SEU Simulation Framework for Xilinx FPGA: First Step Towards Testing Fault Tolerant Systems

Martin Straka, Jan Kastil, Zdenek Kotasek
Brno University of Technology
Faculty of Information Technology
Bozetechova 2, Brno, 612 66, Czech Republic
Email: (strakam, ikastil, kotasek)@fit.vutbr.cz

Keywords: SEU, generator, framework, FPGA, fault tolerant system, partial reconfiguration, testing, simulation

***Abstract*—In the paper, the SEU simulation framework for testing fault tolerant system designs implemented into FPGA is presented. The framework is based on SEU generation outside FPGA (in personal computer) and the transport of modified bitstream through the JTAG interface and subsequent dynamic reconfiguration of FPGA. It allows to select region of the FPGA for SEU placing. The SEU simulator does not require any changes in the tested design and is fully independent on the function implemented into FPGA. The requirements on the SEU generator and its properties are described in the paper as well. The external SEU generator for Xilinx FPGA was implemented and verified on evaluation board ML506 with Virtex5 for different types of RTL circuits and fault tolerant architectures. The experimental results demonstrated the effectiveness of the methodology.**

I. Introduction

Recent developments of the microelectronics allow to build complex system on chip into reconfigurable architectures such as Field Programmable Gate Arrays (FPGA). The use of the FPGA presents many advantages from the point of the industry because FPGA can compute many problems hundreds times faster than modern processors while their reconfigurability allows almost the same flexibility as processors [1]. Moreover, the power consumption of such FPGA is much lower than the power consumptions of the processors because FPGA operates on the lower frequency. Due to its reconfigurability, FPGA allows to specify the function of the device after the manufacturing process which can further reduce the cost of the solution because several types of the same device can be built in the same assembly line.

The use of the FPGA will probably increase in the future with the expansion of the Partial Dynamic Reconfiguration (PDR). PDR allows the user to change part of the functionality in the FPGA without stopping the computational process. The PDR can be used in many ways but the main idea behind it is that requirements on systems change in time as user wants to use different functionality of the system [2]. Therefore it is not required to have all functionality implemented at once as long as we can switch the functionality in the device according to user needs. As the result system can be implemented in the smaller chip with lower power consumption.

The downside of FPGAs came with the reconfiguration memory. The functionality of the chip is defined by the sequence of bits in the configuration memory. Most of the modern FPGA has SRAM configuration memory. Even smallest change in the configuration memory can lead to different functionality implemented in the chip. Since FPGAs can be used in highly reliable systems, such as space application or car control, such changes can result in tragic consequences.

In a SRAM-based FPGA, the combinational and sequential logic are implemented in programmable complex logic blocks (CLBs), which are customized by loading configuration data (bitstream) in SRAM cells of the program memory. When a charged particle strikes a memory cell in the program memory, the effect can produce an inversion in the stored value - this can modify the function of design, (as a result of Single Event Upset - SEU) [3]. Recently, SEU effects on SRAM-based FPGA resources have been a field of research [4],[5].

FPGA based designs offer new possibilities for the activities which aim at designing Fault Tolerant Systems (FTS) with high reliability and availability of the system. The main problems combined with the modern FT systems include error detection caused by SEU during system operation, fast fault location, quick recovery or repair from fault and bringing the system back to the state in which it operates correctly [6].

In FPGA, a faulty module can be repaired by reconfiguring the chip. In this situation, the function mode of the system is interrupted and reconfiguration process of FPGA chip is initiated. For this purpose, the principles of PDR can be used where reconfiguration process is applied on the faulty module without interrupting the system. This type of fault repair during system runtime is supported by hardware redundancy architectures, such as Triple Modular Redundancy (TMR) or duplex system with Concurrent Error Detection (CED) techniques [7],[8].

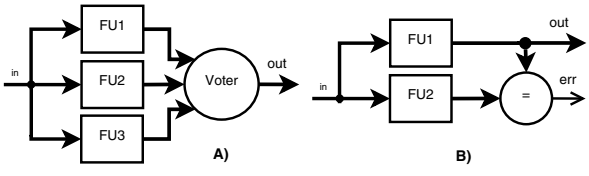

Fig. 1. FT architectures: A) TMR system, B) Duplex system

978-0-7695-4494-6/11 $26.00 © 2011 IEEE
DOI 10.1109/DSD.2011.32

TMR uses hardware redundancy to mask any single design failure by voting on the result of three identical copies of the circuit (see in Figure 1A). Duplication of system ensures CED and is a popular technique used in many FT schemes. It requires duplication of a functional module, a comparator and a control path to propagate the error signal to various parts of the system (see in Figure 1B).

