International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 24 Issue 5 – APRIL 2017.

Verification of Fault Tolerant Techniques in Finite State Machines using Simulation Based Fault Injection Targeted at FPGAs for SEU Mitigation

T. S. Nidhin[1], Anindya Bhattacharyya[2], R. P. Behera[3], T. Jayanthi[4], K. Velusamy[5]

[1]Senior Research Fellow, [1,2,3,4,5]Indira Gandhi Centre for Atomic Research, [1]Homi Bhabha National Institute, Kalpakkam, Tamilnadu-603102, India

***Abstract*— Field Programmable Gate Arrays (FPGA) are susceptible to soft errors due to the shrinkage of feature size and reduction in core voltage which reduces the critical charge required to change the state of a circuit element. To improve the reliability and availability of the FPGA based designs used in Nuclear Power Plants special care has to be taken against these emerging risks. In this paper, the effects of radiation on Finite State Machines (FSM) is reviewed and resource utilization and performance penalty are analyzed by using the fault tolerant techniques like Triple Modular Redundancy (TMR), Hamming-3 encoding and safe FSM synthesis. A novel scripting based fault injection technique is proposed for verifying the fault tolerant techniques at netlist level. The PREP3 state machine is used as a benchmark circuit in this paper. This work predominantly focuses on the practical use of fault tolerant techniques such as TMR, Error Detection and Correction by using Hamming-3 encoding for state register and Safe FSM implementation in live designs targeted at Nuclear Power Plants in India. The major objective of this work is to review the various field proven fault tolerant techniques targeted at FPGAs and develop a simple scalable methodology for verification of the same.**

***Index Terms*— FPGA, Single Event Upset (SEU), Fault tolerance, Finite State Machine, TMR, Fault Injection.**

I. INTRODUCTION

The interest towards the use of Hardware Programmable Device (HPD) technologies like Field Programmable Gate Array (FPGA) and Complex Programmable Logic Device (CPLD) in Nuclear Power Plant (NPP) safety automation is rising internationally as the various advantages of the HPDs over the currently used analog or microprocessor-based software technology are being recognized [1]. The high complexity and the difficulties in demonstrating the safety of software-based applications are among the reasons that have led the nuclear power utilities and Instrumentation and Control (I&C) system developers to look for new technologies such as the HPDs for implementing the safety automation applications [1]. HPDs offer greater simplicity and less burdensome regulatory approval because the end product can be designed to be purely hardware, with independent, parallel signal paths similar to conventional analog electronics. As the feature sizes of FPGA technologies become smaller, new challenges emerge in the area of reliability [2]. New generation transistors are characterized by a reduction in core voltage, a decrease in transistor geometry and an increase in switching speeds. Due to these features, random noise and signal integrity problems including inductive or capacitive crosstalk can be sources of errors in electronic circuits, especially if such effects are not taken into account in the circuit design [3], [4]. Radiation effects are increasingly seen as a major contributor to the overall error rate [4]. Mitigation of radiation effects has been required for Military/Aerospace applications for decades because of the intense radiation fields in which such applications must function. More recently, radiation effects have become a concern for terrestrial applications such as medical, automotive and safety critical applications. This problem necessitates the use of fault tolerant techniques for radiation induced effects in HPDs consisting of FPGAs [5].
The rest of the paper is organized as follows. Section II depicts the SEU mechanism and susceptibility of current FPGA technologies. Section III gives a survey on fault tolerant design techniques used for FSM designs along with verification methodologies based on fault injection. The state machines used for analysis are illustrated in section IV. The proposed fault injection method is explained in section V. Synthesis and simulation results are depicted in section VI and the paper is concluded with future scope in section VII.

II. SINGLE EVENT UPSET MECHANISM AND CURRENT FPGA TECHNOLOGIES SUSCEPTIBILITY TO SEU

The current FPGA technologies can be divided into three major categories according to the type of routing fabric used in the devices. SRAM-based FPGAs takes advantage of latest fabrication processes and offers much higher integration and logic capacity when compared to flash/antifuse based processes which are typically two generations behind the pure CMOS processes. For typical reactor applications, Flash or antifuse based technologies are preferred over SRAM FPGAs and hence in this paper designs targeted at flash based FPGAs are discussed.

