An Experimental Evaluation of Fault-Tolerant FPGA-based Robot Controller

Jakub Podivinsky, Jakub Lojda, Zdenek Kotasek
Faculty of Information Technology, Brno University of Technology,
Centre of Excellence IT4Innovations
Bozetechova 2, 612 66 Brno, Czech Republic
{ipodivinsky, ilojda, kotasek}@fit.vutbr.cz

Abstract

Field Programmable Gate Arrays (FPGAs) are becoming more popular in various areas. Single Event Upsets (SEUs) are faults caused by a charged particle in the configuration memory of SRAM-based FPGAs. Such a charged particle can cause incorrect behavior in the whole system. This problem becomes greater if such a system operates in an environment with increased radiation (e.g. space applications). Lots of techniques to harden FPGAs against faults exist and new techniques targeted to FPGA are in scope of many researchers. One such technique is called Triple Modular Redundancy (TMR). It is important to evaluate these techniques on a real system with a real FPGA. An evaluation platform based on an artificial fault injection and a functional verification for testing fault tolerance methodologies is introduced in this paper. Parts of our experimental system are hardened by using TMR and its experimental evaluation is one of the main parts of this paper. In this paper, we focus on the TMR fault tolerance method and change the target functional unit, on which the method is applied. This allows us to determine the reliability gain obtained through the hardening of a particular functional unit and allows us to compare the results. We propose experiments with various fault injection strategies (multiple and single faults) and monitor impact of faults on both the electronic and mechanical parts of the experimental system.

1. Introduction

Field Programmable Gate Arrays (FPGAs) are becoming more popular for a number of reasons. SRAM-based FPGAs can provide hardware implementation of applications that are often faster than processor-based implementations and require lower costs than Application-Specific Integrated Circuits (ASICs) [4]. Moreover, their reconfigurability makes them flexible almost like processors do and offers us to change implemented application during the life cycle of such a system. FPGAs can be used in different areas, e.g. automotive, aerospace or space. SRAM-based FPGAs consist of programmable components (configurable logic blocks, look-up tables, flip-flops, etc.) and their interconnection. Configuration of FPGA is stored as a bitstream in the configuration memory. Many FPGAs use an SRAM memory as their configuration memory. Sensitivity to faults caused by charged particles is the problem of FPGAs from the reliability point of view. A hit of charged particles can lead to the inversion of a bit in configuration memory which can change implemented behavior of the whole system. This problem is called Single Event Upset (SEU). This may be a problem, especially when FPGAs are used in areas with increased radiation, e.g. space applications [12].

Although 80-99% of SEUs hit unused bits of the configuration bitstream [4], it is important to harden FPGA-based systems against faults. Lots of fault tolerance techniques targeted to FPGAs exist and new ones are subject of investigation. The use of spatial redundancy for hardening user logic against SEUs is presented in [12]. One of the main approaches of spatial redundancy is Triple Modular Redundancy (TMR), which many researchers are trying to improve. For example, paper [14] proposes a new fault tolerance method targeted to SRAM-based FPGAs. This technique is based on a heuristic and uses inaccurate modules in combination with TMR. This approach reduces the power and area overhead of the hardened design.

Paper [15] presents improved TMR which interest *don't care bits* of LUT configuration bits. These LUTs are classified to the set of SEU-sensitive and SEU-insensitive. This improved approach just triplicates

978-1-5386-5710-2/18/$31.00

sensitive LUTs and reduces the area overhead unlike the classical TMR approach.

It is important to evaluate fault tolerance techniques targeted to FPGAs. Three main approaches are presented in [4]: 1) modeling tools, 2) fault emulation testing and 3) accelerated radiation testing. Authors of [2] present emulation method which emulates effect of SEUs in the configuration of FPGAs. Presented method is based on combination of simulation and topological analysis of the circuit configured in FPGA. Methotds based on synthesizable model of faults are presented in [11]. Authors propose fault injection tool which allows to inject faults to FPGA, but various synthesizable fault models must be added to the original design. This method requires modifications of the original circuit in the VHDL, additional gates and wires must be inserted. FLIPPER, the evaluation platform based on the fault injection directly into an physical FPGA without modification of the original circuit description, is presented in [1]. Thank to the dynamic reconfiguration, it is possible to read, modify and write back the configuration memory. FLIPPER uses two boards with FPGAs, one is the main control board and second is board where tested circuit is implemented. The main board control fault injection which can be driven by the application operating on a computer. Fault injection can be controller by a additional functional unit implemented the same FPGA. This approach is presented in [6]. The main reason is the acceleration of the fault impact evaluation. Fault injection process can be initialized from computer to which experimental FPGA is connected.

