A Complex Control System for Testing Fault-Tolerance Methodologies

Jakub Podivinsky · Marcela Simkova · Zdenek Kotasek

Abstract The aim of this paper is to present a new platform for estimating the fault-tolerance quality of electro-mechanical (EM) applications based on FPGAs. We demonstrate one working example of such EM application: the mechanical robot and its electronic controller in an FPGA. Different building blocks of the electronic robot controller design allow to model different effects of faults on the whole mission of the robot (searching a path in a maze). The mechanical robot is simulated in the Player/Stage 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 its performance.

Keywords FPGA · robot controller · fault-tolerance · electro-mechanical application

1 Introduction

Field Programmable Gate Arrays (FPGAs) present many advantages from the industrial point of view because they can compute many problems hundreds times faster than modern processors while their reconfigurability allows almost the same flexibility as processors.

The problem is that FPGAs are quite sensitive to faults caused by charged particles [1]. These particles can induce an inversion of a bit in the configuration memory of an FPGA and this may lead to a change in its behavior. This event is called the *Single Event Upset* (SEU). Sensitivity to these faults is one of the reasons why so many fault-tolerance methodologies inclined to FPGAs have been developed and new ones are under investigation [2],[6].

Nevertheless, we have identified two areas in the research of fault-tolerant FPGA-based systems that should be improved or at least explored in more detail.

The first one is that methodologies are validated and demonstrated only on simple electronic circuits implemented in FPGAs. For instance, methodologies that focus on simple memories in [5] are validated without the additional logic around. In [4], the fault-tolerance technique is presented only on a two-input multiplexer, one simple adder and one counter. Other methodology dedicated to harden finite state machines [3] is applied only on a simple finite state machine.

Of course, for the demonstration purpose such circuits are satisfactory. However, in real FPGA systems, different types of blocks must be protected against faults simultaneously and must communicate with each other. Therefore, a new evaluation platform for testing, analysis and comparison of more complex systems is needed, where alone-working or cooperating fault-tolerance methodologies can be tested.

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 can be either done in simulation or in a physical realisation.

The paper is organised as follows. The interconnection scheme of our platform can be found in Section 2. The architecture of the robot controller is described in Section 3. Section 4 presents ideas about the future work and Section 5 concludes the paper.

2 The Evaluation Platform

On the basis of the identified problems we have developed a new evaluation platform for checking the resilience of complex electro-mechanical applications aga-

FIT, Brno University of Technology, Czech Republic
Tel.: +420 54114-1361, Fax: +420 54114-1270
E-mail: {ipodivinsky,isimkova,kotasek}@fit.vutbr.cz

inst faults. Under the term electro-mechanical application we understand a mechanical device and its electronic controller implemented in an FPGA. In our working example, they are represented by a robot device and its controller.

The evaluation platform is composed of three basic components:

- the Virtex5 FPGA board, where the robot controller is implemented,
- the simulation environment Player/Stage for checking responses of the mechanical device to instructions from the robot controller (Figure 1),
- the external fault injector (PC) which inserts faults into the robot controller [7].

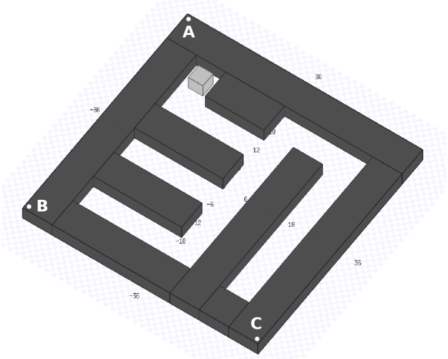


Fig. 1 The robot in a maze.

Figure 2 shows the overall connection of the PC and the FPGA board in our platform. Note that there are two FITkits [8]in both directions, from the PC to the FPGA and vice versa. FITkits represent a communication layer and serve as a debugging point for communication between the PC and the FPGA board. The SEU injector runs on the PC and is connected through the JTAG interface directly to the main FPGA board where the robot controller is situated. Via the connection between the SEU injector and the simulation environment (as shown in Figure 2), we are able to control the SEU injection process into the robot controller for every mission and to see effects of faults directly in simulation.

