Scientific original paper

Informacije MIDEM
Journal of Microelectronics,
Electronic Components and Materials
Vol. 42, No. 3 (2012), 144 – 151

On-line Testing and Recovery of Systems on SRAM-based FPGA

Uroš Legat

Jožef Stefan Institute, Ljubljana, Slovenia

Abstract: This paper outlines the techniques of on-line testing, error-mitigation and error recovery for SRAM-based FPGAs and gives guidelines how to make a small and efficient error recovery mechanism. The mechanism checks the configuration memory of the FPGA and reconfigures the FPGA if the error occurs. Triple-modular redundancy was applied to the mechanism to increase its reliability. The error recovery mechanism was implemented in Virtex 5 FPGA and verified on two user applications.

Keywords: *FPGA, single-event upset, on-line testing, error-mitigation, error-recovery*

Sprotno testiranje in popravljanje sistemov osnovanih na vezjih FPGA

Izvleček: Uvodoma članek razloži tehnike sprotnega testiranja, izogibanja napak in popravljanja napak na vezjih FPGA osnovanih na statičnem pomnilniku, kasneje pa poda navodila za izdelavo majhnega in učinkovitega mehanizma za sprotno popravljanje napak. Mehanizem sproti pregleduje konfiguracijski spomin vezij FPGA in reprogamira vezje na mestu, kjer najde napako. Mehanizen je implementiran po metodi trojne modularne redundance, s čemer mu povečamo zanesljivost. Mehanizem za popravljanje napak je narejen in preverjen za Virtex 5 FPGA, vendar ga z manjšimi modifikacijami lahko prenesemo tudi na druge tipe FPGA vezij. .

Ključne besede: FPGA, napake SEU, sprotno testiranje, izogibanje napakam, popravljannje napak

* *Corresponding Author's e-mail: uros.legat@ijs.si*

1. Introduction

SRAM-based FPGAs have become an attractive solution for many applications where a short development time, low-cost for low-production volumes and in-the-field-programming ability are important issues. The flexibility of SRAM-based FPGAs comes from the adoption of a configuration memory that defines the operations of the circuit that the FPGA implements. It is therefore fundamental that the content of the configuration memory preserves the correct values during the FPGA operation. An important concern for the reliability and dependability of SRAM-based FPGAs are radiation-induced soft-errors that corrupt the configuration memory (produce bit-flips). These errors often occur in the space environment; however, because of increasing integration density they are also not uncommon at sea-level.

Different fault-tolerance techniques have been developed to increase the reliability and dependability of applications on FPGAs [1]. These techniques function concurrently (on-line) with the system to monitor its operation. On-line testing techniques detect the errors in the system, error mitigation techniques are able to enhance the system to work despite faults, and error-recovery techniques recover the faults from the system. The goals of the fault-tolerance techniques are to minimize the hardware, timing, and power overhead, and maximize the reliability of the system.

The paper is organized as follows. Section 2 describes how soft-errors corrupt the operation of SRAM-based FPGAs. Section 3 explains the state of the art fault tolerance techniques. In section 4 error recovery mechanism (ERM) in different FPGAs is described. Section 5 shows the implementation of the ERM in Xilinx Virtex 5 FPGA and section 6 concludes the peper.

2. Soft errors in SRAM-based FPGAs

SRAM-based FPGAs are susceptible to radiation-induced soft-errors. The main FPGA reliability concern is a type of soft-error called single-event upset (SEU).

A SEU occurs when a charged particle strikes a memory cell and changes its state. For example, a typical SRAM memory cell is comprised of four transistors shown in Figure 1. A memory cell has two stable states that represent one bit of stored information. In each state two transistors are turned off (SEU target drains). When a charged particle strikes a drain in an off state transistor as in Figure 1, it can generate a transient current pulse to turn the gate of the opposite transistor on, which changes the state of the memory cell.