Many methodologies to build reliable or even FT systems in FPGA is presented in the literature while new ones are still being developed [9],[10]. However, to build reliable FTS a testing tool is needed to test developed system by producing or simulating errors by changing the configuration memory [11].

The main contribution of this work is to present such tool together with the discussion about possible problems combined with the simulation of soft errors as SEUs in FPGAs.

II. Motivation and Goals of the Research

Today, the robustness and complexity of various systems have a significant impact on testability and diagnostic features of these systems. For some types of FPGAs, techniques exist which allow to detect and repair any soft errors (SEUs) in design implemented into FPGA by bitstream scrubbing with TMR.

A. Previous Research in Fault Tolerant System Design

Our previous research was oriented to creating a methodology which allows to construct on-line checkers for components on different levels, e.g. module or Register-Transfer Level (RTL) components. An on-line checker can be used for error detection and fault localization in the faulty module or for identification of faulty units in the FT architecture (e.g. duplex). The principle of methodology was presented in [12].

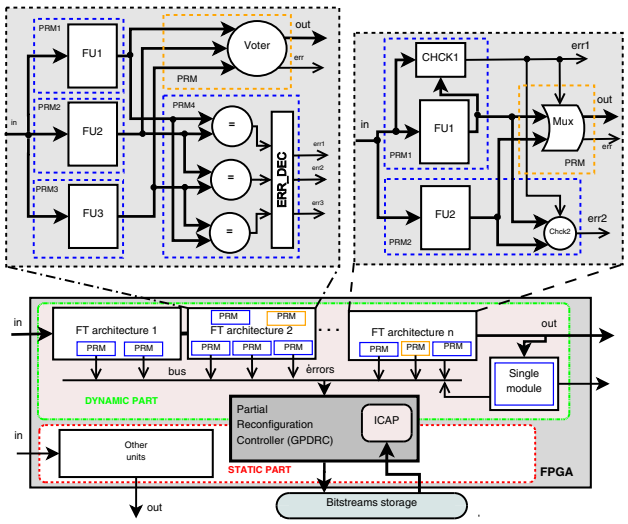

Fig. 2. Fault Tolerant Structure for SRAM-based FPGA via Partial Dynamic Reconfiguration Using GPDRC

The activities which aim at defining a methodology of FT systems design into SRAM-based FPGA platforms were presented in [13]. The main principles of PDR were described together with the FT architectures based on TMR and duplex. Several architectures using online checkers for error detection were introduced in [12]. An error detected by the on-line checker initiates reconfiguration process of the faulty unit. The modification of FT architectures into Partial Reconfigurable Modules (PRM) and positive features of PDR when used in FT system design where demonstrated as well.

The FT structure for SRAM-based FPGA via PDR with Generic Partial Dynamic Reconfiguration Controller (GPDRC) inside FPGA was presented in [14]. The main role of GPDRC in FT system is seen in the identification of faulty PRM and the initiation of reconfiguration process of the faulty module in FT architectures through Internal Configuration Access Port (ICAP) interface. The problems with synchronization of PRMs after reconfiguration of one of them were demonstrated as well. The FT structure based on PRMs can be seen in Figure 2.

B. Another Fault Tolerant Design Methodologies

Attempts to categorize and compare various FT techniques and discuss how they can work together to provide a synergistic approach for fault tolerant FPGA design is presented in [15], unfortunately no experimental comparisons were presented.

The adoption of the TMR coupled with the PDR of SRAM-based FPGAs to mitigate the effects of SEU in such class of device platforms is presented in [8]. The authors propose an exploration of the design space with respect to several parameters as area and recovery time in order to select the most convenient way to apply this technique to the device under consideration.

In [16], an innovative method that allows the use of PDR combined with TMR in SRAM-based FPGAs FT designs is presented. The method uses coarse-grain TMR with special voters capable to identify faulty module and check point states that allow the sequential synchronization of the recovered module. The synchronization of the recovered module is performed while the others are kept running.

The FT scheme for Xilinx FPGA Virtex4 is presented in [17]. The scheme consists of two parts: the Partially Reconfigurable Functional Region (PRFR) with several PRM and reconfiguration controller is based on built-in Xilinx PowerPC-405 processor and ICAP interface.

C. Problem Definition and Goals of the Research

At the moment, many techniques how to develop SRAM-based FPGA systems exist. Many of them are based on replication of functional units (e. g. TMR architectures, duplex architectures) combined with CED techniques. They are not usually tested how vulnerable against SEU effects they are. It is not also described which technique was used to verify FT features of the architectures developed. It is important to say that some tools based on the use of ICAP to insert SEUs into FPGA bitstream exist [18],[19]. These techniques can achieve high speed of the injection but their main disadvantage is that the injector is implemented into the same FPGA as

the application and thus can damage itself by injecting SEU. Moreover, the presence of testing circuitry affect place and route phases of the design and therefore produces different designs with different fault tolerant properties.