A. SEU Mechanism

Chip packaging materials, in general, contain small amounts of radioactive contaminants that can cause soft errors through alpha-particle emission [6]. The positively charged alpha particle travels through the semiconductor and disturbs the distribution of electrons within the device. Controlling alpha particle emission rates for critical packaging materials to less than a level of 0.001 counts per hour per square centimeter (cph/cm2) is required for the reliable performance of most circuits. By comparison, the count rate of a typical sole of a shoe is between 0.1 and 10cph/cm2 [7], [8]. Cosmic-ray flux another dominant cause of errors creates a shower of energetic secondary particles. Cosmic ray particle flux depends on altitude, the rate of upsets in aircraft can be more than 300 times the upset rate at sea level [8]. Logic circuits with a higher capacitance and higher logic voltage levels are less likely to suffer an error [3].

Critical charge defined as the minimum amount of induced charge required at a circuit node to cause a voltage pulse to propagate from that node to the output and to be of sufficient duration and magnitude to disturb a memory element varies greatly depending on the technology. For example, in SRAM-based FPGAs, the SRAM cell storing the configuration bit is typically much smaller (that is, lower capacitance) than a normal flip-flop. Because the internal nodes in an SRAM cell configuration have a smaller capacitance, the critical charge required for upsetting the configuration bits is much smaller than that of a flip-flop. As described later in this document, this effect will influence the strategy employed to mitigate errors due to radiation.

B. SEU on FPGA

The radiation induced soft errors are the events in which the data is corrupted, but the device itself is not permanently damaged [9]. Soft errors can be categorized as Single Event Transients (SET) and Single Event Upsets (SEU). In SET, a node in the circuit temporarily holds an incorrect value. Normally SETs are transitory in nature where the functionality of the circuit is unaffected. However the transient can be captured into flip-flops or other memory elements can lead to functional failure of the system. Systems working on high frequencies are more susceptible to SET captures because the probability of capture increases with frequency. In SEUs the charge collected by the energetic particle strike is greater than the critical charge and causes a change of state of a memory cell, register, latch or flip-flop. The radiation induced susceptibility matrix is given in table 1.

TABLE I. FPGA SUSCEPTIBILITY MATRIX

Resource	Flash	SRAM
I/O	No	Low
BRAM	Very high	Very high
Registers	Medium	Medium
Logic Cells	No	High
Routing matrix	No	High

III. PREVIOUS WORK

The control parts of most FPGA based designs are built by Finite State Machines and any bit-flips due to Single Event Effects (SEE) in FSMs can adversely affect the performance and reliability of the overall system [10].

A. Survey of Design Techniques

There are varieties of issues regarding the design of finite state machine using the hardware description languages. The areas to be focused are coding style, state encoding schemes, decoding logic and output generation as outlined in [11]. There are multiple fault tolerant techniques available; Triple Modular Redundancy (TMR) is the standard for system level architectures in safety critical systems. Duplex architecture, Explicit Error Correction (EEC) architecture, Modified EEC and Implicit Error Correction can also be used with TMR Architecture [12] for fault tolerance. These architectures particularly suitable for VLSI implementation using low power CMOS technology are identified, with single flip-flop errors. TMR with Error Correction Codes is explained in [13] and TMR with partial reconfiguration [14] are also commonly used for improving the reliability. Error Detection and Correction (EDAC) codes can be used as an efficient fault tolerant technique for memory elements and message passing circuits. Most popular single error correcting code is Hamming codes [15]. Single Error Correction (SEC) code with minimum Hamming distance of three is used with different architectures such as Single Independent Decoder (SID) architecture, Distributed Error Correction (DEC) architecture, Upset-oriented SID (UPS) and Upset-oriented DEC (UPD) to achieve fault tolerance in FSMs [16].

In the fault tolerant and fail safe designs the selection of encoding style play a significant role on the dependability of the state when a soft error or SEU occurs. The most commonly used encoding techniques are Sequential or Binary code, One-Hot code and Gray code. One-Hot encoding provides an optimal design in terms of area, performance and reliability for most of the FSM implementations [17]. A simple parity calculation on the state register can detect the onset of SEU and mitigative action can be taken. Safe state machines are critical in high-reliability designs. Synthesis tools, by default, perform reachability analysis and optimize unused states and logic. Designers get lulled into a sense of false security by writing the "default/others" clause in HDL codes which during synthesis is optimized away rendering the state machine unsafe.

