FPGA-based Robot Controller: An Experimental Evaluation of Fault Tolerance Properties

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
Email: {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 ones are under investigation. 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 student paper. Parts of our experimental system are hardened by using TMR and its experimental evaluation is one of the main parts of this student paper. We propose experiments with multiple fault injection strategy and monitor faults impact on both the electronic and mechanical parts of the experimental system.**

I. 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) [1]. Moreover, their reconfigurability provides almost the same flexibility as 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 are composed of programmable components (configurable logic blocks, look-up tables, flip-flops, etc.) and their interconnection. FPGA configuration is stored as a bitstream in the configuration SRAM memory. The problem from the reliability point of view is the sensitivity of FPGAs to faults caused by charged particles. A hit of these particles can cause the inversion of a configuration bit which can change the behavior of the whole circuit. This event 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 [2].

Although 80-99% of SEUs hit unused bits of the configuration bitstream [1], it is important to harden FPGA-based systems against faults. Lots of fault tolerance methodologies inclined to FPGAs exist and new ones are under investigation. The use of spatial redundancy for hardening user logic against SEUs is presented in [2]. One of the main approaches of spatial redundancy is Triple Modular Redundancy (TMR), which many researchers are trying to improve. For example, paper [3] proposes a novel technique based on a heuristic to tolerate faults in SRAM-based FPGAs by using inexact modules in conjunction with TMR, thus reducing the area and power overhead of the design. Paper [4] presents an improved approach to TMR which concerns *don't care bits* of LUT configuration bits and hence classifies the set of LUTs into SEU-sensitive and SEU-insensitive. Unlike the full TMR approach, the improved approach only triplicates SEU-sensitive LUTs and can greatly reduce the area overhead while maintaining circuit reliability.

It is important to evaluate fault tolerance techniques targeted to FPGAs. Three main approaches are presented in [1]: 1) modeling tools, 2) fault emulation testing and 3) accelerated radiation testing. The method for the emulation of the effects of SEUs in the configuration memory of FPGAs is presented in [5]. This approach combines simulation and topological analysis of the design mapped on the FPGA. An FPGA-based fault injection tool, which is presented in [6], supports several synthesizable fault models of digital systems and is implemented using VHDL. The fault injection requires an addition of some extra gates and wires to the original design and thus modifies the original VHDL. In [7], a platform called FLIPPER, which is based on the fault injection into a real FPGA board without changing the original design, was presented. This technique is based on dynamic reconfiguration which allows us to read, modify and write back the configuration bitstream. This platform is composed of two boards with FPGAs – the main board and the DUT board. The fault injection is controlled by the main board, which is driven by the software application running on a PC. The authors in [8] focus on the speed of the fault impact evaluation, where the fault injection is fully controlled by a part of the design on the FPGA. Communication with a PC is used only for the initial configuration of the fault injection process.

In our previous work, we developed the evaluation platform for testing fault tolerance properties [9]. The evaluation platform is based on functional verification and uses our fault injector [10]. The main advantage of our platform is its ability to test the impacts of faults not only on electronic controller, but also on the controlled mechanical part, 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 electro-mechanical system (robot in a maze) 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 field 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.

This paper is organized as follows. Section II introduces our functional verification-based evaluation platform and experimental electro-mechanical system. Experiments and results with multiple fault injection are summarized in Section III. Section IV contains the conclusion of the paper and presents our plans for future research.

II. Evaluation Platform and Experimental System

This section briefly describes our evaluation platform for testing fault tolerance methodologies which was presented in journal publication [9]. Our platform is based on functional verification [11] and standardized Universal Verification Methodology (UVM) [12]. Functional verification checks whether a hardware system satisfies 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 which is shown in Figure 1, is composed of several components running on a computer and on an FPGA development board. The main 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 (mechanical part and its electronic controller). 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 [10]. The fault injector is based on partial reconfiguration, it reads part of the configuration memory, inverts the specified bit and writes it back to the configuration memory.

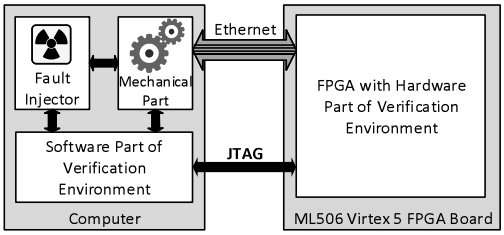

Fig. 1. The evaluation platform architecture.

Together with the platform, we need to introduce a process for verifying fault tolerance properties which is divided into three phases. The *first phase* is simulation based verification. Verification is completely done in an RTL simulator (ModelSim) in this phase. The task of this phase is to test on whether the electronic controller works correctly according to the specification. The *second phase* consists of the verification of the electronic controller implemented into an FPGA with the scenarios obtained during the previous phase, but in addition, artificial faults are injected into FPGAs using an implemented fault injector. The impact of faults on the electronic part is monitored in this phase. The analysis of the faults which corrupted the mechanical part is the goal of the *third phase*. 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 impact of faults on communication between the mechanical part and its controller. The third phase checks the values of sensors on the mechanical part and monitors its behavior.