Our approach is based on the external SEU generator which allows us to test the behaviour of our FT architectures and their reaction to SEU inserted into particular positions of the bitstream.

To summarize, we present SEU simulation framework for testing of fault tolerant system designs implemented into FPGA. The framework is based on SEU generation from Personal Computer (PC) through the JTAG interface and dynamic reconfiguration into FPGA and allows to select region of the FPGA for SEU placing.

The paper is organized as follows. Section III describes soft errors (SEU) principles in FPGA caused by high energy particles. Section IV presents basic description of the configuration process of the Virtex 5 chip. SEU simulation strategy and requirements on the SEU generator are described in Section V while the basic features and implementation of external SEU generator are demonstrated in section VI. Then, SEU simulation framework for testing fault tolerant system design in FPGA is presented in section VII. Finally, the results of our experiments (section VIII) together with the goals of our future research are mentioned (section IX).

III. Single Event Upset

SEU effect is defined as a modification of memory cell caused by the high energy particles that collided with memory cell. The probability of the SEU for one memory cell in earth condition is extremely small. However, this probability increases with the memory size or with the radiation background. Therefore SEU presents a real problem for example in space or aircraft designs. The FPGA vendors made large measurement of the SEU probabilities [20].

The problem of SEU can be dealt with in two ways. The first is in the prevention of the occurrence of the SEU. This is done by FPGA vendors, however as the size of technology decreases and the size of the device increases the devices became even more susceptible to SEUs. The result of the SEU prevention minimizes these effect but the SEU is still a problem. The second way to deal with the SEU is at the level of the designer. The system itself is responsible for detection and the mitigation of SEUs. Many of these methods are implemented by the FPGA vendors in form of additional tools or IPcores [18].

A. Maintain Strategy

The idea behind this approach came from the observation that SEU is relatively uncommon and therefore there is only a small chance that system will be effected by it in the limited amount of time. To mitigate SEU, system is shut down and fully reconfigured. The maintain strategy divides to the scheduled maintain and emergency maintain. The scheduled maintain strategy periodically reconfigures the FPGA to make sure that SEU effect cannot cumulate and that if SEU occurs it is repaired at most after given time. Emergency maintain has the ability to detect SEU in the configuration memory. The reconfiguration process is initialized immediately after detection of the SEU.

B. Bitstream Scrubbing

Bitstream scrubbing is used mostly for mitigation of SEU effect in the configuration memory but it allows the correction of the configuration memory once SEU is detected. The bitstream is periodically read frame by frame and every frame is checked with the reliable memory. If there is a difference, it is supposed that it was caused by SEU and it can be repaired by downloading the correct value back into the configuration memory. This process can be seen as a partial dynamic reconfiguration since part of the chip (one bit) is reconfigured without disruption of the computation. Obviously, the values of registers and RAMs must be masked out during the scrubbing since its value changes during the computation. The real life implementation of the bitstream scrubbing does not use reliable memory to save whole configuration but only its checksums to reduce memory overhead. Modern Xilinx FPGA contain the checksum as the part of the configuration memory. Therefore no additional memory is required for the bitstream scrubbing.

IV. Configuration Process

The configuration process of the FPGA is driven by commands contained in the bitstream generated by the FPGA tools. In this and the following sections we focus on the Xilinx Virtex 5 FPGA. The configuration memory is divided into frames with constant size of 1312 bits. Every frame contains 12 bits of ECC information which allows to detect two and correct one configuration error. According to the Xilinx documentation [21] frames are the smallest addressable parts of the configuration memory. Therefore, if the user wants to write one bit change into the bitstream he or she has to manipulate with the complete frame.

A. Reconfiguration Types

The reconfiguration process of the FPGA do not differentiate between any types of the reconfiguration used by the development tools. The basic sequence of the commands initialized is described in the [21]. Only command sequence changes according to the type of the configuration. For example, only difference between full and partial reconfiguration from the point of the configuration process is the beginning address of the write into the configuration memory and smaller size of written data. The difference between static and dynamic reconfiguration is represented by the shutdown sequence at the beginning of the bitstream.

V. SEU Simulation Strategy

SEU simulation is the process of changing one bit information in the configuration memory or in the memory of the FPGA design such as registers or BlockRAM. The SEU do not cause any error since not all parts of the configuration memory

are used in the design. According [22], only 10% of the configuration memory is used to define the design functionality in average. Unfortunately, it is not possible to predict if given bit is required for the design functionality or not in general due to the undocumented structure of the configuration memory and non-deterministic nature of the routing process.

