Testing Fault-Tolerance Properties in FPGA based Systems Controlling Electro-mechanical Applications

Jakub Podivínský
Computer Science and Engineering, 1st class, full-time study
Supervisor: Zdeněk Kotásek

Faculty of Information Technology, Brno University of Technology
Božetěchova 2, Brno 612 66

ipodivinsky@fit.vutbr.cz

Abstract. The aim of this paper is to present a new platform for estimating the fault-tolerance quality of electro-mechanical applications based on FPGAs. We demonstrate one working example of such EM application that was evaluated using our platform: the mechanical robot and its electronic controller in an FPGA. In the experiments, the mechanical robot is simulated in the simulation environment, where the effects of faults injected into its controller can be seen. In this way, it is possible to differentiate between the fault that causes the failure of the system and the fault that only decreases the performance.

Keywords. Fault Tolerance, Electro-mechanical Systems, Fault Injection, SEU.

1 Introduction

In several areas, such as aerospace and space applications or automotive safety-critical applications, fault tolerant electro-mechanical (EM) systems are highly desirable. In these systems, the mechanical part is controlled by its electronic controller. Currently, a trend is to add even more electronics into EM systems. For example, in aerospace, extending of the electronic part results in a lower weight that helps reduce the operating cost [1]. It is obvious that the fault-tolerance methodologies are targeted mainly to the electronic components because they perform the actual computation. However, as the electronics can be realized on different hardware platforms (processors, ASICs, FPGAs, etc.), specific fault-tolerance techniques dedicated for these platforms must be developed.

Our research is targeted to *Field Programmable Gate Array*s (FPGAs) as they present many advantages from the industrial point of view. They can compute many problems hundreds times faster than modern processors. Moreover, their reconfigurability allows almost the same flexibility as processors. FPGAs are composed of *Configurable Logic Blocks* (CLBs) that are interconnected by a programmable interconnection net. Every CLB consists of *Look-Up Tables*(LUTs) that realize the logic function, a multiplexer and a flip-flop. The configuration of CLBs and of the interconnection net is stored in the SRAM memory. The problem from the reliability point of view is that FPGAs are quite sensitive to faults caused by charged particles [2]. These particles can induce an inversion of a bit in the configuration SRAM memory of an FPGA and this may leads to a change in its behaviour. This event is called the *Single Event Upset* (SEU). Sensitivity to faults (SEUs) and the possibility of reconfiguration are the main reasons why so many fault-tolerance methodologies inclined to FPGAs have been developed and new ones are under investigation [3].

The paper is organized as follows. The goals of our research and the platform for estimating the quality of EM applications can be found in Section 2. The architecture of our experimental robot controller is provided in Section 3. A description of the fault injection process are described in Section 4. Results of the experiments with the robot controller are available in Section 5. The future work that includes using *functional verification* for automated evaluation of impacts of faults is presented in Section 6. Finally, Section 8 concludes the paper.

2 The Goals of the Research

From the above facts, we have identified two areas that we would like to focus on in our research of fault-tolerant FPGA-based systems controlling electro-mechanical applications.

The first one is that methodologies are validated and demonstrated only on simple electronic circuits implemented in FPGAs. For instance, methodologies focused on the memory in [4] are validated on simple memories without the additional logic around. In [5], the fault-tolerance technique is presented only on a two-input multiplexer, one simple adder and one counter. However, in real systems different types of blocks must be protected against faults at the same time and must communicate with each other. Therefore, a general evaluation platform for testing, analysis and comparison of alone-working or cooperating fault-tolerance methodologies is needed.

As for the second area of the research and the main contribution of our work, we feel that it must be possible to check the reactions of the mechanical part of the system if the functionality of its electronic controller is corrupted by faults. It is either done in simulation or in a physical realization. In our opinion, it is important to find a relation between the level of functional corruption of the electronic controller and the corruption of the mechanical functionality in the EM applications (i.e. between the robot controller and the simulated mechanical robot).

According to the identified problems we have formulated our goal in the following way:

> *To develop an evaluation platform based on the FPGA technology for checking the resilience of EM applications against faults.*

Under the term EM application we understand a mechanical device and its electronic controller implemented in an FPGA. In our experiments, these components are represented by a robot device and its controller, which drives the movement of a robot in a maze. At this point, we wanted to target also the issue of complexity. We have implemented the evaluation platform that consists of three basic parts:

- the Virtex5 FPGA board into which the robot controller is configured,
- the simulation environment for simulating robot and its environment,
- the external fault injector (PC) which inserts faults into the robot controller [6].

3 The Robot Controller - Structure and Principles

In Figure 1, the block diagram of the implemented robot controller is available. The control unit is connected to the PC (where the simulation environment is located) via the Interface Block. Through this block, data from the simulation are received and in the opposite direction, instructions about the movement of the robot are sent back.

