Microprocessors and Microsystems 52 (2017) 145–159

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Functional verification based platform for evaluating fault tolerance properties

Jakub Podivinsky*, Ondrej Cekan, Jakub Lojda, Marcela Zachariasova, Martin Krcma, Zdenek Kotasek

Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Bozetechova 2, 612 66 Brno, Czech Republic

ARTICLE INFO

Article history:
Received 9 January 2017
Revised 15 March 2017
Accepted 4 June 2017

Keywords:
Functional verification
Robot controller
Electro-mechanical systems
Fault tolerance
Maze generation

ABSTRACT

The fundamental topic of this article is the interconnection of simulation-based functional verification, which is standardly used for removing design errors from simulated hardware systems, with fault-tolerant mechanisms that serve for hardening electro-mechanical FPGA SRAM-based systems against faults. For this purpose, an evaluation platform that connects these two approaches was designed and tested for one particular casestudy: a robot that moves through a maze (its electronic part is the robot controller and the mechanical part is the robot itself). However, in order to make the evaluation platform generally applicable for various electro-mechanical systems, several subtopics and sub-problems need to solved. For example, the electronic controller can have several representations (hard-coded, processor based, neural-network based) and for each option, extendability of verification environment must be possible. Furthermore, in order to check complex behavior of verified systems, different verification scenarios must be prepared and this is the role of random generators or effective regression tests scenarios. Also, despite the transfer of the controller to the SRAM-based FPGA which was solved together with an injection of artificial faults, many more experiments must be done in order to create a sufficient fault-tolerant methodology that indicates how a general electronic controller can be hardened against faults by different fault-tolerant mechanisms in order to make it reliable enough in the real environment. All these additional topics are presented in this article together with some side experiments that led to their integration into the evaluation platform.

1. Introduction

Digital systems play an important role in our everyday lives. They are widely 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 of digital systems is rising, which leads to an increased susceptibility to faults. It is possible to specify two main sources of faults [1]: 1) *Design faults* (bugs) are always the consequence of an incorrect design, an ambiguous specification or misinterpretation of the specification and 2) *Hardware/physical faults* (defects) which arise during manufacturing or system operation.

The approach dealing with *design faults* is called *functional verification* [2] which currently has an irreplaceable position in the development cycle of digital systems. It runs in a simulation (RTL - *Register-Transfer Level* simulators are typically used, like QuestaSim from Mentor Graphics or VCS from Synopsys) and uses sophisticated testbenches which are prepared according to UVM (Universal Verification Methodology) [3,4] which ensures scalability and re-usability. Functional verification checks whether a hardware system satisfies a given specification. The main purpose is to find as many design faults as possible before the system is deployed. The main principle of functional verification is to apply a huge number of input stimuli to the input ports of the verified circuit (DUT - *Device Under Test*) and on the input ports of the reference model. Afterwards, the behavior of DUT and the reference model is compared for these stimuli. The reference model is prepared by a verification engineer in SystemVerilog, C/C++ or other supported language and implements the reference behavior.

Coverage is an important metric in verification. It measures how well input stimuli cover the behavior of DUT and provides feedback that determines when the verification process can be ended. Depending on the coverage criterion considered, the following *coverage* metrics can serve as an example:

* Corresponding author.
E-mail addresses: ipodivinsky@fit.vutbr.cz (J. Podivinsky), icekan@fit.vutbr.cz (O. Cekan), ilojda@fit.vutbr.cz (J. Lojda), zachariasova@fit.vutbr.cz (M. Zachariasova), ikrcma@fit.vutbr.cz (M. Krcma), kotasek@fit.vutbr.cz (Z. Kotasek).

http://dx.doi.org/10.1016/j.micpro.2017.06.004
0141-9331/

- *Code coverage* measures how well input stimuli cover the source code of DUT. Typical code coverage metrics are toggle, statement, branch, condition, expression or FSM coverage.
- *Functional coverage* measures how well input stimuli cover the functional specification of DUT. The user defines the coverage points for the functions to be covered in a verified circuit, e.g.: Did the verification test cover all possible commands or did the simulation trigger a buffer overflow?

Moreover, standard languages, methodologies and libraries were defined for functional verification. The most commonly known ones are the SystemVerilog IEEE language standard [5], Universal Verification Methodology and the open-source UVM library (with all the basic components of verification environments).