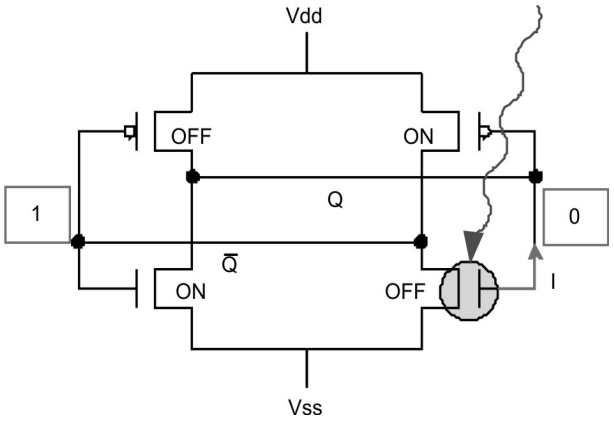

Figure 1: SEU in a SRAM memory cell

Configuration memory of a SRAM-based FPGA is comprised of SRAM memory cells. A charged particle can cause a bit-flip in the configuration memory cell and consequently alter the FPGA functionality.

A configuration bit is associated with a particular part of the FPGA. It can be a part of an internal memory of the device like an internal RAM or flip-flop, or it can represent a functional part of the design, like a Logic Block (LB), or internal routing [4].

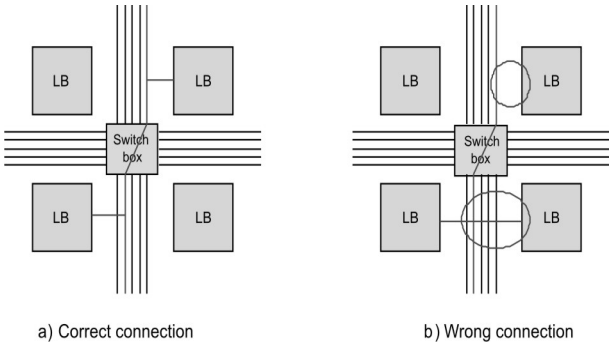

Figure 2: SEU in an internal FPGA routing

The internal routing interconnects the LBs, I/O blocks and other functional blocks of the FPGA. The routing consists of switch boxes that connect the main wires and smaller wiring segments that connect the main wires to LBs, shown in Figure 2 a. These connections are determined by the logic state of their configuration bits. A SEU affecting these configuration bits could disconnect the original LB connection, or in another case, connect wrong LBs. For illustration, some typical faults are marked in Figure 2 b.

The simplified structure of a LB is shown in Figure 3. The LB in Xilinx FPGA consists of a number of look-up tables, flip-flops and internal carry and control logic. The SEU can alter the logic function of the LUT, alter the connections inside the Carry and control logic, change the contents of the flip-flop, etc.

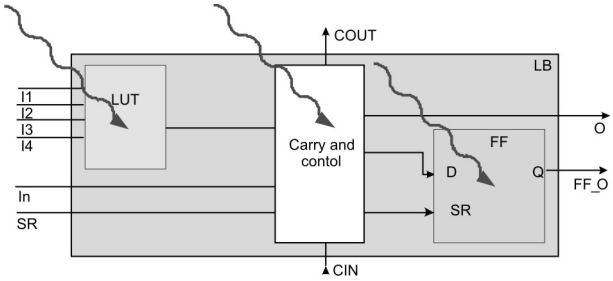

Figure 3: SEU in Logic Block

3. On-line testing and fault tolerance techniques

Different techniques have been proposed to test and protect SRAM FPGAs from SEU. On-line testing techniques detect errors during the normal operation of the system. On-line detection of errors shortens fault-detection latency, which is very important in order to prevent the fault from propagating further through the system. On the other hand, error-mitigation techniques can tolerate faults that occur during the system operation. If a fault occurs in one part of the circuit, then a redundant part of the circuit is used to provide the correct and uninterrupted operation of the system. When a fault is detected inside a system it can be repaired (recovered) by error-recovery techniques.

On-line testing techniques

On-line testing is performed while the circuit is performing its assigned task. Two types of on-line test are distinguished in the literature: the concurrent and the non-concurrent on-line test.

A non-concurrent on-line test is usually triggered in phases of system inactivity or in periodic and scheduled times when the normal function of the system is interrupted. Non-concurrent testing is used to detect permanent faults (SEU) and cannot detect transient faults (SET), whose effects disappear quickly. The non-concurrent on-line test is performed only virtu-

ally in parallel to system operation. Therefore, it is in some literature classified as an off-line technique. We give some examples of non-concurrent on-line testing techniques. The authors in [5] used scan chains to periodically check the system while [6] used logic BIST. Authors in [7] implemented a periodic on-line test in an embedded microprocessor.

