Functional Verification as a Tool for Monitoring Impact of Faults in SRAM-based FPGAs

Jakub Podivinsky, Ondrej Cekan, Jakub Lojda, Zdenek Kotasek
Faculty of Information Technology, Brno University of Technology
Bozetechnova 2, 612 66 Brno, Czech Republic
Email: {ipodivinsky, icekan, ilojda, kotasek}@fit.vutbr.cz

***Abstract*—The aim of this paper is to present a new platform for evaluating impact of faults on electro-mechanical systems based on SRAM-based FPGAs. Functional verification together with the fault injector serve as a tool for the fault tolerance evaluation. The article demonstrates the use of the verification environment for evaluating impacts of faults in electro-mechanical systems. Our system consists of mechanical robot and its electronic controller implemented into FPGA. The experimental results gained from the verification process are also presented and discussed in the paper.**

Keywords*—*Functional Verification, Robot Controller, Electro-mechanical Systems, Fault Tolerance, Maze.

I. Introduction

Digital systems play an important role in our lives. They are used in industry, medicine and other safety critical sectors. Not only the loss of a huge amount of money, but also the loss of human lives may occur in case of their failure. The current trend is that the complexity rises, which leads to an increased susceptibility to faults. The approach called *Fault tolerance* [1] is the ability of a system to continue performing its correct function even in the presence of unexpected faults. There have been many fault-tolerant methodologies inclined, among others, to *Field Programmable Gate Arrays* (FPGAs) developed and new ones are under investigation [2], because FPGAs are becoming more popular due to their flexibility and re-configurability. The second reason why so many techniques are inclined to FPGAs is their sensitivity to faults and ability to be reconfigured in the case of fault occurrence. The configuration of FPGAs is stored as a *bitstream* in SRAM memory. The problem is that FPGAs are quite sensitive to faults caused by charged particles. This particle can induce inversion of a bit in bitstream and this may lead to a change in its behaviour. This event is called *Single Event Upset* (SEU).

It is important to evaluate these techniques. Various approaches to the evaluation of fault tolerance exist, some of them are performed on a theoretical level, for example, a simulation method for SEU emulation is presented in [3]. Another approach is in the use of fault injection directly to the design implemented in FPGA. Special evaluation boards are developed for these purposes, one is presented in [4]. The systems implemented as fault-tolerant often consist of two parts - an electronic one and a mechanical one. The mechanical part is controlled by its electronic controller. It can be stated that such areas exist in which electro-mechanical applications are implemented as fault-tolerant - aerospace and space applications can serve as an example. We feel that for electro-mechanical systems it must be possible to check the reactions of the mechanical component if the functionality of its electronic controller is corrupted.

The basic concepts of our evaluation platform were presented in [5]. Based on previous experiments we found functional verification [6] as an appropriate technique for evaluation impact of faults. Functional verification checks whether a hardware system satisfies a given specification. The main principle of functional verification is to compare the outputs of verified circuits with those of the reference model. Different coverage metrics are defined in order to assess that the design has been adequately exercised. These include *code coverage* which gives information about how many lines and how many times expressions and branches are executed.

Our experimental platform (see Figure 1) is composed of a few components running on a computer or on an FPGA evaluation board: 1) software part of verification environment running on computer, 2) software simulation environment for robot simulation (Player/Stage) running on computer, 3) robot controller implemented to FPGA, and 4) external fault injector [7] running on a computer. The connection between a computer and an FPGA is realized by JTAG and Ethernet. JTAG interface is used for fault injection and FPGA programming. The software and hardware part of verification environment are connected through Ethernet.

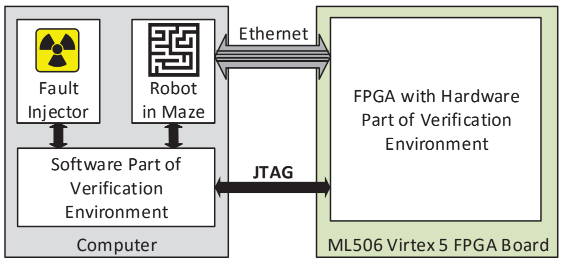
Fault Injector
Robot in Maze
Software Part of Verification Environment
Computer
Ethernet
JTAG
FPGA with Hardware Part of Verification Environment
ML506 Virtex 5 FPGA Board

Fig. 1. The structure of the experimental platform.

The fault injector [7] which allows us to simulate real faults in FPGA is based on the SEU generation outside of the FPGA (in PC), so it is not targeted to a specific FPGA board. The process of the SEU generation is divided into four steps: 1) specifying the location of the fault injection, 2) reading the related part of the configuration bitstream, 3) the SEU generation (i.e. the inversion of the specified bit of the bitstream), and 4) applying the bitstream using *Partial Dynamic Reconfiguration* (PDR) without stopping the FPGA.

