International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 1, January 2013 ISSN 2319 - 4847

Fault Tolerant Techniques for Reconfigurable Devices: a brief Survey

Mrs.Jamuna.S[1], Dr. V.K. Agrawal[2]

[1]Associate Professor, Department of ECE, DSCE, Bangalore, India

[2]Director, CORI, PESIT, Bangalore

ABSTRACT

Reconfigurable devices are the programmable devices used to implement complex functions in applications like space missions, communication systems, nuclear systems and adaptive computing systems etc. They are popularly known as the field programmable gate arrays (FPGAs). When FPGAs are used in such mission critical applications, reliability of the hardware is very much important. The system should be designed as a fault tolerant system (FT). A suitable fault handling mechanism should be incorporated while designing reliable systems. Fault tolerance is the ability of a system to operate normally given the presence of malfunctioning resources, faults or defects. SRAM-based FPGAs has made it possible to incorporate fault tolerance into systems at cheaper cost. Various FT methods have been developed for the FPGAs. In this paper we have tried to present some of them.

Key words— BIST, CLB, FPGAs, FT, SEU, TMR

1. INTRODUCTION

Today FPGAs are widely used in product prototyping and development because of their ability of configuration and re-configuration. FPGAs have a regular structure of logic blocks and interconnect which facilitates for making it a fault tolerant system. With the development in the integrated circuit technology and increase in logic density, FPGAs have become more vulnerable to faults like any other IC chips. SRAM based FPGAs are prone to both transient (single event upsets) and permanent faults. Faults may occur anywhere in the device. Many researchers have developed techniques for handling these faults. FT system design involves mainly three phases, fault detection, diagnosis and correction of faults. Testing is done for finding the faulty component and diagnosis is needed for locating the fault. Once faulty part is identified, repair process is done without disturbing normal function of the system. Factors considered by researchers while developing FT technique are i) area overhead ii) latency period iii) reliability etc

This paper is organized to give the overview of FPGA architecture and information about faults in section II, brief explanations about available FT techniques are presented in section III and Paper concludes in section IV.

2. OVERVIEW OF FPGA ARCHITECTURE

There are several families of FPGAs available from different semiconductor companies. These device families slightly differ in their architecture and feature set, however most of them follow a common approach: A regular, flexible, programmable architecture of Configurable Logic Blocks (CLBs), surrounded by a perimeter of programmable Input/output Blocks (IOBs). These functional elements are interconnected by a powerful hierarchy of versatile routing channels.

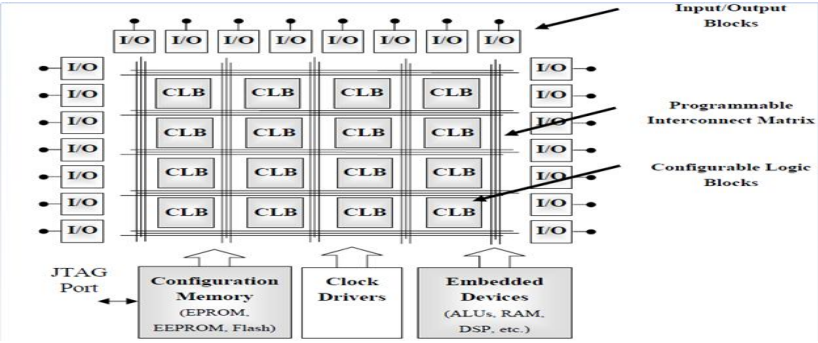

Fig1. FPGA architecture

Fault & Fault Models

A fault is the representation of a defect reflecting a physical condition that causes a circuit to fail to perform in a required manner. A failure is a deviation in the performance of a circuit or system from its specified behaviour and represents an irreversible state of a component such that it must be repaired in order for it to provide its intended design function. Faults can be divided into two categories:

1. **Permanent Faults:** Permanent faults may occur because of electro migration or with the aging of the device; small manufacturing imperfections that are not detected by production testing may become effective during the lifetime. Design errors can also cause a device to stop functioning in response to rare sequences of input [1].

2. **Transient Faults**: Transient faults also called as soft errors are mainly due to environmental conditions. These faults occur because of energetic nuclear particle or electrical source [2] .The nuclear particles which cause errors are either cosmic ray which bombards the earth constantly from space radiations. Power supply noise or electromagnetic interference (EMI) may also induce the errors.