The robot controller is composed of various blocks, their function is described in [7]. Here, we only summarize the main characteristics of every component. The central block of the robot controller is a bus through which the communication between each block is accomplished. The Position Evaluation Unit (PEU) calculates position of the robot in the maze and provided them to other units as coordinates x and y. The Barrier Detection Unit (BDU) uses four sensors and provides information about the distance to the surrounding barriers as four-bit vector. Map updating provided by the Map Unit (MU) is based on the information about the position of the robot and the four-bit barriers vector. The Map Memory Unit (MMU) stores the information about the up-to-date map. Path Finding Unit (PFU) implements simple iteration algorithm for finding a path through the maze according to the information about the current and the desired target position. The mechanical parts of the robot are driven by the setting of the speed in the required direction of the movement by the Engine Control Module (ECM).

The robot controller is designed as a complex system with specific components that will allow testing and validating various types of individual or cooperating fault-tolerance methodologies focused on FPGAs. There are combinational circuits, sequential circuits, finite state machines, memories or buses.

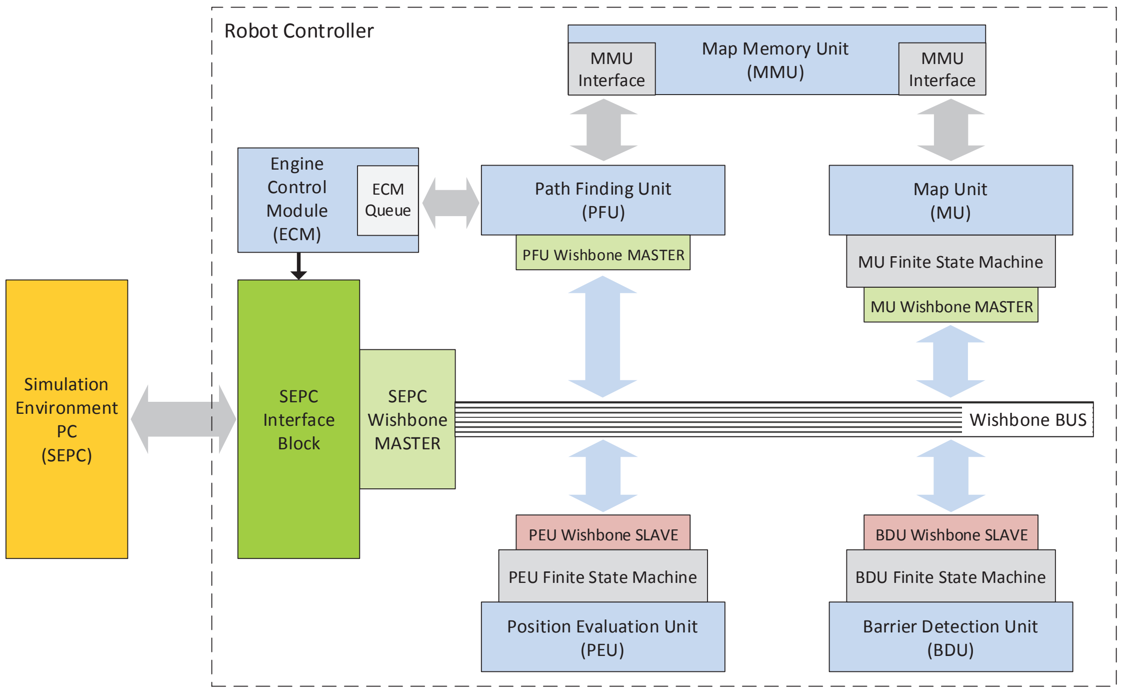

Figure 1: The block diagram of the robot controller.

4 Evaluation of Reliability by Fault Injection

During testing the resilience of systems against faults, waiting for their natural appearance is not feasible. A typical reason is the *Mean Time Between Failures* (MTBF) parameter that can be in the order of years. The most popular techniques to artificially accelerate fault occurrence is called *fault injection*.

Therefore, to simulate the effects of faults in the FPGA, it could be done by a direct change of the configuration bitstream which is loaded into the configuration memory. For this purpose, a fault injector [6] was implemented which allows to modify single or multiple specified bits of the bitstream in order to simulate single and multiple faults.

For effective testing of fault effects on a system composed of several blocks, we need to determine the block in which the fault will be injected. In the case of injecting faults into the whole FPGA we are not sure which block is affected, or if the useful part of the bitstream is hit. The list of bits representing each component can be obtained through several steps by using the PlanAhead [8] tool for the layout of the components on the FPGA. The knowledge about component layout allows us to use the RapidSmith [9] tool for analysing the design. This tool is able to generate a list of the bitstream bits that correspond to the identified areas of the FPGA, while we know what components are configured into particular area. The disadvantage of such approach is that this process provides only a list of bitstream bits that correspond to *Lookup Tables* (LUTs).