A. Requirements on Testing Platform and SEU Generator

Every SEU simulator should meet a few criteria to be useful for the testing of the FTS. The proposed criteria was selected according to the authors experience with developing a fault tolerant methodology for FPGAs. The main requirements on SEU generator are:

- **Universality** – the SEU generator should be able to place SEU at any place of the FPGA not only to the configuration memory but also to the circuity in which the function is implemented. The universality property is required for testing of the design level mitigation techniques, such as TMR architecture or duplex system with checkers and multiplexer.
- **Locality** – the SEU generator should be able to place SEU into pre-determined area of FPGA and guarantee that other areas will remain unmodified. This property allows different level of testing to be used in different parts of FT architectures. Every reliable architecture has its weak points. For example, NMR and TMR architectures have voter unit as a very weak point of the architecture. If this unit fails then entire architecture fails as well. However, if implemented correctly, TMR will mask any error in the function unit. Property of locality ensures that the SEU generator is able to do exhaustive testing of the function unit without attacking voter.
- **Separateness** – the SEU generator should be separated and independent on the function implemented into FPGA. The separateness property also means that the SEU generator should be able to operate on any FPGA design without the need to rebuild the design. There are several reasons for this property to be satisfied. First, if the design is rebuilt for testing purposes, then the unit under test is different from the unit used in production and therefore the units can have different reactions to SEU injections. The second reason for the separateness is to guarantee that the generator will not damage itself. This reason is valid only for some parts of the designs. The PDR is the legitimate function of the FPGA and it can be also used in an FT system. SEU generator has to be separated from the design to ensure that it will not interfere with the PDR done by the design itself.
- **Atomicity** – the SEU injection should be seen as an atomic process from the design point of view to ensure that for example SEU in the register will not be replaced by a new value during its injection. The atomic property means, that the SEU injection has to be performed faster than the period between two pulses of maximal clock frequency in the design or that the FPGA logic has to be shut down during SEU injection. On the other hand, shutting down the logic can cause problems if the design is interacting with its environment, such as DRAM, Ethernet or other external function unit. Since at least one frame has to be written into the configuration memory, it is virtually impossible to place SEU in one clock cycle for any reasonable clock speed. For these reasons, the implementation of the atomicity in the generator will present problems and some users might wish to disable it for specific situations.

B. Xilinx's Solution

Xilinx offers its own solution called Soft Error Mitigation (SEM) [18]. However, this solution is designed mostly for the SEU mitigation and the SEU generation is only an additional function. This presents a problem from the theoretical point of view since one can argue that unit that is tested should not be used for the test generation. SEM is an IPcore generated only for Virtex6.

SEM has to be connected to the ICAP inside the FPGA to be able to mitigate SEU and therefore SEM does not meet the separateness property. Due to this fact, it is not possible to use ICAP without the functional change in the design itself.

The previous version of the SEM was called SEU Controller Macro and had only a limited support for locality. User was able to address every bit in the bitstream by linear addressing but it was not clear how the linear address was transferred to the frame address to test specified part of the FPGA only. The SEM supports a physical frame addressing which allows easier support for locality.

The atomicity issue is not solved by SEM. The SEM operates inside the FPGA and therefore it is not possible to shut down the FPGA for the injection of SEU. According [23] the switching characteristics of SelectMAP and therefore also ICAP for Virtex6 is 100MHz. The injection of SEU at this speed may take up to hundreds of clock cycles in the design.

It is important to point out that the purpose of SEM is not to generate SEU but to mitigate them. The SEU generation is only an additional function that may be used for the better understanding of the design behaviour in the presence of the SEU and even for simple tests. However, SEM is not sufficient for the full evaluation of the fault tolerant designs. Another problem may arise when the design itself is uses ICAP interface.

VI. Proposed Solution: External SEU Generator

This work proposes to implement SEU generation as a distinct tool working outside FPGA. The SEU generator should use JTAG interface since this interface has highest priority and therefore it can interrupt any other configuration interface.

A. Properties of SEU Generator

The basic principle of the SEU generator is to combine readback and dynamic reconfiguration. The process of the SEU generation can be described in four steps.

1) **Selecting frame** – This step selects frame into SEU will be generated. The selected frame has to be described

by four variables (row index, column index and minor address together with top or bottom bit). Generator then constructs the frame address.

2) **Readback of the frame** – This step readbacks the whole frame without stopping the computation in the FPGA.
3) **SEU generation** – This step changes one bit of the read data. The position of the changed data can be generated by random function or by the given SEU generation policy.
4) **Write frame** – This step writes the changed frame back into the configuration memory of the FPGA. The write is currently implemented without stopping the FPGA.