Other techniques as reported in the literature are Mapping of FSM into Synchronous Embedded Memory Block (SEMB) enhances the runtime reliability without a significant increase in power consumption [18]. A functional decomposition concept for implementing FSM into embedded memory can also be used [19]. Duplication with Comparison (DWC) and Concurrent Error Detection (CED) are the other popular methods [20], [21]. DWC is targeted only at SRAM devices and is used as a trigger for partial reconfiguration or memory scrubbing. The reliability of sequential circuits can be improved by adding redundant equivalent states to the states with a high probability of occurrence [22]. Temporal Data Sampling and Majority voting have also been explained as a mitigation technique for SEUs [23]. The Single event upset mitigation techniques for configuration memory of SRAM based FPGAs are explained detail in [24].

B. *Survey of Design Techniques*

There are varieties of issues regarding the design of finite state machine using the hardware description languages. The areas to be focused are coding style, state encoding schemes, decoding logic and output generation as outlined in [11]. There are multiple fault tolerant techniques available; Triple Modular Redundancy (TMR) is the standard for system level architectures in safety critical systems. Duplex architecture, Explicit Error Correction (EEC) architecture, Modified EEC and Implicit Error Correction can also be used with TMR Architecture [12] for fault tolerance. These architectures particularly suitable for VLSI implementation using low power CMOS technology are identified, with single flip-flop errors. TMR with Error Correction Codes is explained in [13] and TMR with partial reconfiguration [14] are also commonly used for improving the reliability. Error Detection and Correction (EDAC) codes can be used as an efficient fault tolerant technique for memory elements and message passing circuits. Most popular single error correcting code is Hamming codes [15]. Single Error Correction (SEC) code with minimum Hamming distance of three is used with different architectures such as Single Independent Decoder (SID) architecture, Distributed Error Correction (DEC) architecture, Upset-oriented SID (UPS) and Upset-oriented DEC (UPD) to achieve fault tolerance in FSMs [16].

In the fault tolerant and fail safe designs the selection of encoding style play a significant role on the dependability of the state when a soft error or SEU occurs. The most commonly used encoding techniques are Sequential or Binary code, One-Hot code and Gray code. One-Hot encoding provides an optimal design in terms of area, performance and reliability for most of the FSM implementations [17]. A simple parity calculation on the state register can detect the onset of SEU and mitigative action can be taken. Safe state machines are critical in high-reliability designs. Synthesis tools, by default, perform reachability analysis and optimize unused states and logic. Designers get lulled into a sense of false security by writing the "default/others" clause in HDL codes which during synthesis is optimized away rendering the state machine unsafe.

Other techniques as reported in the literature are Mapping of FSM into Synchronous Embedded Memory Block (SEMB) enhances the runtime reliability without a significant increase in power consumption [18]. A functional decomposition concept for implementing FSM into embedded memory can also be used [19]. Duplication with Comparison (DWC) and Concurrent Error Detection (CED) are the other popular methods [20], [21]. DWC is targeted only at SRAM devices and is used as a trigger for partial reconfiguration or memory scrubbing. The reliability of sequential circuits can be improved by adding redundant equivalent states to the states with a high probability of occurrence [22]. Temporal Data Sampling and Majority voting have also been explained as a mitigation technique for SEUs [23].

A. *Survey of Fault Injection Techniques*

To evaluate the sensitivity of a design to soft errors the design has to be verified thoroughly either at simulation level or validated at FPGA level on the test board. This entails the use of fault injection methodology. Fault injection is well documented in the literature as a verification/validation technique for characterizing the reliability of HPDs. VHDL-based Evaluation of Reliability by Injecting Faults efficientlY (VERIFY) introduces a new way for defining the behavior of hardware components in case of faults by extending the VHDL language with fault injection signals together with their rate of occurrence [25]. Autonomous Multilevel emulation-based fault injection for Soft Error Evaluation (AMUSE) is a method that can inject SET faults by integrating both Register Transfer Level (RTL) and netlist level [26]. An easy to develop and flexible FPGA fault injection technique which utilizes the debugging facilities of Altera FPGA in order to inject SEU and MBU fault models in flip-flops and other memory units is presented in [27]. Another technique based on simulator commands, saboteurs and mutants are presented in [28]. There are many other simulation/emulation or hybrid fault-injection tools and methods available which include a method/tool called NETFI (NETlist Fault Injection) it can inject fault in any HDL model, VHDL, Verilog etc. [29]. In addition, to the speed and automation, NETFI can target the block RAMs of a given circuit.