A concurrent on-line test runs in concurrence with the system and does not interrupt its normal operation. The concurrent testing techniques use different kinds of redundancies to detect errors. Time redundancy is normally used to detect a transient faults in combinational circuits, while hardware redundancy is used to detect a SEU in sequential circuits or configuration memory. The on-line testing principle is depicted in Figure 4. Test vectors are generated by the normal operational inputs. Besides the original circuit there is a redundant part of the circuit that produces additional encoded outputs. A checker is monitoring these outputs and thus performs error detection.

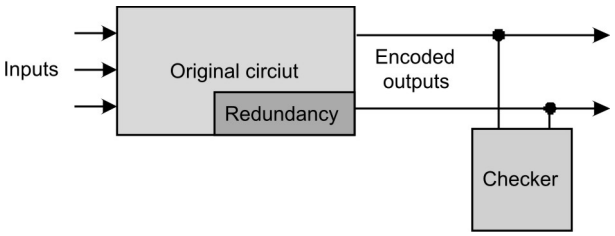

Figure 4: Concurrent on-line testing principle

A system protected with the concurrent on-line testing technique is also called a self-checking system [8]. The desirable goal of self-checking systems is to achieve the so-called totally self-checking property. This property requires that every fault in the system is detected before or at the time this fault produces an erroneous output. To achieve this goal the system has to meet the following criteria [9]:

- Fault secure criterion requires that any fault in the system produces erroneous outputs that can be detected at the output. This criterion assures that every single fault can be detected.
- Self-testing criterion requires that for each fault there is at least one input vector, occurring during normal operation of the circuit, which detects it.

The most straight forward hardware redundancy technique is duplication and comparison. The principle of the technique is that we make two copies of the circuit which run in parallel. The duplicates receive identical inputs. The outputs of the both circuits are compared by the comparator circuit. This technique increases the hardware cost by more than 100%. Duplication and comparison is used in a variety of different systems. In this way, on-line testing of embedded processor cores was improved by [10]. Processor cores are duplicated and a checker monitors whether the outputs of both cores match. If the outputs mismatch the processor state is restored from the previously saved states (checkpoint and rollback recovery method). Applicability of duplication and comparison in asynchronous circuits was investigated in [11]. Testing of finite-state machines using a technique similar to duplication and comparison was proposed in [12].

To reduce hardware cost, other more elaborate techniques are employed. These techniques use error-detecting codes (EDC) with costs lower than the duplication. The EDCs are used in sequential circuits and in memories. The codes that are used for error detection are: Parity codes, Hamming codes, Dual-rail codes, m out of n codes, Berger codes, and Arithmetic codes.

The most commonly used codes for error detection in FPGA configuration memories are the so called Single-error detection double-error correction (SEC-DED) Hamming codes [13] and Cyclic Redundancy Check (CRC) [14].

Error-mitigation techniques

For mission-critical systems it is sometimes not enough to only detect a fault, but also to operate in the presence of a fault which is possible by applying error-mitigation techniques. These techniques are also based on different kinds of redundancies.

The best-known hardware-redundancy mitigation technique is Triple Modular Redundancy (TMR). This technique is one of the n-modular redundancy techniques which were derived by [15]. The basic TMR technique triplicates the entire circuit into three modules and places the majority voter at the output of the modules. This method is effective against SETs and SEUs that occur in a single design module. However, if the upset occurs in the majority voter circuit the basic TMR is ineffective and a wrong value will be presented at the output. The hardware overhead of this method is three times the original design plus the voter circuit. While the hardware overhead is large, some have proposed partial TMR techniques which are focused only triplicating the specific sensitive logic [16].

The basic TMR solution does not avoid the accumulation of upsets. The FPGAs cope with this problem by implementing an on-line error-recovery technique.

To apply the TMR technique effectively in the FPGA device additional restrictions have to be considered. The triplicated modules have to be placed isolated from

each other (different clock regions) and the internal signals have to be carefully routed to limit the possibility that an upset would affect more than one module. All the modules have to have separate clock and input signals. Routing the TMR design in the FPGA is a particularly hard problem. Various algorithms and design methods have been proposed to reduce the number of such errors [17,18].