B. Implementation of SEU Generator

The proposed solution was implemented in the TCL language with the use of the ChipScope libraries. The correct function of the implementation was experimentally verified on the number of designs. No changes in the design were needed. The external PC SEU generator structure can be seen in Figure 3.

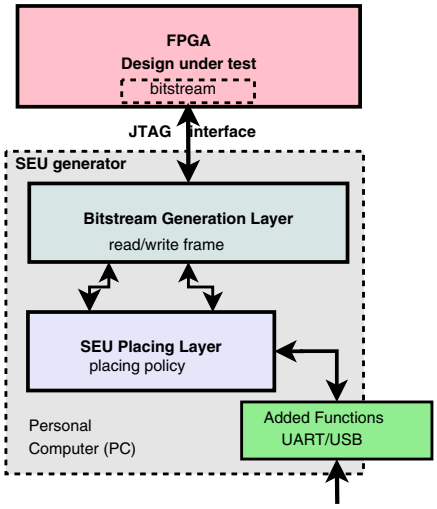

Fig. 3. External PC SEU generator structure

The TCL implementation is divided into two basic layers. The first layer is responsible for communication with the FPGA. It is called Bitstream Generation Layer. This layer uses the ChipScope libraries to send and read data through JTAG interface. However, it is possible to change this layer to use any other JTAG drivers. Our decision to use ChipScope was based on the fact that ChipScope offers TCL functions for manipulation with JTAG. The Bitstream generation layer accepts the frame address and frame data from the SEU placing layer and generates bitstreams that will readback or write data for the given frame. The frame address is specified by four values corresponding to the parts of the FAR register in FPGA. The first parameter is TOP. This variable can be 1 or zero and specifies which half of the FPGA contains the frame. The second parameter specifies the ROW of the FPGA in the given half (top or bottom). The last two variables describe the column. The first is called column and specifies which column of device is used counting by the columns visible in the PlanAhead software. Therefore by setting these three values a user is able to address any given column of the CLB. The configuration of the CLB is contained in 36 frames but this number is different for other types of columns, such as BlockRAMs. The fourth variable (Minor) is used to specify which frame of the given CLBs should be addressed. It is important to keep in mind that the frame contains configuration of 20 CLBs.

SEU placing layer is responsible for generation of the read and write frame requests according to the given SEU placing policy. The user will probably make its own alterations in this layer by adding a new functions for SEU placing. The typical functionality of the function in this layer is to compute position of the SEU according to the implemented policy, readback the frame containing the affected part of the memory, change the bit at the computed position and write the frame back into the FPGA. Since the frame is the smallest addressable part of the configuration memory, the SEU has to be placed into the given position by the function of this layer.

Currently, several placing policies are implemented:

- change any bit in one frame of bitstream or change several bits in frame (multiple SEU).
- change random bit in one frame of bitstream or change randomly several bits in frame.
- fill one frame by zero value or several frames set on zero values.
- fill one CLB with zero values.

The implemented solution was evaluated with four basic requirements presented in this paper. The solution is universal, since no requirements on the design in the FPGA exist. The locality is guaranteed by the addressing scheme which is the same as the addressing scheme used by the FPGA itself. Therefore, SEU generator achieves the same locality as the FPGA reconfiguration process. Separateness property is met by the implementation in the TCL on the PC. Atomicity is another feature available in the implementation. The extension of atomicity is fairly simple by stopping the FPGA. However, stopping the whole FPGA for the SEU generation can present synchronization problems between FPGA and the other components on the board. Therefore we chose to have shutdown sequence as an additional feature that is not part of the SEU generator.

The last block in the diagram of the SEU generator is called Added Functions. This block contains functions for interfacing the SEU generator to its environment. These functions make it possible to drive SEU generation by external sources, such as UART or external program. Currently, only serial communication is implemented.

VII. SEU Simulation Framework for Testing Fault Tolerant Systems Based on FPGA

The goal of our research was to develop an external SEU generator and verify its ability to insert SEUs to the required position in the bitstream. This gives us the opportunity to

test the behaviour of FT architectures and their reaction to SEUs. The framework allows us to insert multiple SEUs in one run and simulate the occurrence of higher number of SEUs. The architecture of the framework is shown in Figure 4. The framework consists of two components.

