The high-level SystemC-based abstraction level allowed us to overcome the complexity of modern heterogenous SoCs and co-simulate several functional blocks such as CPU, FPGA-based co-processor, a central bus, and memory storages. At the same time, the radiation effects at the logic level had to be projected to the high level of TLM objects due to the absence of knowledge about the logic level of used COTS electronic components. While such projection can be precise for SEU modeling, the modeling of multiple memory cells upset can be inaccurate due to the absence of knowledge about the physical circuit layout.

The simulation approach has been demonstrated on a real case study of a satellite OBC with Microsemi SmartFusion SoC. The performed radiation environment estimation justified the choice of one SEU per algorithm execution as a fault model. The typical small satellite aluminium shielding of 1.5 mm prevents the permanent errors in modern electronic components but fault-tolerance techniques still have to be applied to prevent mission failures caused by soft errors.

The case study revealed the importance of exception handling for the OBCs: the proper exception handling may fix up to 11% of wrong computation results. Until such simulation results have been obtained, the satellite designers did not keep the exception handlers that were not needed from a pure functional point of view, which could have lead to the mission failure or to the reduced satellite availability. In addition, the method enabled the reduction of the scrubbing turnaround period and the required memory resources almost by half since only 55% of the used memory resources caused detrimental consequences in the presence of a bit-flip and had to be protected.

For the first time such modeling technique allowed us to estimate and compare the efficiency of hardware and software fault-tolerance techniques. Consequently, using the proposed methodology, well-known fault detection and mitigation techniques can be investigated and optimized for different applications. In addition, our methodology can reveal critical fault-tolerance design drawbacks at early development stages.

Table A1
Simulation time of the case studies.

Case study (one run)	Number of instructions	User time (s)	System time (s)	Elapsed time (s)
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 (432×288 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

Appendix A. Simulation time

All simulations have been conducted on a PC with an Intel Core i5-2430 M and 4 GB of DDR3 memory.

References

[1] A.S. Keys, M.D. Watson, Radiation Hardened Electronics for Extreme Environments, NASA Marshall Space Flight Center, GOMACTech, 2007.

[2] F. Ghenassia, Transaction Level Modeling with SystemC TLM Concepts and Applications for Embedded Systems, Springer. ISBN-10: 0387262326, 2006.

[3] SmartFusion Customizable System-on-Chip (CSoC), Microsemi Corporation, August 2011 (Revision 7).

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

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

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

[7] Y.Y. Chen, C.-H. Hsu, K.-L. Leu, SoC-level risk assessment using FMEA approach in system design with SystemC, in: IEEE International Symposium on Industrial Embedded Systems, 2009, pp. 82–89.

[8] Y.-Y. Chen, C.-H. Hsu, K.-L. Leu, Analysis of system bus transaction vulnerability in SystemC TLM design platform, in: Proceedings of the Third WSEAS International Conference on Computer Engineering and Applications, 2009, pp. 284–289.

[9] J.L. Kaschmitter, D.L. Shaeffer, N.J. Colella, Operation of commercial R3000 processors in the LEO space environment, IEEE Trans. Nucl. Sci. 38 (6) (1991) 1415–1420.

[10] M.M. Ibrahim, A.M. Tobal, M. Nahas, M. Refai, FPGA-based on-board computer for LEO satellites, in: IEEE International Conference on Space Science and Communication (IconSpace), 2011, pp. 314–319.

[11] N. Battezzati, F. Decuzzi, L. Sterpone, M. Violante, Soft errors in flash-based FPGAs: analysis methodologies and first results, in: International Conference on Field Programmable Logic and Applications, 2009, pp. 723–724.

[12] European Space Agency (ESA). The official web-site of SPENVIS project, ⟨http://www.spenvis.oma.be/⟩.

[13] K. Anderson, Low-cost, radiation-tolerant, on-board processing solution, in: IEEE Aerospace Conference, 2005, pp. 1–8.

[14] R. Lawrence, Radiation characterization of 512 Mb SDRAMs, in: IEEE Radiation Effects Data Workshop, 2007, pp. 204–207.

[15] E. Cannon, et al., Heavy ion, high-energy, and low-energy proton SEE sensitivity of 90-nm RHBD SDRAMs, IEEE Trans. Nucl. Sci. 57 (6) (2010) 3493–3499.

[16] F. Irom, D.N. Nguyen, Single event effect characterization of high density commercial NAND and NOR nonvolatile flash memories, IEEE Trans. Nucl. Sci. 23 (2007) 2547–2553.

[17] K. Anderson, Low-cost, radiation-tolerant, on-board processing solution, in: IEEE Aerospace Conference, 2005, pp. 1–8.

[18] G.I. Zebrev, I.O. Ishutin, R.G. Useinov, V.S. Anashin, Methodology of soft error rate computation in modern microelectronics, IEEE Trans. Nucl. Sci. 57 (6) (2010) 3725–3733.

[19] T. Takano, T. Yamada, K. Shutoh, N. Kanekawa, In-orbit experiment on the fault-tolerant space computer aboard the satellite Hiten, IEEE Trans. Reliab. 45 (2002) 624–631.

[20] J.-C. Ruiz, P. Yuste, P. Gil, L. Lemus, On benchmarking the dependability of automotive engine control applications, in: International Conference on Dependable Systems and Networks, 2004, pp. 857–866.

[21] The official web-site of the Open Virtual Platforms (OVP) project, ⟨http://www.ovpworld.org/⟩.