For Xilinx FPGAs a special hardened TMR architecture has been proposed in [19]. This architecture can also be automatically generated by their tool (Xilinx TMRTool). The XTMR is exploited based on the states recovery TMR method and uses specific FPGA resources to implement the majority voters. The Xilinx TMR is depicted in Figure 5:

- Inputs are connected outside of the FPGA and separated inside the FPGA.
- Combinational logic is triplicated.
- Sequential logic is implemented with majority voters and feedback loops.
- Outputs are implemented using tri-state output buffers in combination with minority voters.

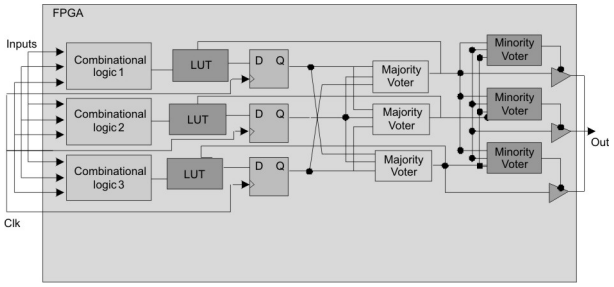

Figure 5: The architecture of Xilinx TMR

Error-recovery techniques in SRAM-based FPGAs

The configuration memory of the FPGA determines the functionality of the FPGA. The original configuration is downloaded into an FPGA at the power-up. During the operation of the device a SEU can occur, which changes the configuration of the FPGA. The error-recovery techniques are used to recover the original state of the configuration memory.

Simple error-recovery techniques (scrubbing techniques) only periodically reconfigure the whole device from the external memory. The external memory has to be radiation hardened and reliable to assure the correctness of the original ("Golden copy") configuration memory. The scrubbing period has to be adjusted to be less than the estimated mean time between two SEUs.

More advanced error-recovery techniques check whether the configuration memory is correct during the normal operation of the user application and can be regarded as on-line recovery techniques. These techniques require an error-recovery mechanism that monitors the configuration memory and in the case of an error recovers only the faulty bits using the partial runtime reconfiguration of the FPGA.

The recovery mechanism uses different ECCs to check the configuration memory. In [20, 21] Hamming code is used. The Hamming check bits are stored within the configuration of the Xilinx FPGAs. Asadi et al. [22] used Cyclic Redundancy Check (CRC) of the configuration memory. The CRC values are stored separately inside the internal memory of the FPGA.

Depending on which FPGA configuration interface is used to reconfigure the device, the recovery techniques are classified as either external or internal. The external techniques use one of the external configuration ports (i.e., JTAG, SelectMap). The external recovery technique requires a reliable recovery mechanism. Hulme et al. [23] proposed a radiation-hardened processor to control the recovery process, Asadi et al. [22] used a small auxiliary FPGA to check the main FPGA for errors, and Berg et al. [24] implemented a controller in ASIC.

An external recovery mechanism produces extra implementation costs. Hence, internal SEU-recovery techniques were proposed. They use internal configuration interface (for example, the internal-configuration-access-port-ICAP). The internal recovery controller is implemented in the FPGA along with the user application. It has to be small, fast and reliable. Heiner et al. [20] and Chapman [21] use an embedded microprocessor (PisoBlaze) as a configuration controller while Legat et. al. [25] use a small hardware mechanism based on finite state machine (FSM).

4. Mechanism for on-line test and recovery of errors in SRAM based FPGAs

In this section the knowledge of on-line testing, error mitigation and error recovery are used to show how to develop an efficient internal error-recovery mechanism (ERM) for SRAM FPGAs. The ERM will be able to test the configuration memory of SRAM FPGAs during the operation of the device (on-line) and recover errors through reconfiguration. The mechanism will be implemented in TMR to increase its reliability.

On-line test of the FPGA configuration memory

The functionality of a SRAM FPGA is determined by the state of its configuration memory (i.e., the SRAM cells).

The configuration memory in Xilinx, Atmel or Altera FPGA devices is organized in a network of configuration frames that are laid out on a device according to their frame address. A configuration frame is the smallest reconfigurable part of an FPGA. The content of the configuration memory is loaded through the configuration interface at the initial configuration of the device and must remain unchanged during its operation.

The basic idea of the mechanism is to continuously read the contents of the configuration memory during the operation of the device through a configuration port and check its integrity. Some of the SRAM-based FPGA devices have an internal configuration port which can be directly accessed by the mechanism; other devices have to use an external connection to configuration pins.