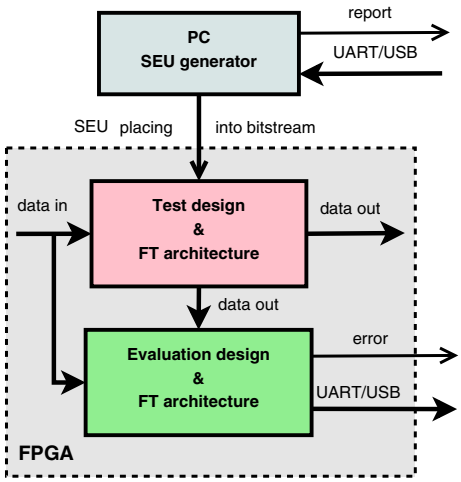

Fig. 4. Proposed SEU Simulation Framework for testing of FT architectures

The first component is the unit under test developed as a PRM or an architecture consisting of more PRMs. The other component is used for the evaluation - it contains another copy of functional unit, timing unit, control unit, comparator and UART controller. The timing unit allows to set the operation speed of the application related to the speed of SEU generation, evaluation of SEU effect and the transport of the result to PC through UART interface. To PC through UART the following information can be transported:

- SEU position and the number of SEUs which changed the behaviour of the unit under test,
- the number of incorrect values on outputs of sequential logic caused by SEU together with the reaction of checkers to SEUs,
- the number of SEUs generated in the same time slice.

We intend to do extensive experiments with various FT architectures and use the implemented SEU simulation framework for these experiments. It will allow us to compare these approaches.

VIII. Experimental Evaluation

The proper function of the SEU generation was tested by the readback verification in the Impact software. The SEU generator was able to change all bits in the frame except of four reserved bits in the middle of the frame. The SEU simulation framework was used to test FT technique presented in [11]. The evaluation logic on FPGA side (FU or FT architecture, timing unit, control unit, comparator and UART controller) were implemented in VHDL language, for the synthesis XILINX ISE 11.3 was used. ISE 11.3 supports PDR into FPGA Virtex5-XC5VSX50T on development board ML 506. For experimenting with framework, a digital circuit (FU) was developed which contains 8-bit counter and 8-bit decoder. The board was used to verify proper function of different types of FT architectures. The tested designs and FT architectures can be seen in Figure 5.

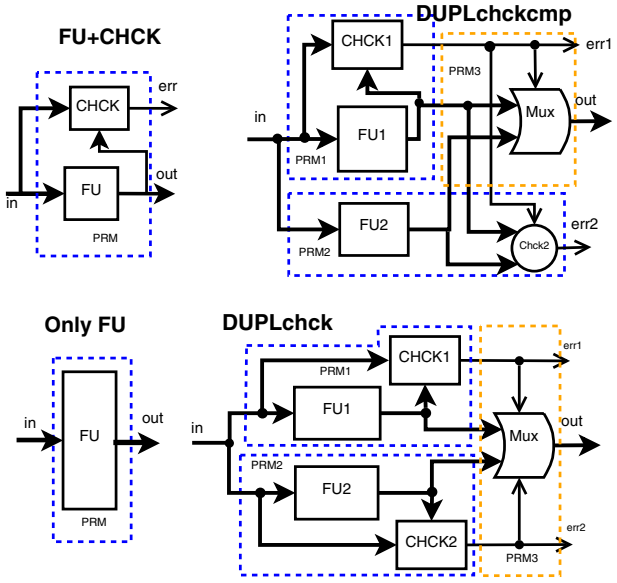

Fig. 5. Tested designs and FT architectures

The basic parameters of tested designs and FT architectures can be seen in Table 1. The meaning of the columns is as follows: column 1 - the type of architecture, column 2 - design size in Virtex5, column 3 - the number of LUTs, 4 - the number of FlipFlop registers, column 4 - the number of frames and last column 5 - the size of bitstream in bits for placing of SEU.

Virtex5 XC5VSX50T Counter8+Decoder8	Size [slices]	# LUT [-]	# FF [-]	# Frame [-]	# bits [-]
Only FU	8	23	8	36	47232
FU+CHCK	12	41	12	36	47232
TMRchckcmp	18	64	16	72	94464
DUPLchck	22	88	24	72	94464

TABLE I
Size of tested designs and FT architectures

The parameters of framework were set as follow: The generation frequency of one SEU into tested design including evaluation time (timer component) was set on 100 μs. The communication between FPGA UART controller and PC was set on 115200 Baud per second, data 8-bits, parity even and 2 stop bits.

The goal of the first experiment was to verify how many SEUs will effect the correct function of the architecture under test and what will be the consequences for the architecture if SEUs are injected into functional units only. One-bit errors were injected into all positions of the bitstream. It was tested how many injected errors will have an impact on the function

and how many errors will be detected by checkers. The results are provided in Table 2.