In our previous work, we focused on several architectures with various fault tolerance methods. Such as in paper [9], we target a processor based system and its hardening. In paper [5], we focused on systems generated with the use of High-Level Synthesis (HLS) and proposed a mechanism to incorporate fault tolerance into storage elements and operations through redefinition of data types behavior.

In our last research, we developed the evaluation platform which allow us to test fault tolerance properties [8]. The developed platform uses functional verification and previously developed fault injector [13]. The main advantage of our platform is its ability to test the impacts of injected faults not only on electronic controller implemented in FPGA, but also on the mechanical part controlled by the tested electronic controller, because lots of electronic systems control the mechanical part which is an important component of the whole system. In this paper, triplication (TMR) is applied on our experimental system which is composed of robot in a maze and its electronic controller and our platform is used to analyze the reliability gained. It should be noted that a great number of fault tolerant systems are electromechanical applications. As an example, the FT systems in planes can be used. That is why we concentrated on a robot with its controller as the experimental system.

In our research group, we also investigate new methods in the area of fault-tolerant systems design automation. Our aim is to create a fully automated environment to fault-tolerant systems design and its evaluation. The experiments presented in this paper are a step towards this new methodology, as it is important to understand the behavior of various components of the system utilizing different proportions of FPGA primitive types during the presence of faults in these components.

The paper has the following structure. Section 2 introduces our functional verification-based evaluation platform and experimental electro-mechanical system. Experiments and results with multiple and single fault injection are presented in Section 4. Section 5 contains the conclusion of this work and presents plans for our future research.

2. Evaluation Platform and Experimental System

This section briefly describes our evaluation platform for evaluating fault tolerance methodologies which was presented in journal publication [8]. Our platform is based on functional verification [7] and standardized Universal Verification Methodology (UVM) [3]. The task of functional verification is to check if a hardware system matches a given specification. In our case, functional verification is used as a tool for checking if injected faults caused some discrepancy on the output of the tested system. The platform is shown in Figure 1. It is composed of several components running on a computer and on an FPGA development board. The important part of the platform is the software part of the verification environment which is running on a computer. The verification environment observes communication between both parts of the experimental electro-mechanical system (electronic controller and controlled mechanical part). The electronic controller is run on an FPGA which is connected to the simulation of the mechanical part (running on a computer) through the Ethernet interface. Artificial faults are injected through JTAG interface which is used by the fault injector [13]. The fault injector uses partial reconfiguration, it reads specified part of the bitstream stored in configuration memory, inverts the specified bit and writes it back to the configuration memory.

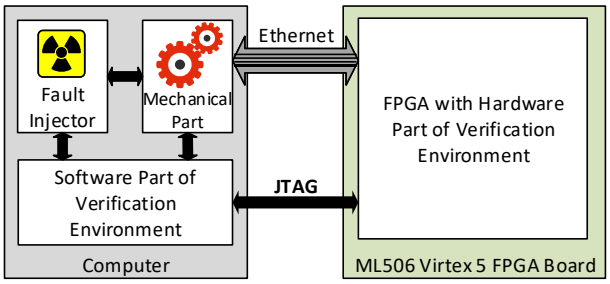

Figure 1. The evaluation platform architecture.