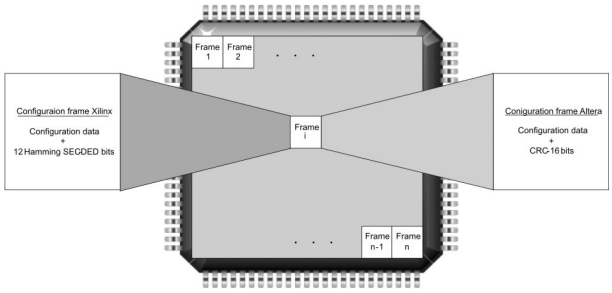

Figure 6: The structure of configuration memory in SRAM-based FPGAs

To validate the integrity of the configuration memory the mechanism reads the configuration memory frame by frame and checks the frame data using an ECC. Different FPGAs use different ECCs in their configuration frames. Figure 6 shows the structure of the configuration frames of Xilinx and Altera FPGAs.

Xilinx FPGAs
Each configuration frame in Xilinx FPGA device contains 12 parity bits. These are the parity bits of the Hamming SEC-DED Error Correction Code (ECC). The parities are pre-calculated and stored with the configuration data. To perform the error-detection we have to implement a *Hamming decoder. As the configuration frame is read through the configuration port the Hamming decoder calculates a syndrome value. The syndrome is calculated from the 12 parity bits and the rest of the read frame data. The first 11 bits of the syndrome value identify the location of a single erroneous bit within the frame (including the errors in the parity bits), while the last bit of the syndrome value indicates the double error in the frame.*

Altera FPGAs
Each configuration frame in Altera FPGA device contains a CRC-16. The 16 check bits are pre-calculated and appended to the configuration data. For error-detection we have to implement a LFSR. The LFSR uses 16 flip-flops and three XOR gates placed at the positions determined by the generator polynomial ($x^{16}+x^{15}+x^2+1$). As the frame is read the configuration frame data is shifted through the LFSR and the output of the LFSR is checked at the end of the readback. The CRC can detect single and multiple faults in the frame, but it cannot determine the position of the error.

Error-recovery procedure

When an error is detected in a configuration frame, the mechanism triggers error recovery procedure. The error recovery is the process of correcting the configuration memory through configuration port. The FPGA devices that do not have partial reconfiguration capabilities have to stop and reconfigure the whole configuration memory. Some of the FPGA devices enable partial runtime reconfiguration (newer Xilinx, Altera, and Atmel FPGAs). These devices can recover a single faulty frame during the operation of the device. The recovery procedure goes as follows:

In Xilinx FPGAs (Virtex family) the ERM does not require an external memory for the recovery of single faults. The SEC-DED Hamming ECC determines the location of a single error inside the configuration frame. The ERM corrects the faulty bit in the read configuration frame and reconfigures it. If a double error occurs in a configuration frame the recovery procedure is only possible from the external memory.

In Altera FPGAs (Stratix and Cyclone families) the location of the error inside the frame is unknown; therefore the original frame data has to be stored in the external memory. The corrupted frame is read from the external memory and reconfigured using partial runtime reconfiguration.

Implementation of error recovery mechanism

The hardware architecture of ERM is shown in Figure 7. It consists of *configuration port, ECC core, internal FPGA RAM, and control logic.*

The FPGA device is configured by writing the configuration commands into the configuration registers. The configuration user guides of particular FPGA devices give a detailed description of the register types and commands to perform the configuration operations.

The configuration port can be internal or external. It has direct access to the configuration registers and configuration data. The error-recovery mechanism uses the port to read and write a configuration frame.