IV. BENCHMARK FSM FOR ANALYSIS

PREP3 benchmark was taken up as a primary benchmark for initial analysis followed by PREP4 and two other general designs. The PREP3 is a mealy state machine with eight inputs and eight outputs, which has eight states and twelve transitions. PREP4 is a large mealy FSM with 16 states and 43 transitions. All the FSMs are implemented using VHDL coding and synthesized for a flash based FPGA for indication purpose.

V. DEVELOPMENT OF SCRIPT BASED FAULT INJECTION TECHNIQUE

The major challenge in any fault tolerant design technique is the methodology used for verification or validation for quality assurance of the final product. The methodology used is often too complex and customized to be used across a substantially big project with multiple designs with different specifications. We propose a simple TCL script based automated fault injection methodology built around the target simulator which can take designs in both RTL and netlist level of abstraction. The proposed methodology parses a design in a guided or automated manner selecting sensitive nodes where the fault is to be injected and generating a TCL script for the same. The "force –freeze" command is used to change the value of any signal/wire. The value can be made stuck or frozen at either '1' or '0' for any particular period of time. For example; "force -freeze sim:/test_prep3/I1/CURRENT_STATE(7) 1 {200 ns} -cancel {250 ns}" stuck the value of 7th bit of register CURRENT_STATE as '1' for a duration of 50 nanoseconds. The process of generating the TCL file containing the fault injection is automated and can be coded in Perl/Python/.net or any other suitable language. The algorithm proposed here parses and finds the nodes with maximum fanouts so a single injected fault caused maximum damage to the functionality of the design.

As shown in fig. 1 the verification setup consists of a self checking assertion based test bench which is used for generating the stimulus and counting the number of errors. If there is any mismatch in the values than the expected, fault counter counts an error.

For indication purposes only, in this paper the fault is injected at the netlist level, particularly focused on state registers of the FSM as the target FPGA is only sensitive to SEUs at the register level. For the change in the technology of target to SRAM, the methodology is flexible can be easily modified to inject fault at combinational circuits which are implemented in look-up tables (LUT). Simple designs have been taken up for proving the effectiveness of the methodology.

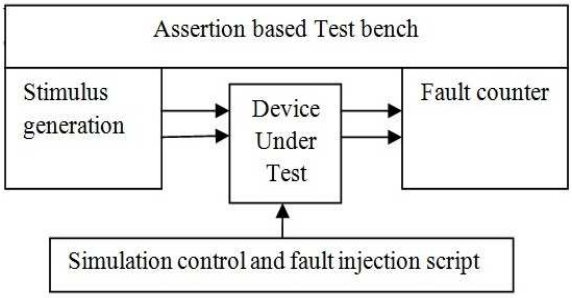

Fig. 1. Verification setup block diagram

VI. Synthesis and Simulation Results

PREP3, PREP4, and two more simple FSM designs are considered for analyzing the resource utilization and timing characteristics. The increase in the percentage of resource utilization and the decrease in timing performance is compared to the normal design is shown in Fig. 2 and Fig. 3. The increased area for Hamming code, when compared to TMR is attributed to the fact that TMR for flash FPGAs are implemented for the state registers only, not the combinational logic part of the design. Combinational logic in flash based FPGAs are implemented using multiplexers which are immune to SEUs, whereas for SRAM-based FPGAs the entire logic has to be triplicated giving a substantial rise in the resource utilization. Safe FSMs also shows a marked increase in resource utilization and is a major reason why synthesizing tools remove the excess logic associated unless the 'safe' attribute for FSM is used. The reduction in frequency is maximum for Hamming-3 implementation. So it is noted that TMR method for flash based FPGAs are superior when compared to the other two in terms of resource utilization and timing while Hamming will have an advantage of indicating and correcting single bit error in state register which can be used to take the FSM to a safe state.

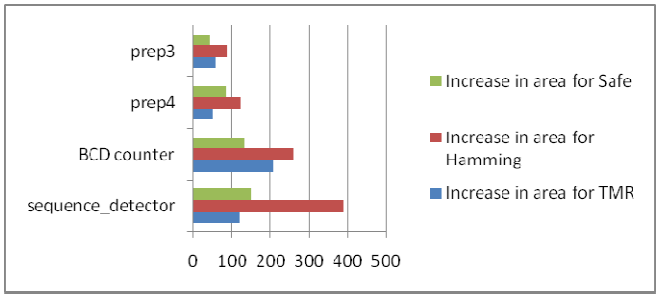

Fig. 2. Resource utilization increase in percentage.