5 The Experiment with the Robot Controller

The aim of the experiment is to identify which parts of the robot controller are vulnerable to faults. The flow of the experiment is displayed in Figure 2. At first, we initiate the environment of the robot in simulation. As the first scenario, we chose a small maze with 8x8 fields. Subsequently, the robot controller is initiated. Then the robot starts to search a path to the end position. At this point, the fault injection takes place. We generate randomly an LUT of every unit of the robot controller into which the fault will be injected. Thanks to the Rapidsmith, just the corresponding bits of the bistream are inverted. Faults are injected one after another until the robot starts to behave incorrectly or has an accident. We were monitoring (1) the number of faults that led to the malfunction of the robot and (2) how the behaviour of the robot was changed.

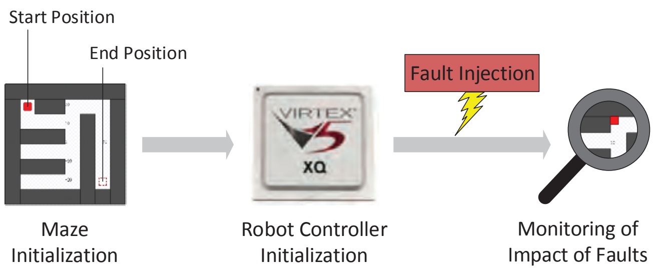

Figure 2: The flow of one experiment.

The results of the experiments are shown in Table 1. In the first column, the list of components of the robot controller is provided. In the second column, the total number of bits of the bitstream that belong to the LUTs of corresponding components is shown. The following three columns represent the number of injected faults into particular components which caused incorrect behaviour of the robot. Injecting faults into all bits of the bitstream would be very time-consuming, because evaluation of faults impact on robot behaviour was monitored manually. Therefore, we utilise the statistic evaluation. 20 experimental runs were performed for each component (320 experimental runs in total). The last column of the table contains the state of the robot that was evaluated as the wrong behaviour.

One interesting conclusion arises from the results. The incorrect behaviour did not appear immediately after the first injection of a fault. We can conclude that some bits of the bitstream, despite they are identified as related to the robot controller, are not used to store a useful information. This can be seen particularly in components PEU_FSM and PEU_WB. Nevertheless, we realised that some components contain more critical bits than others and thus they should be preferred while hardening against faults by some fault-tolerance methods.

The most common consequences of injected faults which are presented in table are Freezing on place, Deadlock, Crashing into a wall and something other. As can be seen from the table, the most common consequence of injected faults is *Freezing on place*. We can also conclude that stopping of the robot is not so critical as for example a collision with the wall. This conclusion can be very critical and useful for different kinds of EM applications.

6 Functional Verification for Automated Evaluation of Fault Impacts

For extensive testing of the behaviour of the robot or any other EM system placed into our evaluation platform, we need to examine various test scenarios. After application of proper test vectors, we can prove the correctness and accuracy of the behaviour of the system with respect to the specification. The manual check of these test vectors is difficult as it requires a full control from the user. The user is responsible for running the test environment, generating test vectors and also analysing the outputs of

Table 1: The experimental results.

Components	Bits of bitstream	Number of injected faults			Consequence
		Min	Median	Max	
PEU	21 632	2	6	12	freezing
PEU_FSM	2 112	>80	-	>80	-
PEU_WB	2 112	41	-	>80	freezing
BDU	320	2	6	21	freezing
BDU_FSM	2 752	3	6	34	freezing
BDU_WB	2 176	3	9	28	freezing
SEPC_INF	1 216	2	3	7	freezing
SEPC_WB	9 088	2	3	7	freezing
ECM	25 664	1	2	7	freezing
PFU	7 488	3	6	12	deadlock
PFU_WB	7 424	2	3	9	freezing
MU	11 840	1	2	3	crashing
MU_FSM	1 280	1	3	5	freezing
MU_WB	7 680	1	3	6	freezing
MMU	3 008	1	3	6	freezing
WB_BUS	5 056	1	3	6	freezing

the system. All these activities are time-demanding and therefore, it is not possible to test the system thoroughly within a reasonable time. It is necessary to apply some kind of automation. An extended technique for automated checking of the correctness of the system is called verification. There are several techniques used in the verification domain. We decided to use an approach called functional verification, as this type of verification fits best to our future experiments.

To be able to inject faults into the FPGA while performing functional verification, we must carry out verification directly in the FPGA (not in the simulation as usually). Advantageously we can use and modify hardware accelerated verification that uses an FPGA as the acceleration board. An example of such accelerator is the framework HAVEN [10]. The DUT (in our case the robot controller) will be placed on the FPGA. The outputs from the FPGA are compared to the outputs of the reference model and they represent also the inputs that are propagated to the simulation of the mechanical part. Thus, the output of the DUT stimulates the movement of the mechanical part of the robot in the simulated maze. The inputs for the FPGA and for the reference model are data from the sensors of the mechanical part of the robot.