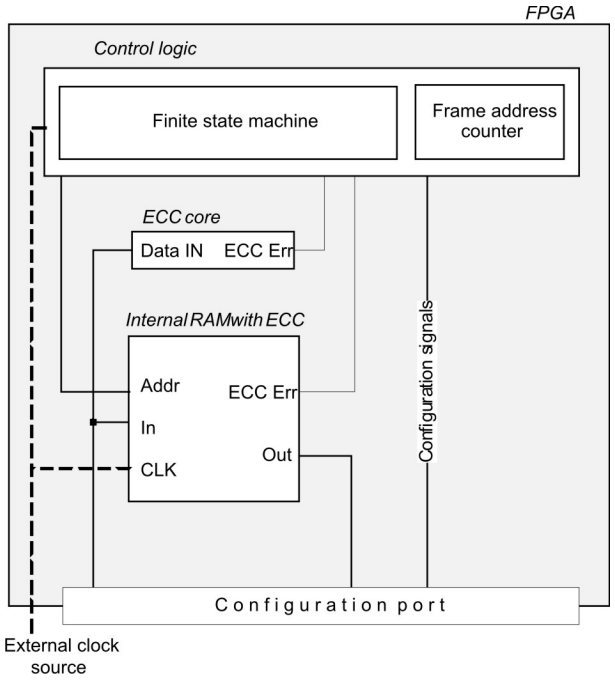

Figure 7: Hardware architecture of the error-recovery mechanism

The readback and reconfiguration operations are performed using an appropriate sequence of configuration commands sent to the inputs of the port. These commands are predefined and stored in the internal FPGA memory.

The ECC core is used to detect errors inside the FPGA configuration frame. Depending on the FPGA family this can be a Hamming decoder or a CRC LFSR. The ECC device checks the configuration frame while the frame is read through the configuration port.

The internal FPGA RAM contains the configuration commands and temporarily stores the current configuration frame. The internal RAM is also susceptible to SEU. To protect the integrity of the RAM content most of the FPGAs devices have an embedded Hamming SEC-DED ECC circuit.

The control logic manages the error detection-and-correction process. The controller is composed of a *Finite State Machine (FSM)* and a *Frame address counter.* The FSM controls the operation of the mechanism. When the next frame address is required the Frame address counter is incremented. When the frame address reaches the last frame, the counter is reset back to the initial value pointing to the first frame.

TMR implementation of the error recovery mechanism

The internal error-recovery mechanism is also susceptible to SEUs. A critical fault in the mechanism could cause a system-wide corruption of the configuration data. Therefore, it is essential that the logic of the ERM is protected by some SEU-mitigation technique.

The ERM can be implemented in TMR. The hardware architecture of the TMR is depicted in Figure 8. The TMR structure can be applied to the control logic, ECC core and the internal RAMs. These components of the ERM are triplicated in three design modules. A majority voter is placed at the inputs of the configuration port. The outputs of the configuration port are triplicated outside the FPGA are connected to the three design modules inside FPGA. The triplicated design modules are clocked by separate synchronous clock signals.

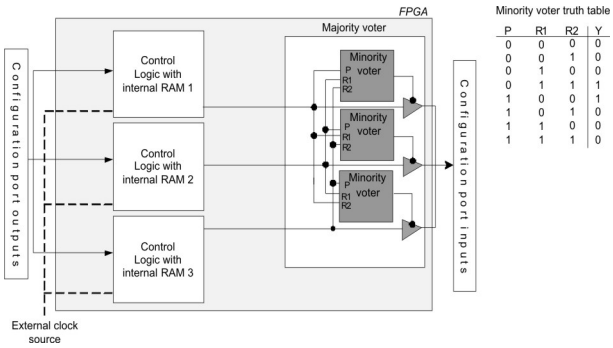

Figure 8: Hardware architecture of the TMR version of the error-recovery mechanism

The majority voter can be regarded as a single point of failure. The recommended implementation of the majority voter in FPGAs is shown in Figure 8. This majority voter implementation is immune to the single points of failures. The majority voter is implemented using three output tri-state buffers that are a part of the FPGA OI blocks, and three minority voters. The operation of the voter is as follows: If the primary signal (P) of the minority voter is a part of the majority (see the minority voter truth table in Figure 8), then the minority voter will enable the corresponding (active low) output buffer allowing the data on the output. If the primary path is not a part of the majority, then the output buffer is disabled placing its output in a high-impedance state allowing the redundant outputs to drive the correct data.

To apply the TMR technique effectively in the FPGA device additional restrictions have to be considered. The triplicated modules have to be placed in such a way that they are isolated from each other and the internal signals have to be carefully routed to limit the possibility that an upset would affect more than one module.

5. Implementation details and operation of ERM

The proposed ERM was implemented in Xilinx Virtex 5 (XC5VLX30) FPGA [25]. An internal-configuration access-port (ICAP) was used to access the configuration memory. A frame ECC device was used to check the frame data and a FSM was built to control the process of configuration scan and recovery. The ERM was also implemented in TMR.