3 Design of the Robot Controller

The proposed electronic controller was constructed as an exemplary FPGA system that includes various aspects of the digital system design (contains sequential and combinational circuits, a bus, different types of memories, etc.). We wish to point out, that the controller will be used for the demonstration and testing of various methodologies aimed at these specific components and on the whole, for designing fault-tolerant systems based on the FPGA technology.

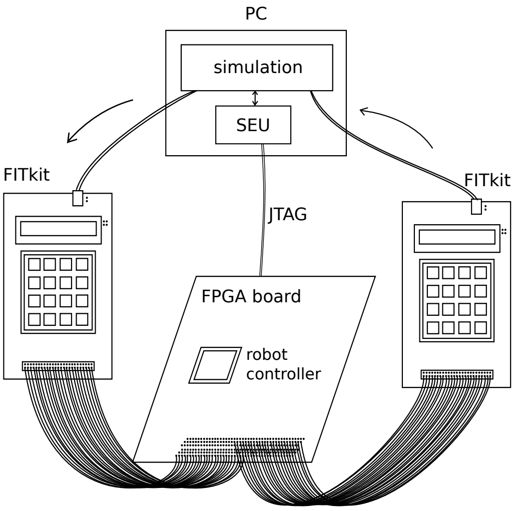


Fig. 2 The interconnection scheme.

The Structure of the Controller

Figure 3 shows the block diagram of the robot controller. The control unit is connected to the remote simulation environment PC (SEPC) via the SEPC Interface Block. Through this block, data from SEPC are received (information about barriers, distances from control points, target positions) and in the other direction, instructions about the movement of the robot are sent (direction and speed).

Position Evaluation Unit (PEU) acquires the distance from the control points, which are located in the fixed positions in the maze. From these, the position of the robot in the maze is calculated and provided to other units as coordinates x and y. The situation is shown in Figure 4, where you can see the position of the robot (APP_X, APP_Y) which is calculated from the distances from the three control points A, B and C (DIST_A, DIST_B, DIST_C).

The position of the robot can be calculated using the Pythagorean Theorem. One can deduce that this block utilises the sequential arithmetic operations such as addition, multiplication, and division. It is an example of the sequential circuit formed by combinational logic and registers, both of which are very interesting from the fault-tolerance point of view.

Barrier Detection Unit (BDU) uses four sensors, each located on one side of the robot (cubical robot) and provides information about the distance to the surrounding barriers. The output is a four-bit vector that represents the four-neighbourhood of the robot and informs about barriers in this area (0 = no barrier, 1 = a barrier). The purpose of this unit consists of comparing the barrier distance with the reference value to determine whether the barrier is located at the neigh-

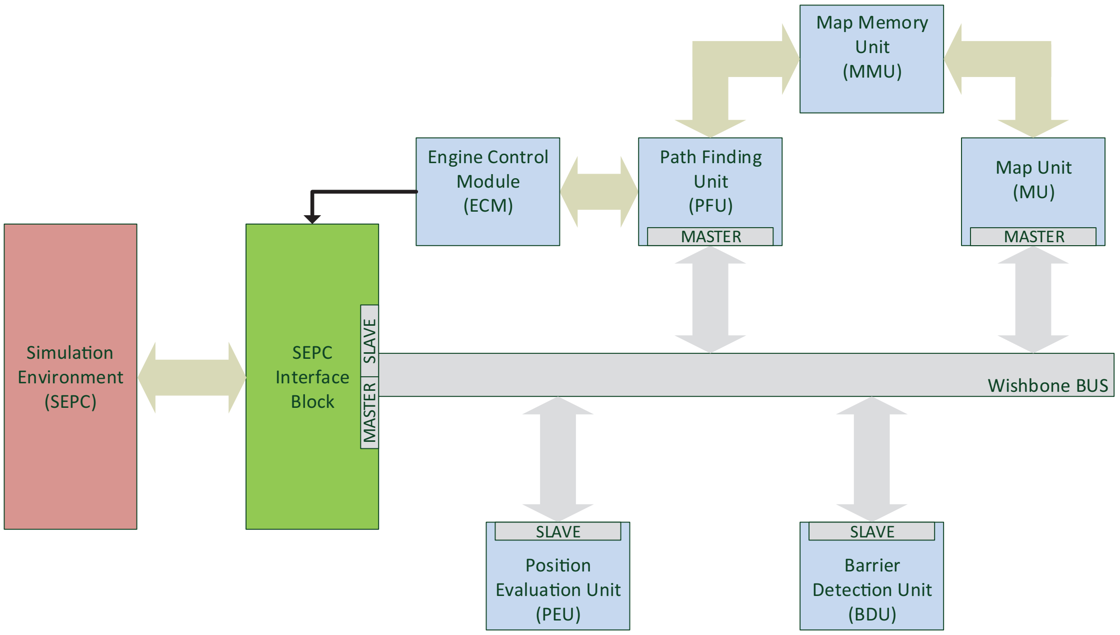


Fig. 3 The block diagram of the robot controller.