7 Goals of the Ph.D. Thesis

In previous text, problems associated with faults in FPGA were presented , in particular those related to the evaluation of the quality of the fault tolerance methodologies. From mentioned findings the goals of the Ph.D. thesis titled *Use of verification for evaluation fault tolerance systems based on FPGAs* arise:

- Create an electromechanical application as an experimental system for testing and validating the fault tolerance methodologies.
- Create a platform for the evaluation quality of fault tolerance methodologies based on the interconnection of two techniques: verification of digital circuits and fault injection.
- The proposition of processes for effective ensuring fault tolerance with using implemented platform.

In this paper, the first version of the platform was presented, now without the use of verification techniques connected with fault injector.

8 Conclusion and Future Work

In this paper, we introduced the evaluation platform for estimating reliability of FPGA designs. As our research focuses on testing EM applications, we presented the experimental design which is composed of the mechanical robot and its electronic controller situated in the FPGA. The robot controller contains a variety of components. During the experiments, random faults were artificially injected into these components and we were monitoring impact of these faults on the behaviour of the robot in the simulation environment. These experiments showed that some faults have an impact on the behaviour of the robot, and others do not have. According to this result we were able to identify the parts/components of the robot controller that need to be hardened by some fault-tolerance techniques.

In addition, we have recognised from the experiments that some kind of automation is unavoidable in our future experiments, especially in the early phases of testing. The reason is that monitoring the behaviour of system in simulation is very time-demanding. Therefore, we have already prepared an innovative extension of our platform - interconnection of fault injection and functional verification environment with advanced test generation. Using this approach we will be able to automatically verify an EM system during the fault injection. The automation is achieved by comparing the outputs of the verified system to the reference model that is in our case represented by the same design but without injected faults.

Acknowledgment

This work was supported by the following projects: BUT project FIT-S-14-2297, National COST LD12036, project IT4Innovations Centre of Excellence (ED1. 1.00/02.0070) and COST Action project ”Manufacturable and Dependable Multicore Architectures at Nanoscale”.

References

[1] S. Cutts, “A collaborative approach to the more electric aircraft,” in *Power Electronics, Machines and Drives, 2002. International Conference on (Conf. Publ. No. 487)*, June 2002, pp. 223–228.

[2] M. Ceschia, M. Violante, M. Reorda, A. Paccagnella, P. Bernardi, M. Rebaudengo, D. Bortolato, M. Bellato, P. Zambolin, and A. Candelori, “Identification and classification of single-event upsets in the configuration memory of SRAM-based FPGAs,” *Nuclear Science, IEEE Transactions on*, vol. 50, no. 6, pp. 2088–2094, 2003.

[3] L. Sterpone, M. Aguirre, J. Tombs, and H. Guzmán-Miranda, “On the Design of Tunable Fault Tolerant Circuits on SRAM-based FPGAs for Safety Critical Applications,” in *DATE ’08: Proceedings of the conference on Design, automation and test in Europe*. New York, NY, USA: ACM, 2008, pp. 336–341.

[4] N. Rollins, M. Fuller, and M. Wirthlin, “A comparison of fault-tolerant memories in sram-based fpgas,” in *Aerospace Conference, 2010 IEEE*, 2010, pp. 1–12.

[5] M. Naseer, P. Sharma, and R. Kshirsagar, “Fault tolerance in fpga architecture using hardware controller - a design approach,” in *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom ’09. International Conference on*, 2009, pp. 906–908.

[6] 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.

[7] J. Podivinsky, M. Simkova, and Z. Kotasek, “Complex Control System for Testing Fault-Tolerance Methodologies,” in *Proceedings of The Third Workshop on Manufacturable and Dependable Multicore Architectures at Nanoscale (MEDIAN 2014)*. COST, European Cooperation in Science and Technology, 2014, pp. 24–27.

[8] N. Dorairaj, E. Shiflet, and M. Goosman, “Planahead software as a platform for partial reconfiguration,” *Xcell Journal*, vol. 55, no. 68-71, p. 84, 2005.

[9] C. Lavin, M. Padilla, P. Lundrigan, B. Nelson, and B. Hutchings, “Rapid prototyping tools for fpga designs: Rapidsmith,” in *Field-Programmable Technology (FPT), 2010 International Conference on*, Dec 2010, pp. 353–356.

[10] M. Simkova, O. Lengal, and M. Kajan, “Haven: An open framework for fpga-accelerated functional verification of hardware,” Tech. Rep., 2011. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php.en?id=9739