Implementation details of original ERM vs. TMR design

Implementation details of the ERM implemented in Virtex 5 are shown in Table 1. It occupies just 72 slices, 115 Flip-flop registers and 1 internal block RAM. A total of 36 kb of RAM is used with embedded ECC. The mechanism utilizes less than 1% of the resources available on the FPGA. Implementation can differ slightly when different options are selected in the synthesis tools. In comparison with the original version of the recovery mechanism TMR increases the number of occupied resources by some more than three times.

The power consumption of both versions of ERM was analyzed using Xilinx XPower tool on Virtex 5 FPGA at 100 MHz clock rate. The dynamic power consumption of the ERM is negligible in comparison with the static leakage of the FPGA. The TMR version of the mechanism has approximately three times higher dynamic power consumption than the original mechanism. On the other hand, due to the high static power consumption of the FPGA the total power consumption of the TMR version is only 4% higher than the power consumption of the original mechanism.

Table 1: Comparison of resources, power, and timing of the original and TMR version of ERM

Virtex 5	Original mechanism	TMR mechanism
Resources		
Slices	72	321
Flip-flops	115	345
BRAM	1	3
Power consumption		
Dynamic (mW)	20	56
Static leakage (mW)	894	895
Total (mW)	914	951
Timing		
Max Clock (MHz)	282	261

Timing analysis was done using Xilinx ISE tool. The maximum clock frequency of the TMR version of the error recovery mechanism is 8% lower than the clock frequency of the original version. The decrease of the maximum clock frequency is the result of a slightly longer critical path. The frequency 261 MHz is still more than enough since the ICAP circuit which is used by the mechanism is recommended to run below 100 MHz.

Error-recovery time

The error-recovery time of the mechanism depends on the size of the configuration memory of particular FPGA device. One configuration frame is checked in 41 clock cycles and recovered in 210 clock cycles. In our case the Virtex 5 (XC5VLX30) FPGA was used. This device has 5515 configuration frames. The worst case error detection time is 226115 clocks or with 100 MHz clock rate 2.3 ms.

Verification of ERM

The operation of error-recovery mechanism was verified by injecting faults into tested user applications and checking if the ERM recovered the errors. The fault injection was performed by our fault injection tool [26].

Two user applications were used. The first device under test was an Advanced Encryption Standard (AES) implementation from [27]. The AES core occupies 537 LUTs, 165 Flip Flops and 3 internal RAM blocks. 337184 single faults were injected in the part of FPGA where the AES was placed. The ERM recovered all the injected faults. The second device under test was a hardware implementation of a secure IEEE 1148.1 standard from [28]. The boundary scan core occupies 65 LUTs and 119 Flip Flops. 181056 single faults were injected in this device and all the injected faults were recovered by the ERM.

6. Conclusions

An error-recovery mechanism for SRAM-based FPGAs was proposed. The guidelines for its implementation in different FPGA families are given including the implementation in triple-modular redundancy. The mechanism was verified on Xilinx Virtex 5 FPGA using two case study applications.

Future work includes a detailed analysis of TMR structure of the ERM. By performing a fault injection experiment possible points of failure will be identified and further hardening solutions will be proposed.