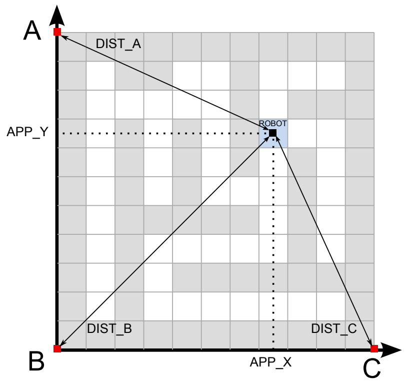


Fig. 4 The position of the robot in the maze.

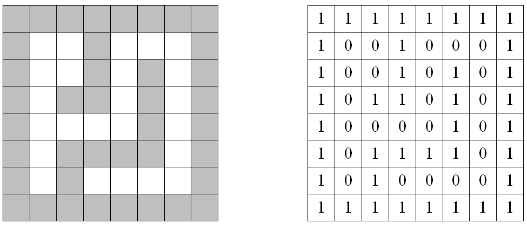
bour coordinate or outside the four-neighbourhood. It represents a pure combinational circuit.

Map updating provided by the *Map Unit* (MU) is based on the information about the position of the robot obtained from the PEU and the information about the occurrence of barriers in a four-neighbourhood provided by BDU.

This unit transforms the inputs (position, barriers) into the relevant addresses of five memory locations (four neighbourhood and robot position) and sets their values according to the presence of barriers.

Map Memory Unit (MMU) stores information about the up-to-date map. Each coordinate in the map is represented by a four-bit vector. The first bit informs about the presence of barriers on this position (see Figure 5), the second bit holds information about the up-to-datedness of the stored data and other two bits are reserved for the future extension. MMU is realised by the block memory (BRAM) available in the FPGA.

Memories (without any fault tolerance mechanism) are in general quite susceptible to faults and the memory used in the robot controller allows us to test specific fault-tolerance methodologies related to memories, e. g. error correction/detection codes, parity or redundancy. From the perspective of fault-tolerance, also the connection between the memory and other system components requires increased attention.



1	1	1	1	1	1	1	1
1	0	0	1	0	0	0	1
1	0	0	1	0	1	0	1
1	0	1	1	0	1	0	1
1	0	0	0	0	1	0	1
1	0	1	1	1	1	0	1
1	0	1	0	0	0	0	1
1	1	1	1	1	1	1	1

Fig. 5 An example of a map and its bit representation.

The most important block that manages the activity of other blocks in the robot controller is *Path Finding Unit* (PFU). PFU implements the simple iteration algorithm for finding a path through the maze according to the information about the current and the desired target position. PFU has a direct access to the memory where the information about barriers is stored. The output of this block is a list of positions that the robot has to route along in the next step. With the movement of the robot the map is updated by MU.

PFU is a complex block that implements a sequential algorithm. During the evaluation of the path searching algorithm, an extensive communication through the bus and with the memory is performed. This involves some specific steps in ensuring fault-tolerance. Alternatively, it could be interesting to implement the path searching algorithm in the processor embedded into the FPGA, such as PicoBlaze or MicroBlaze, which are provided by Xilinx. This step would require additional mechanisms, which target other specific types of faults.

The mechanical robot is driven by the setting of the speed in the required direction of the movement by the *Engine Control Module* (ECM). The distance the robot should travel determines the time during which the robot is in motion. The list of the fields in the maze that the robot needs to route along is the input of ECU. According to this list, the speed and the direction of the movement of the robot are continuously adjusted. Fault resilience of this block is very important as it directly controls the robot and an accidental fault may cause an incorrect management or even a collision.