XC5VSX50T CNT8+DEC8	Bitsream size [bits]	# Detected SEUs in FUs	SEU detected by checker	# Output data errs
FU+CHCK	47232	1996	100%	1996
DUPLchckcmp	94464	4040	99%	2988
DUPLchck	94464	4423	100%	3064

TABLE II
NUMBER OF DETECTED SEU IN FUS OF ARCHITECTURE

The meaning of the columns is as follows: column 1 - the type of architecture, column 2 - the size of bitstream in bits for placing of SEU, column 3 - the number of detected SEUs in FUs of architecture, 4 - the number of detected SEUs in FUs by checkers and last column 5 - the number of incorrect data on the outputs of architecture.

During another experiment it was verified how many SEUs destroy the correct function of the architecture under test including checkers and output logic. It was done in the same way as other experiments, i. e. one bit errors were injected to all bitstream positions. It was evaluated how many SEUs will have an impact on the correct function and how many errors cause complete non-operation of the system (see Table 3).

Virtex5 XC5VSX50T Counter8+Decoder8	Bitsream size [bits]	# Detected SEU	# Total errors
Only FU	47232	1996	1348
DUPLchckcmp	94464	5857	1262
DUPLchck	94464	6850	1606

TABLE III
NUMBER OF DETECTED SEU IN FT ARCHITECTURE

The meaning of the columns is as follows: column 1 - the type of architecture, column 2 - the size of bitstream in bits for placing of SEU, column 3 - the number of detected SEUs in architecture, 4 - the number of errors, where all states of sequential logic were incorrect.

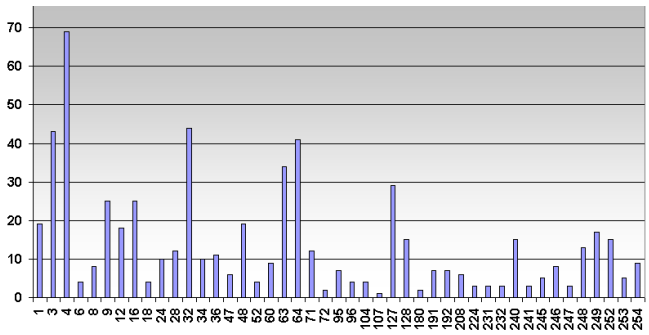

Fig. 6. Histogram of error states of FU

In Figure 6, the histogram reflecting the testing of FU is provided. It demonstrates, how many different values appeared on FU output (caused by SEU injection) in 8-bit counter and how often they occurred. The X axis represent the number of errors during one step of SEU generation process and the Y axis represent the number of such SEU. In Figure 7 it is same for DUPLchck FT architecture.

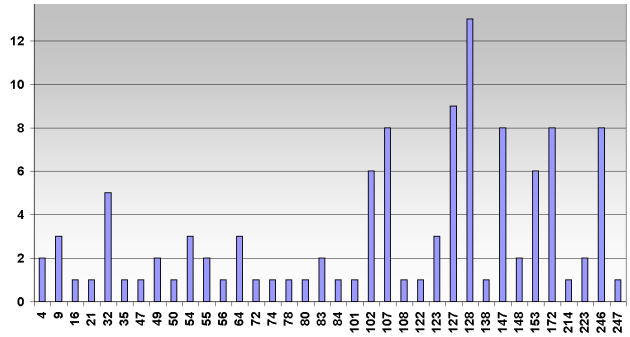

Fig. 7. Histogram of error states of FT architecture

Finally, it was verified that all checkers and other checking units were able to detect SEU injected into architecture. The digital component continued to cover its function during the existence of the SEU and also during the reconfiguration process and provided correct outputs. It was proven, that, if correctly designed, the unit equipped with the checkers are FT. However, routing tools routes signals through reconfiguration region, which allowed damage of the static part of the design by the SEU in the dynamic parts.

IX. CONCLUSION

In the paper, SEU simulation framework for testing fault tolerant system designs implemented into FPGA was presented. The framework is based on SEU generation from personal computer through JTAG interface and reconfiguration of FPGA bitstream. The framework allows to select a region of the FPGA for SEU placing. Four basic features which must be satisfied by a reliable SEU generator were defined and reflected in our SEU injector implementation. The SEU simulator does not require any changes in the tested design and is fully independent of the function implemented into FPGA.

The SEU simulation framework is based on the proposed SEU generator and the ML506 development boards to allow interaction of the tested unit with the environment and to be able to evaluate exactly the number and relevance of mistakes occurring in the unit under test. The experimental results demonstrated that all checkers of FT architectures and other checking units were able to detect SEUs injected into design.

Future work will be focused is on extensive testing of various architectures of fault tolerant systems implemented into FPGA, the presented SEU simulation framework will be used. Results gained from experiments will be used to compute parameters of reliability models of researched architectures.