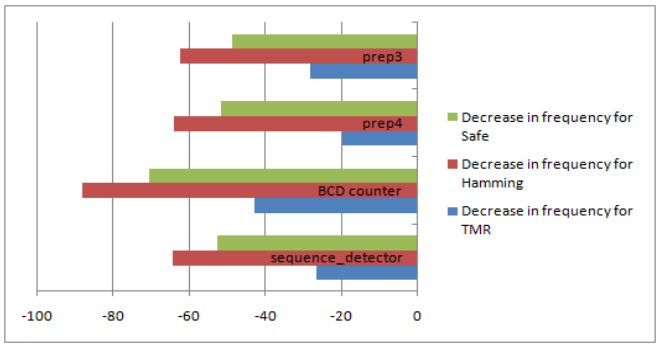

Fig. 3. Maximum frequency reduction in percentage.

The value of the state register is changed from '1' to '0' or '0' to '1' for particular time periods and the number of errors generated due to the injected fault is measured by the self checking assertion based test bench simulating the netlist file. This is repeated for all the netlist files which are generated after implementing the fault tolerant techniques and the results are analyzed. The technique developed can be suitably modified to parse the netlist files and pick up random signals/nets for fault injection and checking.

Fig. 4 shows the simulation waveform of PREP3 FSM after injecting fault in the state and output registers randomly. Total of 24 faults injected into the signals CURRENT_STATE(7) to CURRENT_STATE(0) and OUTT_I_C(7) to OUTT_I_C(0) in random time intervals and it has generated 20 errors.

Fig. 5 shows the waveform for TMR. Faults are injected to the state and output registers of one of the TMR logic blocks. It uses 2 out of 3 voting logic so it has corrected all the faults injected in one TMR block. The fault injected in more than one TMR logic block and voting logic cannot be corrected by TMR method, this was analyzed and verified by injecting fault in these blocks.

By injecting fault in one of the state registers, CURRENT_STATE_DUP(7) values is forced to high from 100ns to 150ns has generated an unreachable state. The safe FSM implementation forced the state machine into a reset state is shown in fig. 6. Hamming distance 3 code is used as a mitigation technique for single bit errors only. Two errors are injected into CURRENT_STATE(6) has generated 3 errors; because Hamming distance 3 code cannot correct more than one error is shown in fig. 7.

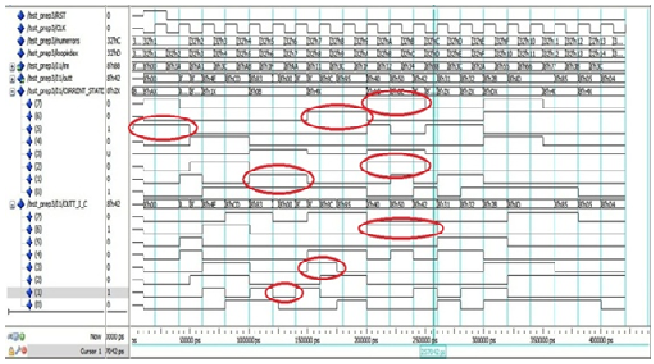

Fig. 4. The Waveform of fault injection without using any fault tolerant methods.

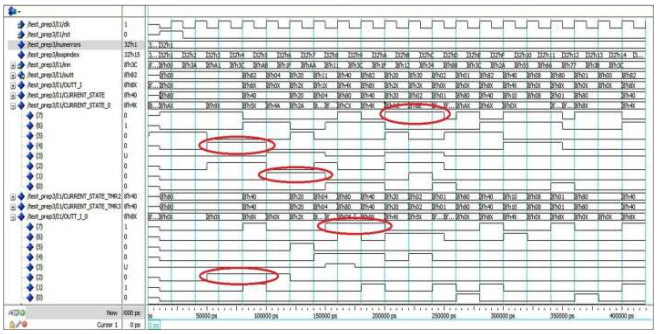

Fig. 5. The Waveform of fault injection in one of the TMR logic.