Of course, UVM-based functional verification does not guarantee 100% correctness of the system as formal verification does. The reason is that formal verification is based on an exhaustive exploration of the state space of DUT, hence it is potentially able to formally prove its correctness. However, the main disadvantage of this method is a state space explosion for real-world systems and the need to provide formal specifications of the behavior of the system which makes this method often hard to use. On the other hand, UVM-based functional verification is much easier to use and aims at covering properties determined by the specification, not the whole state space. Nevertheless, if these properties are selected accurately, all key aspects of the system are properly verified.

The approaches which deal with *hardware/physical* faults are techniques called *Fault avoidance* or *Fault tolerance* [6]. *Fault avoidance* is mainly obtained by the use of radiation hardened technologies, improved design of storage elements or asynchronous circuits. *Fault tolerance* is the ability of a system to continue performing its correct function even in the presence of unexpected faults. Many fault-tolerant methodologies have been developed inclined, among others, to *Field Programmable Gate Arrays* (FPGAs) and new ones are under investigation [7], because FPGAs are becoming more popular due to their flexibility and reconfigurability. 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. FPGAs are composed of configurable logic blocks [8] which are connected by programmable interconnections. The configuration is stored as a *bitstream* in SRAM memory. The problem is that FPGAs are quite sensitive to faults caused by charged particles [9]. This particle can induce an inversion of a bit in the bitstream and this may lead to a change in its behaviour. This event is called *Single Event Upset* (SEU).

It is important to test and evaluate these techniques. Various approaches to the evaluation of fault tolerance exist and some of them are performed on a theoretical level, for example, a simulation method for SEU emulation is presented in [10]. Another approach is in the use of fault injection directly into the design implemented in FPGA. Special evaluation boards are developed for these purposes, one of them is presented in [11] or [12].

The systems implemented as fault-tolerant very 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. Until now, our work was dedicated to verification of fault-tolerant qualities that allow us to check just the resilience of electronic components. However, for electro-mechanical systems, the approach must be different. It must be possible to check what are the reactions of the mechanical component if the functionality of its electronic controller is corrupted by external attacks.

This paper is organized as follows. The goals of our research are described in Section 2. Section 3 introduces three phases of the evaluation process based on our platform. We focus on introducing every phase theoretically and at the same time, we elaborate on making the platform general for various electro-mechanical systems. The first phase is mentioned in Section 4 together with verification environment architecture. Different possibilities for implementing the electronic controller (DUT) are mentioned in Section 5. This can be considered as the first step to generalization. The second step is preparing various verification scenarios for different DUTs and this process is summarized in Section 6. FPGA-based verification environment which is needed for the second phase is presented in Section 7. Principles which are used for checking reactions of the mechanical part in the third phase are introduced in Section 8. For the demonstration of our evaluation platform we created a case-study presented in Section 9 which is supplemented by experiments and their results in Section 10. Section 11 summarizes the results and proposes our plans for future research.

2. Goals of the research

Based on our previous analysis of actual research in the area of fault tolerance methodologies and their evaluation, we have identified the main goals that we would like to focus on in our research of fault-tolerant FPGA-based systems.

- The first point is to develop an *evaluation platform based on FPGA technology for testing fault tolerance techniques*. The basic concepts and the first version of the evaluation platform were presented in our previous work [13]. Based on experiments with our platform we realized the necessity to automate the process of a fault impact evaluation. We found functional verification as an appropriate technique for this purpose.
- The important task is to propose the *process describing the use of the developed evaluation platform for fault tolerance properties improvement* in general electro-mechanical systems. It means that our evaluation platform will be supplemented with a description on how to configure the environment for the selected experimental system, especially how to evaluate fault tolerance properties and search for the possibilities of its improvement.
- As was mentioned above, we need to *take into account the mechanical part* which is usually driven by an electronic controller in real systems. Therefore, the verification environment should take into account also the operation of the mechanical part when evaluating the correctness of operations.

The following sections describe our progress in achieving these goals. Firstly, the basic concept of the evaluation process is described and is divided into three phases. Each of these phases needs a specific verification environment with a specific configuration, so the evaluation platform is described on a theoretical level for every phase separately. The evaluation platform is demonstrated on one case-study: a robot searching a path through a maze and its electronic controller.

3. Basic concept of the evaluation process

The proposed process of the fault impact evaluation, which is shown in Fig. 1, is divided into three phases. In the first phase, we use the simulation-based functional verification where the VHDL description of the electronic controller is used as the DUT. In this phase, the correctness of the electronic controller design is evaluated. The main output of the first phase is a test on whether the electronic controller works correctly according to the specification. It is important because we have to ensure that the electronic controller does not contain any functional errors in the implementation. It is also important to point out that in this phase we acquire