The process of the fault impact evaluation is divided into three phases. In the *first phase*, we use the simulation-based functional verification where the VHDL description of the electronic robot controller is verified. In this phase, testing whether the robot controller works correctly according to the specification is done. In this phase we acquire a set of verification scenarios (different mazes with different start and goal positions) that will be used in the subsequent phase. The *second phase* consists of the verification of the robot controller implemented into FPGA with the scenarios obtained during the previous phase and uses a previously implemented fault injector. The analysis of the faults which corrupted the mechanical part is the goal of the *third phase*. The outputs are evaluated verification scenarios supplemented by information about injected faults and its impact on the electronical and mechanical part. Various strategies of fault injection may be used in this phase (e.g. one fault for one verification run or multiple faults in the same functional unit).

II. Experiments and Results

The outputs of the first phase experiments are: 1) the electronic part without bugs (robot controller), 2) the list of the used verification scenarios, and 3) achieved coverage. We used mazes which differ in their dimensions, we chose 7x7, 15x15 and 31x31 cells. With the growing size of the maze the number of steps that the robot must go through increases. Resizing the maze from 7x7 to 15x15 cells led to a slight increase of code coverage. When increasing the size of maze to 31x31 cells, the coverage was not changed. Such studies show that the 15x15 cells maze is the proper size for the next phase of fault impact evaluation process.

For the second phase experiments, fault injection is used. No fault tolerance methodology implemented in the robot controller for these experiments was used and the goals of the experiment are: 1) detailed reliability analysis of the robot controller and its functional units, and 2) demonstration that the evaluation platform can be used for the fault tolerance evaluation. The function of used robot controller which consists of various blocks is described in [8].

We have decided to perform 50 and 100 verification runs and inject one fault into one functional unit (single fault) during one verification run. The robot controller consists of 16 functional units which leads to 1600 evaluation runs. The results of our experiments are shown in Figure 2. The bar chart expresses a percentage number of faults with their impact on the robot and its controller. Faults that cause a failure of electronic, but do not cause any collision of mechanical part are usually leading to the robot stop, which is more safety situation than collision. As can be seen some anomalies in the results of the experiments exist, some of functional units are more prone to the faults than others, but the average number of faults with the impact on electronic part is around 60%. This is especially important for future applications of fault-tolerant methodologies. A system designer obtains the information which blocks need more attention from a reliability point of view. The chart also shows that results ale similar for both 50 and 100 verification runs.

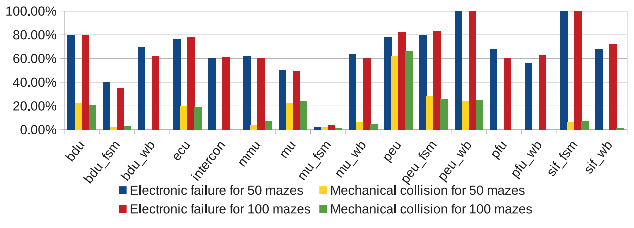

Fig. 2. Impact of faults on electro-mechanical system.

III. Conclusions and Future Research

In this work, we introduced a new evaluation platform for testing fault tolerance methodologies targeted to FPGAs. This platform is composed of several components running on the FPGA and on the computer. The main advantage and novelties of our platform is that we can monitor impact of faults on the electronic part, but also on mechanical part of the experimental electro-mechanical system. The functional verification technique is used as the main tool for monitoring impact of fault. We do not use classical simulation based verification environment, but the evaluated electronic component is running on the FPGA. The performed experiments were also briefly mentioned and described. During the second phase, the reliability analysis was done by means of the fault injection into the FPGA. The result is the ratio of faults that caused an incorrect output of the electronic controller and also number of faults that causes a collision of mechanical part.

The goal of our future work is to apply various fault tolerance methodologies on the robot controller and evaluate them with our evaluation platform. For example, we plan to construct our robot controller as a fault tolerant neural network. We will focus on testing fault tolerance methodologies targeted to FPGAs in the context of electro-mechanical systems and applications which is often the way of using fault-tolerant electronic controllers. As the final result of our research, generally usable principles for testing fault tolerance properties of electromechanical systems will be defined.

Acknowledgement

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, ARTEMIS JU under grant agreement no 621439 (ALMARVI) and BUT project FIT-S-14-2297.

References

[1] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[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] 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.

[4] 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.

[5] J. Podivinsky, O. Cekan, J. Lojda, and Z. Kotasek, “Verification of Robot Controller for Evaluating Impacts of Faults in Electro-mechanical Systems,” in *Digital System Design (DSD), 2016 19th Euromicro Conference on*. IEEE, 2016, pp. 487–494.

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

[7] 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.

[8] 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.