The central block of the robot controller is a *bus* through which the communication to other blocks is accomplished. Using the bus can facilitate the future extension of the robot controller by additional units, e.g. an advanced path searching algorithm. Figure 3 shows how each block is related to the bus, if it is master, slave, or master and slave at the same time. From the fault-tolerance perspective, the bus is a very critical block as it is located in the centre of the whole system and its failure will undermine the correctness of every traffic flow going through it. The use of the bus in the robot controller is convenient not only from the perspective of future extensibility but also the possibility of a suitable methodology for testing bus resistance.

4 The Future Work

In our future experimental work we plan to focus on testing, analysis and comparison of different fault tolerance methodologies applied to the electronic part of the system (the robot controller). Simultaneously, we will monitor the behaviour of its mechanical counterpart (the robot device) in the simulation environment after the faults are injected into the controller.

In the first experiment we will inject faults into every bit of the bitstream (configuration data for the FPGA) which corresponds to the robot controller. No fault-tolerance mechanisms will be applied to the controller at this stage. After the bit-flip of one bit of the bitstream, the FPGA will be reconfigured. The aim of this experiment is to check if such bit-flip has or has not the impact to the functionality of the robot controller and if the robot is still able to fulfil its mission. According to this experiment we will be able to identify the most critical parts of the robot controller.

In the following experiments we will incrementally harden the critical parts of the controller against faults. We plan to apply different methodologies to specific logic blocks and to repeat the procedure described in the first experiment. Thanks to the simulation we will see if the fault-tolerance mechanisms increase the reliability of the robot or not. Other possible use case is the selection of proper mechanisms with respect to consumed resources, performance, or power consumption.

5 Conclusions

This paper introduced a new evaluation platform for analysis and comparison of different fault-tolerance methodologies targeted to electro-mechanical FPGA systems. The platform consists of three main components: the FPGA board with the electronic benchmark circuit (the robot controller) and the PC with the simulated mechanical counterpart (the robot) and the SEU injector. The main contribution of this work is in designing the robot controller in the way it allows to apply different kinds of fault-tolerance methodologies targeted to specific design components. Thanks to the simulation environment we can receive a visual feedback about the movements of the robot in the maze and therefore, to directly see the effects of faults on the behaviour of the robot and its outputs.

Acknowledgements This work was supported by the following projects: National COST LD12036, research project No.MSM 0021630528, project IT4Innovations Centre of Excellence (ED1.1.00/02.0070), COST Action project MEDIAN, and internal grant "FIT-S-11-1".

References

1. Ceschia, M., Violante, M., Reorda, M., Paccagnella, A., Bernardi, P., Rebaudengo, M., Bortolato, D., Bellato, M., Zambolin, P., Candelori, A.: Identification and classification of single-event upsets in the configuration memory of SRAM-based FPGAs. Nuclear Science, IEEE Transactions on **50**(6), 2088–2094 (2003)
2. Cheatham, J.A., Emmert, J.M., Baumgart, S.: A Survey of Fault Tolerant Methodologies for FPGAs. pp. 501–533. ACM, New York, NY, USA (2006)
3. Frigerio, L., Salice, F.: RAM-based fault tolerant state machines for FPGAs. In: DFT '07., pp. 312–320 (2007)
4. Naseer, M., Sharma, P., Kshirsagar, R.: Fault tolerance in FPGA architecture using hardware controller - a design approach. In: ARTCom '09., pp. 906–908 (2009)
5. Rollins, N., Fuller, M., Wirthlin, M.: A comparison of fault-tolerant memories in SRAM-based FPGAs. In: Aerospace Conference, 2010 IEEE, pp. 1–12 (2010)
6. Sterpone, L., Aguirre, M., Tombs, J., Guzmán-Miranda, H.: On the Design of Tunable Fault Tolerant Circuits on SRAM-based FPGAs for Safety Critical Applications. In: DATE '08, pp. 336–341. ACM, New York, NY, USA (2008)
7. Straka, M., Kastil, J., Kotasek, Z.: SEU simulation framework for xilinx FPGA: First step towards testing fault tolerant systems. In: 14th EUROMICRO DSD, pp. 223–230. IEEE Computer Society (2011)
8. Vasicek, Z.: FITkit (2013). URL www.fit.vutbr.cz/FITkit