Acknowledgements

This work was supported by the Grant Agency of the Czech Republic (GACR) No.102/09/1668 - ”SoC circuits reliability

and availability improvement" and by Research Project No. MSM 0021630528 - "Security-Oriented Research in Information Technology" and the grant "BUT FIT-S-10-1".

REFERENCES

[1] P. Sundararajan, "High performance computing using fpgas."

[2] B. Osterloh, H. Michalik, S. A. Habinc, and B. Fiethe, "Dynamic partial reconfiguration in space applications," vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 336–343.

[3] R. Oliveira, A. Jagirdar, and T. J. Chakraborty, "A tmr scheme for seu mitigation in scan flip-flops," in *ISQED '07: Proceedings of the 8th International Symposium on Quality Electronic Design*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 905–910.

[4] P. Kenterlis, N. Kranitis, A. Paschalis, D. Gizopoulos, and M. Psarakis, "A low-cost seu fault emulation platform for sram-based fpgas," in *12th IEEE International On-Line Testing Symposium (IOLTS'06)*. New York, NY, USA: ACM, 2006, pp. 235–241.

[5] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator validation of an fpga seu simulator," vol. 50, no. 6, 2003, pp. 2147 – 2157.

[6] F. L. Kastensmidt, G. Neuberger, L. Carro, and R. Reis, "Designing and testing fault-tolerant techniques for sram-based fpgas," in *CF '04: Proceedings of the 1st conference on Computing frontiers*. New York, NY, USA: ACM, 2004, pp. 419–432.

[7] J. Emmert, C. Stroud, B. Skaggs, and M. Abramovici, "Dynamic fault tolerance in fpgas via partial reconfiguration," in *FCCM '00: Proceedings of the 2000 IEEE Symposium on Field-Programmable Custom Computing Machines*. Washington, DC, USA: IEEE Computer Society, 2000, pp. 165–170.

[8] C. Bolchini, A. Miele, and M. D. Santambrogio, "Tmr and partial dynamic reconfiguration to mitigate seu faults in fpgas," in *DFT '07: Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 87–95.

[9] J. A. Cheatham, J. M. Emmert, and S. Baumgart, "A survey of fault tolerant methodologies for fpgas," vol. 11, no. 2. New York, NY, USA: ACM, 2006, pp. 501–533.

[10] 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.

[11] M. Rebaudengo, M. S. Reorda, and M. Violante, "Simulation-based analysis of seu effects on sram-based fpgas," in *Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications*, ser. FPL '02. London, UK, UK: Springer-Verlag, 2002, pp. 607–615.

[12] M. Straka, Z. Kotasek, and J. Winter, "Digital systems architectures based on on-line checkers," in *11th EUROMICRO Conference on Digital System Design DSD 2008*. IEEE Computer Society, 2008, pp. 81–87.

[13] M. Straka, J. Kastil, and Z. Kotasek, "Modern fault tolerant architectures based on partial dynamic reconfiguration in fpgas," in *13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*. New York, NY, USA: IEEE Computer Society, 2010, pp. 336–341.

[14] ——, "Fault tolerant structure for sram-based fpga via partial dynamic reconfiguration," in *13th EUROMICRO Conference on Digital System Design DSD 2010*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 87–95.

[15] U. Sharma, "Fault tolerant techniques for reconfigurable platforms," in *A2CWiC '10: Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*. New York, NY, USA: ACM, 2010, pp. 1–4.

[16] C. Pilotto, J. R. Azambuja, and F. L. Kastensmidt, "Synchronizing triple modular redundant designs in dynamic partial reconfiguration applications," in *SBCCI '08: Proceedings of the 21st annual symposium on Integrated circuits and system design*. New York, NY, USA: ACM, 2008, pp. 199–204.

[17] X. Iturbe, M. Azkarate, I. Martinez, J. Perez, and A. Astarloa, "A novel seu, mbu and she handling strategy for xilinx virtex-4 fpgas," in *International Conference on Field Programmable Logic and Applications, 2009. FPL 2009.* Washington, DC, USA: IEEE Computer Society, 2009, pp. 569–573.

[18] XILINX, "User guide: Logicoretm ip soft error mitigation controller v1.1."

[19] L. Sterpone and M. Violante, "A new partial reconfiguration-based fault-injection system to evaluate seu effects in sram-based fpgas," vol. 54, no. 4, 2007, pp. 965 –970.

[20] A. Lesea, "Continuing experiments of atmospheric neutron effects on deep submicron integrated circuits."

[21] XILINX, "User guide: Virtex-5 fpga configuration user guide."

[22] K. Chapman, "Xapp864: Seu strategies for virtex-5 devices."

[23] XILINX, "Virtex-6 fpga data sheet:dc and switching characteristics."