Together with the platform, we need to introduce a process for verifying fault tolerance properties which is composed of three phases. The *first phase* of evaluation process is simulation based verification. Verification is completely done in an RTL simulator (ModelSim) in this phase. The task of this phase is to check if the electronic controller operates correctly according to the given specification. The task of the *second phase* is verification of the electronic controller which is implemented into an FPGA. Scenarios which was obtained during first phase are repeated together artificial faults injection into FPGA. Previously developed fault injector is used in this phase. The impact of injected faults on the behavior of electronic part is monitored in this phase. The goal of the *third phase* is the analysisof the faults which lead to corruption of the mechanical part. The second and third phases use the FPGA-based verification environment, where a device under test is run on the FPGA. The second phase monitors the effect of faults on communication between the electronic controller an the mechanical part. The third phase checks the values of sensors on the mechanical part and monitors its behavior.

3. Case Study: Robot in A Maze

Our goal is to demonstrate the proposed platform and evaluation process on a real electro-mechanical system. Our experimental electro-mechanical system was developed for these purposes. It is composed of a robot for seeking a path in a maze and its electronic controller implemented in FPGA. The robot controller is not a very complex system, but it is split into various components (bus, finite state machines, memories, etc.) which allow us to evaluate a wide scale of fault tolerance methodologies. We do not have a real robot, and, thus, we simulate its behavior. We use the Player/Stage simulation tool which is able to simulate robot and its environment. In our case we simulate the robot in a maze. The simulation tool is running on a computer from which data must be transfered to FPGA board. We use Ethernet interface which allows us to transfer data between the robot in simulation (computer) and its electronic controller (FPGA).

The main component of the Robot controller [10] is central bus through which communication between various functional units is accomplished. Controller consist of 16 functional units, the most important are Position Evaluation Unit (PEU) together with the Barrier Detection Unit (BDU). The main task of BDU is to calculate actual position of robot and also to detect obstacles in the neighborhood. The obtained informations are stored in the Map Memory Unit (MMU) through the Map Unit (MU). Path Finding Unit (PFU) implements the algorithm for seeking path in maze which is based on informations stored in memory (MMU). The Engine Control Unit (ECU) controls mechanical parts of the robot in maze. A control finite state machine (FSM) and bus wrapper are important accessories of almost all functional units.

Figure 2 shows a combined FPGA-based verification environment for the second and the third phases of the proposed evaluation process. The verification environment is composed of two parts: 1) the UVM-based verification environment and 2) the experimental electromechanical system (robot in a maze). The verification environment operates just as an observer which checks communication of the robot in a maze with the FPGA without direct intervention. The golden model is used for the comparison of expected data and really transfered data. Some discrepancy is indicated as an error on the output of the electronic controller. On the other hand, as the third phase, data from sensors and the correct behavior of the mechanical robot are monitored.

4. Experiments and Results

In our experiments, we decided to examine the impact of faults on particular components of the robot controller unit. The experiments comprise the comparison of results obtained from selected unhardened components of the controller unit with their hardened versions. As a method of redundancy insertion, the TMR was selected. Three components of the robot controller unit, the *PEU*, the *BDU* and the *ECU*, were selected for comparison. The reason for this choice was that the *PEU* and the *BDU* components compute the input values for the path-searching algorithm. From this point of view, these components are very important and the complete controller unit is function-dependent on them. The *ECU* component directly affects the

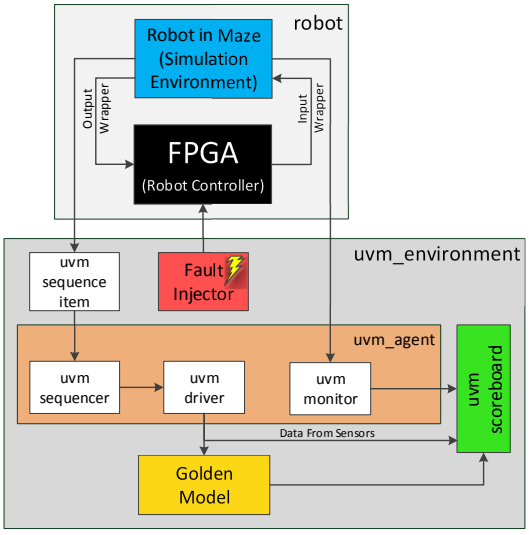

Figure 2. The FPGA-based verification environment for the second and third phases of the robot controller evaluation.

movement of the robot, thus failure of this component would cause the complete controller unit to fail.