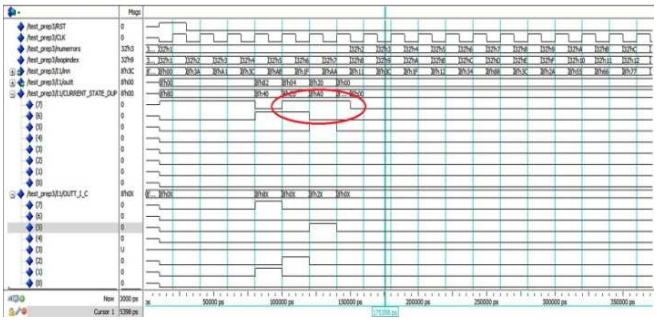

Fig. 6. The Waveform of fault injection after implementing safe FSM.

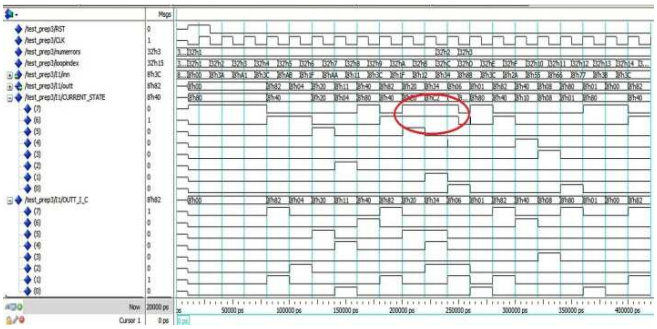

Fig. 7. The Waveform of 2 errors injected after Hamming 3 implementation.

VII. CONCLUSION

Most of the simulation based fault injection techniques are developed using VHDL/Verilog coding, TCL scripting based technique makes it simpler. This technique is the best alternative for other state of the art simulation based fault injection techniques in terms of ease of development and implementation. It provides better platform independency, controllability and observability.

Various fault tolerant FSM design techniques and fault injection techniques were discussed for different target FPGA architectures like SRAM and Flash. A Flash based FPGA was used as a notational target device and results were presented in terms of resource utilization and timing. It is found that the TCL scripting based fault injection tool could able to inject any number of faults at netlist level efficiently. From the data presented it is concluded that TMR gives the best performance in terms of area and timing. Hamming-3 encoding shall only be used when the probability of single bit upset exists and safe FSM implementation takes the FSM to a fail-safe state when an invalid state occurs. Future work involves the development of an efficient fault tolerant design technique and validation of the designs through irradiation experiments with neutrons and heavy ion beams.

ACKNOWLEDGMENT

The first author gratefully acknowledges the grant of the research fellowship from the Department of Atomic Energy, Government of India.

REFERENCES

[1] Jukka Ranta “The current state of FPGA technology in the nuclear domain” published by VTT Technical Research centre of Finland, March 2012.

[2] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, L. Alvisi, “Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic,” Proc. of the Intl. Conf. on Dependable Systems andNetworks (DSN’02), Washington D.C., June 2002.

[3] A.H Johnson “Radiation effects in Advanced Microelectronic Technologies” IEEE TRANSACTIONS ON NUCLEAR SCIENCE,VOL.45,NO.3,JUNE 1998.

[4] Raoul Velazo, Pascal Fouillat, Ricardo Reis “Radiation effects on Embedded Systems” book published by Springer, 2007.

[5] Frank Hall Schmidt, Jr. “Fault Tolerant Design Implementation on Radiation Hardened By Design SRAM-Based FPGAs” thesis submitted at the “MASSACHUSETTS INSTITUTE OF TECHNOLOGY”.

[6] Robert C. Baumann “Radiation-Induced Soft Errors in Advanced Semiconductor Technologies” IEEE TRANSACTIONS ON DEVICE AND MATERIALS RELIABILITY, VOL 5, NO.3, SEPTEMBER 2005.

[7] Santosh kumar, Shalu Agarwal and Jae Pil Jung “ Soft Error issue and importance of low alpha solders for Microelectronics packaging” Rev. Adv. Mater. Sci. 34 (2013) 185-202.

[8] Jiri Kvasnicka “Reliability Analysis of SRAM-based Field Programmable Gate Arrays” Ph.D. Thesis submitted to Chech technical University in Prague, August 2013.

[9] Michael Nicolaidis “ Soft errors in modern electronic systems” Springer; 2011 edition.