References

1. Lima Kastensmidt, F.; Carro, L.; Reis, R. *Fault-Tolerance Techniques for SRAM-Based FPGAs* (Springer, Dordrecht, 2006).
2. Schrimpf, R. D.; Fleetwood, D. M. *Radiation Effects And Soft Errors In Integrated Circuits and Electronic Devices* (World Scientific, London, 2004).
3. Messenger, G. C. Collection of charge on junction nodes from ion tracks. *IEEE Trans. On Nuclear Science* 29, 2024 (1982).
4. Rebaudengo, M.; Sonza Reorda, M.; Violante, M. A new functional fault model for FPGA Application-Orianted testing. In: *Proceedings of Defect and Fault Tolerance in VLSI Systems*. 372–380 (2002).
5. Al-Asaad, H.; Shringi, M. On-line built-in self-test for operational faults. In: *Proceedings of IEEE AUTOTESTCON*. 28–32 (2000).
6. Yang, F.; Chakravarty, S.; Devta-Prasanna, N.; Reddy, S. M.; Pomeranz, S. M. R. An Enhanced Logic BIST Architecture for Online Testing. In: *Proceedings of 14th IEEE Int. On-Line Testing Symp-IOLTS*. 10–15 (2008).
7. Paschalis, A.; Gizopoulos, D. Effective software-based self-test strategies for on-line periodic testing of embedded processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24, 88 (2005).
8. Nicolaidis, M.; Zorian, Y., On-Line Testing for VLSI-A Compendium of Approaches. *Journal of Electronic Testing* 12, 7 (1998).
9. Carter, W. C.; Schneider, P. R. Design of dynamically checked computers. In: *Proceedings of 4th IFIP Congress*. 878–883 (1968).
10. Violante, M.; Meinhardt, C.; Sonza Reorda, M.; Reis, R. A Low-Cost Solution for Deploying Processor Cores in Harsh Environments. *IEEE Transactions on Industrial Electronics 58*, 2617 (2011).
11. Verdel, T.; Makris, Y. Duplication-based concurrent error detection in asynchronous circuits: shortcomings and remedies. In: *Proceedings of 17th IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems-DFT*. 345–353 (2002).
12. Drineas, P. and Makris, Y. SPaRe: selective partial replication for concurrent fault-detection in FSMs. *IEEE Transactions on Instrumentation and Measurement* 52, 818733 (2003).
13. Hamming, R. W. Error detecting and correcting codes. *Bell System Technical Journal 29, 147 (1950)*.
14. Peterson, W. *Error-correcting codes* (The Mit Press, Cambridge, 1980).
15. Von Neumann, J. Probabilistic logics and synthesis of reliable organizms from unreliable components. In: *Automata Studies*. 43–98 (NJ: Princeton Univ. Press, New York, 1956).
16. Pratt, B. et al., Fine-Grain SEU Mitigation for FPGAs Using Partial TMR, *IEEE Transactions on Nuclear Science* 55, 2274 (2008).
17. Lima Kastensmidt, F.; Sterpone, L.; Sonza Reorda, M.; Carro, L. On the Optimal De-sign of Triple Modular Redundancy Logic for SRAM-Based FPGAs. In: *IEEE Proc. Design, Automation and Test in Europe Conference*. 1290–1295 (2005).
18. Sterpone, L.; Violante, M. A New Algorithm for the Analysis of the MCUs Sensitiveness of TMR Architectures in SRAM-Based FPGAs. *IEEE Transactions on Nuclear Science* 55, 2019 (2008).
19. Carmichael, C. Triple Module Redundancy Design Techniques for Virtex® Series FPGA. *Xilinx Application manual XAPP 197* (2000).
20. Heiner, J.; Collins, N.; Wirthlin, M. Fault Tolerant ICAP Controller for High-Reliable Internal Scrubbing. In: *IEEE Aerospace Conf.* 1–10 (2008).
21. Chapman, K. SEU strategies for Virtex-5 Devices. *Xilinx Application manual XAPP864* (2010).
22. Asadi, H.; Tahoori, M. B. Soft error mitigation for SRAM-based FPGAs. In: 23rd *IEEE VLSI Test Symp. 207–212* (2005).
23. Hulme, C. A.; Loomis, H. H.; Ross, A.; Rong Yuan, A. Configurable fault-tolerant processor (CFTP) for spacecraft onboard processing. In: Proc. *IEEE Aerospace Conf. 2269–2276* (2004).
24. Berg M. et al. Effectiveness of Internal Versus External SEU Scrubbing Mitigation Strategies in a Xilinx FPGA: Design, Test, and Analysis. *IEEE Transactions on Nuclear Science* 55, 2259 (2008).
25. Legat, U.; Biasizzo, A.; Novak, F. Self-reparable system on FPGA for single event upset recovery. In: *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*. 1–6 (2011).
26. Legat, U.; Biasizzo, A.; Novak, F. Automated SEU fault emulation using partial FPGA reconfiguration. *Design and Diagnostics of Electronic Circuits and Systems (DDECS)*. 24–27 (2010).
27. Legat, U.; Biasizzo, A.; Novak, F. A compact AES core with on-line error-detection for FPGA applications with modest hardware resources. *Microprocessors & Microsystems* 35, 405 (2011).
28. Novak, F.; Biasizzo, A. Security Extension for IEEE Std 1149.1. *Journal of Electronic Testing 22*, 301 (2006).

Arrived: 16. 08. 2012
Accepted: 23. 09. 2012