In this experimentation, we use our evaluation platform, which is based on a permanent configuration bitstream bit-flip, and, thus, we use this type of error in our evaluation. The experiments were performed with the usage of two fault injection strategies. The first strategy was to inject a *single fault* into an individual component per verification run before the robot was started and observe its ability to reach the target position. The second strategy comprised *multiple faults* injections per verification run. In this case, a number of faults were injected until the first failure propagated to the controller outputs was observed.

4.1. Multiple Faults Injection

In the *multiple faults* injection experiment, permanent bit-flips were injected into utilized *Look-up Tables* (LUTs) contents with a constant period of $5s$. This period was experimentally chosen based on the system failure manifestation time. This means that each $5s$ only one SEU was injected into the particular component of the robot controller unit LUT contents (only utilized LUT bits are considered) until the robot failed or reached the target position.

At first, the *multiple faults* were injected into the *unhardened* version of the robot controller. The reason for this was to find out the behavior of the whole system without any fault tolerance method involved. In this stage, one set of 1000 verification runs was done for each of the selected components in which faults were only injected into the particular controller unit LUTs contents. The statistical results are shown by the *PEU_noft*, the *BDU_noft* and the *ECU_noft* bars of the box plot in Figure 3. Box plot shows for each component the minimum, the first quartile (25%), median, the second quartile (75%) and maximum of the numbers of faults injected into FPGA that for each particular run were enough to manifest a failure on the controller outputs. As can be seen, each component has its own level of susceptibility to SEUs.

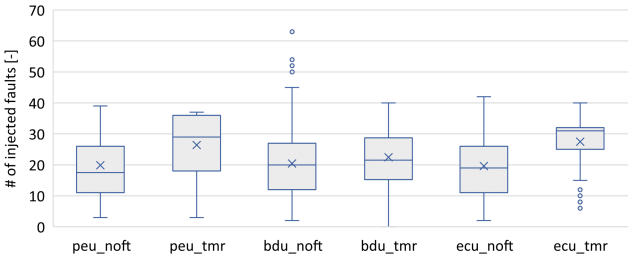

Figure 3. Box plot shows statistical evaluation of number of faults injected into FPGA which led to the electronic failure.

Then, three other robot controller unit designs with the TMR applied selectively to the *PEU*, the *BDU* and the *ECU* components were created. Equivalent experiments were repeated with the three new designs in which only the faults were injected into the particular *hardened* component. The bars *PEU_tmr*, *BDU_tmr* and *ECU_tmr* in the box plot in Figure 3 show the susceptibility to faults with the TMR applied. As can be seen, each of the bars representing the *hardened* version is above the *unhardened* one, therefore, more fault injections were required to cause a malfunction of the complete controller unit.

We must note that when multiple faults were injected, the *hardened* version failed in a smaller number of cases than the *unhardened* version. The numbers of cases in which the complete controller unit failed while faults were injected into the selected components are shown in Table 1. As can be seen, the application of the TMR led up to 93.3% decrease of failure manifestations. We can conclude that the TMR led to a lower number of electronic failures and also led to the increased number of faults injected into FPGA which caused a failure.

Besides the influence of faults on the electronic part of the system, we also observed its influence on the mechanical part. The electronic failure usually stopped

Table 1. The impact of multiple faults injected into the unhardened and hardened versions of robot controller both on the electronic controller and mechanical part.

Monitored impact	*PEU*		*BDU*		*ECU*	
	noft	*tmr*	*noft*	*tmr*	*noft*	*tmr*
Electronic OK [–]	656	977	361	917	226	622
Electronic failed [–]	344	23	639	83	774	378
Goal not reached [–]	344	23	639	83	774	378
Collision with wall [–]	0	0	0	0	15	3
Robot stop on place [–]	344	23	639	83	759	375
Reliability improvement [%]	93.3%		87.0%		51.2%	

the robot on its position and in some cases the failure led to a collision with a wall. It can be noted that the stopping of the robot on its position is a less serious failure consequence than the collision. Table 1 also shows the numbers of cases in which the robot crashed into the wall and the numbers of cases in which the robot stopped at a place.