[10] Chu, pong P, “RTL hardware design using VHDL”, A Wiley InterScience Publication, USA, ISBN: 978-0-471-72092-8, pp.313-373,2006.

[11] Iuliana Chiuchisan. Alin Dan Potorac, Adrian Graur, “Finite State Machine Design and VHDL Coding Techniques”, 10th international conference on development and application systems, Suceava, Romania, May 27-29, 2010.

[12] Shailesh Niranjan , James F. Frenze l “A Simplified Approach to Fault Tolerant State Machine Design for Single Event Upsets” IEEE TRANSACTIONS ON RELIABILITY, VOL. 45. NO. 1, 1996 MARCH.

[13] Fernanda Gusmao de Lima,Thesis on “Designing single event upset mitigation techniques for large SRAM-based FPGA devices” Porto Alegre, February 11th, 2002.

[14] Carl Carmichael “Triple Module Redundancy Design Techniques for Virtex FPGAs”, Xilinx application note (virtex series).

[15] R. W. Hamming, “Error detecting and correcting codes,” Bell Syst.Tech. J., vol. 29, pp. 147–160, Apr. 1950.

[16] R. Rochet, R. Leveugle, G. Saucier “Analysis and Comparison of Fault Tolerant FSM architectures based on SEC codes” International Workshop on Defect and Fault Tolerance in VLSI Systems, IEEE, 1993.

[17] Maico Cassel, Fernanda Lima Kastensmidt “Evaluating One-Hot Encoding Finite State Machines for SEU Reliability in SRAM-based FPGAs” Proceedings of the 12th IEEE International On-Line Testing Symposium (IOLTS'06).

[18] Anurag Tiwari, Karen A. Tomko, “Enhanced Reliability of Finite-State Machines in FPGA through Efficient Fault Detection and Correction” IEEE TRANSACTIONS ON RELIABILITY, VOL. 54, NO. 3, SEPTEMBER 2005.

[19] Henry Selvaraj, Mariusz Rawski, Tadeusz Łuba “FSM Implementation in embedded memory blocks of programmable logic devices using functional decomposition” Proc. Int. Conf. on Information Technology: Coding and Computing, pp. 355-360, 2002.

[20] Fernanda Lima, Luigi Carro, Ricardo Reis "Designing Fault Tolerant Systems into SRAM-based FPGAs", DAC'03, June 2-6, 2003, Anaheim, California, USA.
[21] Andrzej Krasniewski "Concurrent Error Detection for FSMs Designed for Implementation with Embedded Memory Blocks of FPGAs" 10th Euromicro Conference on Digital System Design Architectures,IEEE,2007.
[22] Aiman H. El-Maleh, Ayed S. Al-Qahtani "A finite state machine based fault tolerance technique for sequential Circuits" Microelectronics Reliability-54 (2014) 654-661, Elsevier.
[23] S. Baloch, T. Arslan, A. Stoica 2,3 "Design of a Single Event Upset (SEU) Mitigation Technique for Programmable Devices" Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED'06), IEEE-2006.
[24] T. S. Nidhin, Anindya Bhattacharyya, R. P. Behera, and T. Jayanthi," A Review on SEU Mitigation Techniques for FPGA Configuration Memory", IETE Technical Review, 23 January 2017. doi:10.1080/02564602.2016.1265905
[25] V. Sieh, O et al., "VERIFY: evaluation of reliability using VHDLmodels with embedded fault description", Proc. of the International Symposium on Fault-Tolerant Computing, Jun. 1997, pp. 32-36.
[26] L. Entrena et al., "Soft Error Sensitivity Evaluation of Microprocessors by Multilevel Emulation-Based Fault Injection, Trans. On Computers, pp.313-322, 2012.
[27] Mojtaba Ebrahimia, Abbas Mohammadi, Alireza Ejlali, Seyed Ghassem Miremadi, "A fast, flexible, and easy-to-develop FPGA-based fault injection technique" Microelectronics Reliability 54 (2014) 1000–1008,Elsevier.
[28] D. Gil, J. Gracia, J.C. Baraza, P.J. Gi," Study, comparison and application of different VHDL-based fault injection techniques for the experimental validation of a fault-tolerant system" Microelectronics Journal 34 (2003) 41–51,Elsevier.
[29] Wassim Mansour, Raoul Velazco, "An automated SEU fault-injection method and tool for HDL-based designs".