Our goal is to demonstrate the proposed platform and evaluation process on a complex system. Our experimental electro-mechanical system consists of a robot for searching a path through a maze and its electronic controller implemented in FPGA was developed for these purposes. 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. Unfortunately, we do not have a real robot device, so we use the simulation tool Player/Stage [13], which allows us to simulate the robot and its environment (in our case the robot in a maze). The robot simulation is executed on a computer which is connected with the FPGA board through the Ethernet transmitting data between the robot and its controller.

The robot controller [14] is composed of various functional units, which are interconnected through the central bus. There is in total 16 functional units, the main ones are the Position Evaluation Unit (PEU) and the Barrier Detection Unit (BDU) which calculates the actual position of the robot in a maze and detects barriers in the robot 4-neighborhood. The Map Unit (MU) stores calculated information into the Map Memory Unit (MMU) on which path searching implemented by the Path Finding Unit (PFU) is based. The mechanical parts of the robot are controlled by the Engine Control Unit (ECU). Almost all of these functional units are equipped with a control finite state machine (FSM) and a bus wrapper.

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 software part of UVM-based verification environment and 2) the experimental electromechanical system (robot in a maze). The verification environment just observes communication between the robot in a maze and 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.

III. 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

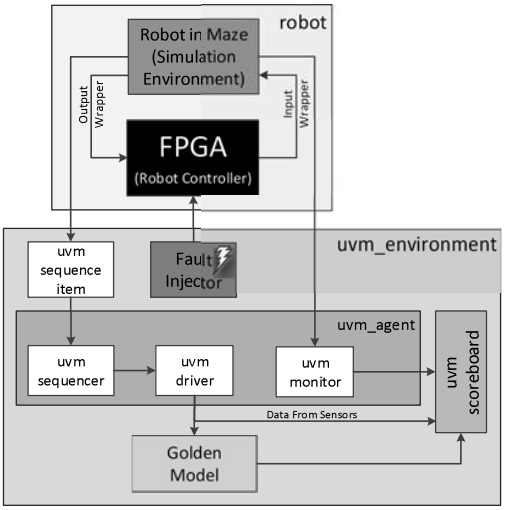

Fig. 2. The architecture of the FPGA-based verification environment for the robot controller.

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

The experiments were performed with the usage of one fault injection strategy. The strategy was to inject *multiple faults* per verification run into an individual component and observe its ability to reach the target position. A number of faults were injected until the first failure propagated to the controller outputs was observed.

A. 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. It is a quartile chart that for each component shows the minimum, the first quartile (25%), median, the second quartile (75%) and maximum of the numbers of injected faults 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.

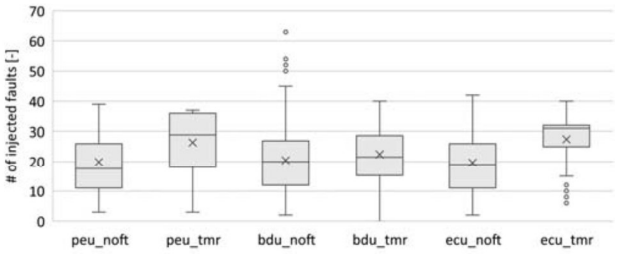

Fig. 3. Box plot shows statistical evaluation of number of injected faults which led to the electronic failure.

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.

It is important to note that while 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 I. 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 injected faults which caused a failure.

TABLE I. 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%	

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 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 I 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.

IV. CONCLUSIONS AND FUTURE RESEARCH

The use of functional verification as a tool for evaluation of impact of faults injected into the configuration memory of

SRAM-based FPGAs was presented in this paper. Our platform was demonstrated on the evaluation of the impact of 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. *Multiple faults* injection strategy was used. Our experiments show the benefit of triplication. 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 led to a failure is higher than in the case of the *unhardened* robot controller.

As for future research, our goal is to use the reconfiguration as a tool for faulty module recovery. We plan to use other fault injection strategy and examine its impact on unhardened and hardened design. 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); project IT4Innovations excellence in science - LQ1602 and BUT project FIT-S-17-39947.

References

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

[2] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam, "Mitigation of Radiation Effects in SRAM-Based FPGAs for Space Applications," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 37:1–37:34, Jan. 2015.

[3] 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*. EDA Consortium, 2016, pp. 810–813.

[4] 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*, vol. 713. Trans Tech Publ, 2015, pp. 1127–1131.

[5] 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*. IEEE, 2012, pp. 115–120.

[6] 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*, vol. 4, no. 5, pp. 522–526, 2012.

[7] 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*. IEEE, 2007, pp. 105–113.

[8] 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*, vol. 54, no. 1, pp. 252–261, 2007.

[9] 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*, vol. 52, pp. 145–159, 2017.

[10] 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*. IEEE Computer Society, 2011, pp. 223–230.

[11] A. Meyer, *Principles of Functional Verification*. Elsevier Science, 2003. [Online]. Available: http://books.google.cz/books?id=qaIiX3hYWL4C

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

[13] B. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for Multi-robot and Distributed Sensor Systems," in *Proceedings of the 11th international conference on advanced robotics*, vol. 1, 2003, pp. 317–323.

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