4.2. Single Faults Injection

In the case of the *single faults* injection experiment, exactly one bit-flip of the utilized LUT contents of a particular component was injected per verification run and its impact on the behavior of the whole controller unit was observed. At first, 1000 verification runs with the *unhardened* design of our robot controller unit were performed. Table 2 shows the numbers of runs that led to an electronic failure. As can be seen, if the *single faults* are injected, the number of failures is significantly lower than in the case of *multiple faults* injection.

Then, the verification runs with *single fault* injections were repeated on the *hardened* design. Table 2 also shows the numbers of runs in which an injection into the selected components with the TMR applied led to a failure (the columns *PEU_tmr*, *BDU_tmr* and *ECU_tmr*). We believe that the fact the *hardened* unit occasionally fails after the *single fault* injection is caused by hitting the *voter* which is needed to interface the particular component with the rest of the system.

The Figure 4 shows the number of faults which lead to the failure of the electronic controller. The figure shows the graphical comparison of the *hardened* and *unhardened* versions. One can see that the *BDU* component is the most vulnerable to faults and this shows that the *BDU* is a really important component of the whole robot controller. One can see, for the *hardened* version of the design, the number of failures is lower for each of the selected components.

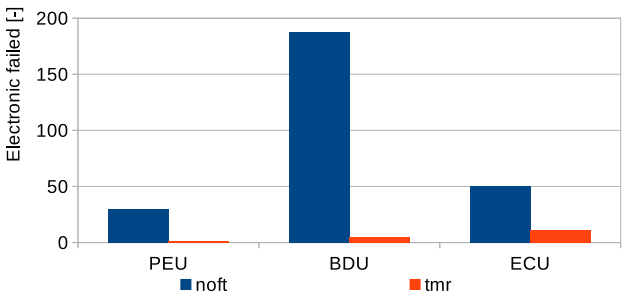

Figure 4. Number of faults injected into FPGA which cause the electronic failure.

As in the previous case, we observed the impact of faults both on the electronic and the mechanical parts of the experimental system as well. Not all of the faults injected into FPGA that caused the electronic controller failure caused the robot to collide with the wall. Table 2 also shows the numbers of wall collisions and cases where the robot stopped during its journey.

5. Conclusions and Future Research

The use of functional verification as a tool for evaluation of impact of artificial faults injected into the configuration of SRAM-based FPGAs was presented in this paper. Our platform was demonstrated on the evaluation of the impact of injected faults on the robot controller which navigates the robot in a maze. The paper shows experimental results with the *hardened* (triplication) along with the *unhardened* robot controller version. Both *single* and *multiple faults* injection strategies were used. Our experiments show the benefit of triplication which in the case of the *single fault* injections led to a lower number of electronic failures. We believe the susceptibility of the *hardened* unit to the *single fault* injection is caused by hitting the

Table 2. The impact of single faults injected on the unhardened and hardened versions of robot controller both on the electronic controller and mechanical part.

Monitored impact	*PEU*		*BDU*		*ECU*	
	noft	*tmr*	*noft*	*tmr*	*noft*	*tmr*
Electronic OK [−]	971	1000	813	996	952	990
Electronic failed [−]	29	0	187	4	48	10
Goal not reached [−]	29	0	187	4	48	10
Collision with wall [−]	0	0	5	0	1	0
Robot stop on place [−]	29	0	182	4	47	10
Reliability improvement [%]	100.0%		97.9%		79.2%	

voter, which is needed to interface the particular component with the rest of the system. In the case of the *multiple faults* injection, it is clearly visible that the triplication led to a lower number of electronic failures, but experiments have also shown that the number of injected faults which cause a failure is higher than in the case of the *unhardened* robot controller. The number of failures is significantly higher than in the case of the *single fault* injections, as in this experiment, *multiple faults* were injected during one verification run.

The results presented in this paper are integral part of our future research in which we will integrate faulty module recovery into our robot controller, which significantly increases the resource utilization, and, thus, we aim to harden only the most sensitive functional units of the robot controller. The results obtained in this research will help us to increase the efficiency of the reliability method.

As for future research, our goal is to use the reconfiguration as a tool for faulty module recovery. We expect that the benefit of recovery will be most obvious in the case of *multiple faults* injection. This expectation will be confirmed or refuted by repeating similar experiments as shown in this paper.

Acknowledgements

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II), the project IT4Innovations excellence in science – LQ1602, the BUT project FIT-S-17-3994 and the JU ECSEL Project SECREDAS (Product Security for Cross Domain Reliable Dependable Automated Systems), Grant agreement No. 783119.

References

[1] M. Alderighi, F. Casini, S. d'Angelo, M. Mancini, S. Pastore, and G. R. Sechi. Evaluation of Single Event Upset Mitigation Schemes for SRAM-based FPGAs Using the FLIPPER Fault Injection Platform. In *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT'07. 22nd IEEE International Symposium on*, pages 105–113. IEEE, 2007.

[2] C. Bernardeschi, L. Cassano, A. Domenici, and L. Sterpone. Accurate Simulation of SEUs in the Configuration Memory of SRAM-based FPGAs. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012 IEEE International Symposium on*, pages 115–120. IEEE, 2012.

[3] V. R. Cooper. *Getting Started with UVM: A Beginner's Guide*. Austin, TX : Verilab, 2013.

[4] P. Gaillardon. *Reconfigurable Logic: Architecture, Tools, and Applications*. Devices, Circuits, and Systems. CRC Press, 2015.

[5] J. Lojda, J. Podivinsky, and Z. Kotasek. Redundant Data Types and Operations in HLS and Their use for a Robot Controller Unit Fault Tolerance Evaluation. In *East-West Design & Test Symposium (EWDTS), 2017 IEEE*, pages 273–278. IEEE, 2017.

[6] C. López-Ongil, M. Garcia-Valderas, M. Portela-García, and L. Entrena. Autonomous fault emulation: a new fpga-based acceleration system for hardness evaluation. *Nuclear Science, IEEE Transactions on*, 54(1):252–261, 2007.

[7] A. Meyer. *Principles of Functional Verification*. Elsevier Science, 2003.

[8] J. Podivinsky, O. Cekan, J. Lojda, M. Zachariasova, M. Krcma, and Z. Kotasek. Functional Verification based Platform for Evaluating Fault Tolerance Properties. *Microprocessors and Microsystems*, 52:145–159, 2017.

[9] J. Podivinsky, J. Lojda, O. Cekan, R. Panek, and Z. Kotasek. Evaluation Platform for Testing Fault Tolerance Properties: Soft-core Processor-based Experimental Robot Controller. In *Digital System Design (DSD), 2018 Euromicro Conference on*. IEEE, 2018.

[10] J. Podivinsky, M. Simkova, and Z. Kotasek. Complex Control System for Testing Fault-Tolerance Methodologies. In *Proceedings of The Third Workshop MEDIAN 2014*, pages 24–27. COST, 2014.

[11] S. Rudrakshi, V. Midasala, and S. Bhavanam. Implementation of fpga based fault injection tool (fito) for testing fault tolerant designs. *IACSIT International Journal of Engineering and Technology*, 4(5):522–526, 2012.

[12] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam. Mitigation of Radiation Effects in SRAM-Based FPGAs for Space Applications. *ACM Comput. Surv.*, 47(2):37:1–37:34, Jan. 2015.

[13] M. Straka, J. Kastil, and Z. Kotasek. SEU Simulation Framework for Xilinx FPGA: First Step Towards Testing Fault Tolerant Systems. In *14th EUROMICRO Conference on Digital System Design*, pages 223–230. IEEE Computer Society, 2011.

[14] S. Venkataraman, R. Santos, and A. Kumar. A Flexible Inexact tmr Technique for SRAM-based FPGAs. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, pages 810–813. EDA Consortium, 2016.

[15] M. S. Zheng, Z. L. Wang, J. Tu, J. Y. Wang, and L. J. Li. Reliability Oriented Selective Triple Modular Redundancy for SRAM-Based FPGAs. In *Applied Mechanics and Materials*, volume 713, pages 1127–1131. Trans Tech Publ, 2015.