Fault models : Faults can occur anywhere across the FPGAs resources (CLBs or the interconnect).To effectively evaluate the quality of a set of tests for a product, as well as to evaluate the effectiveness of a built-in self test (BIST) approach in its application to that product, fault models are required for emulation of faults or defects in a simulation environment. Fault models are necessary for generating and evaluating a set of test vectors. Various fault models assumed in the literature are[3] I) stuck- at fault model across look up tables in the CLBs (s-a-0/s-a-1). II) Transistor stuck-on/off fault model across the switch matrix.

III) Wire open /short on interconnect IV) delay fault model.

3. FAULT TOLERANT TECHNIQUES

Fault tolerance is the ability of any system to work normally even in presence of faults. In order to tolerate faults the system should have an efficient mechanism for managing the faults which occur during its lifetime. A regular array structure and ability to reconfigure in FPGAs facilitates for the implementation of fault tolerance. FT design has two stages. i) Fault detection & diagnosis and ii) Fault repair process. A lot of research is been done in fault detection and diagnosis for internal resources of FPGAs. Detail explanation can be found in [4].Some methods have been developed for detecting faults only across the CLBs and others have exclusively developed for the interconnect faults.

In this paper we have concentrated only on various fault repair mechanisms. Many people have developed techniques for providing the FT in FPGAs. In the literature we find the general methods of FT techniques are redundancy based , on-line test based and evolvable hardware based[5].These techniques can be broadly classified in to two categories depending on the type fault recovery mechanism adopted.[6]

1. Hardware level fault tolerance 2.Configuration level fault tolerance.

Hardware level FT mainly concentrates on defects which occur during fabrication process. It considers the spare resources for fault repair where as the second group makes changes in the configuration data which is mainly responsible for realising any logic. Methodologies as in [7],[8], [9], [10], [11] belong to hardware level FT design. [7] and [8] gives a method based on row/column shifting. They target defects which occur during manufacturing and not on the faults which occur in the field. Fabricated extra rows and columns will act as spare resources. Repair process is done by a controller in the form of a multiplexer which avoids a faulty row or a column. The approach requires that the test circuit to be fabricated within the architecture. Number of faults tolerated depends on the extra rows/ columns. [9] Gives an approach for defect tolerance in a bus based FPGAs. It mainly concentrates on the interconnect faults. [10] and [11] methods are same as row / column shifting but add more by pass connections and help in tolerating more faults.

Configuration level FT methods are the most widely used. Many researchers have developed FT approaches for configuration level of FPGAs. It mainly depends on two factors. They are reconfiguration and availability of unused resources in device. Implementation of configuration level FT has different forms. It can be realised as,

i) Pre- compiled alternative configurations.

ii) Incremental mapping, placement and routing.

iii) Chain shifting

iv) Partial reconfiguration

The pre-compiled configuration based technique [12] creates alternative configurations at design time that use different equivalent columns of FPGA resources. In their non-overlapping scheme, which has the least resource overhead, a total of $C(k+m, m) = (k+m)! / (m!k!)$ Configurations are required to tolerate faults in m columns, where k is the number of columns in the base configuration. The required design-time effort for this approach is high, as it requires

manual modification of the design to fit into column sets. Also, the number of horizontal routes available to the designer is reduced by the resources consumed by the approach.
Spare Resource-based methods such as the one proposed by Lach et al[13], [14], [15] rely on the availability of standby resources of varying granularity to address faults. Lach's deterministic approach provided redundant resources at design time. This approach segments the FPGA into static tiles at design time with a known functionality, some redundant resources, and pre-designed alternate configurations. Multiple configurations for a tile are created. Whenever a fault is detected, spare tiles are selected, but their functionality is predetermined and thus limited. Fig 2.This method cannot be used for multiple fault tolerance as the number of pre compiled configurations will increase rapidly.

Dutt et al[16] provide an incremental re-routing method for increased flexibility to tolerate fault on-the-fly. In this method, the FPGA is initially routed without any extra interconnects for reconfiguration. The technique relies on node-covering in which reconfiguration is achieved by constructing replacement chains of cells from faulty cells to spare or unused cells. Using a cost-directed depth-first search strategy, they minimize the overheads involved in rerouting interconnects when responding to faults.

Other innovative methods to tolerate faults using spare resources include [17] and Lakamraju and Tessier's[18] intra-cluster repair. The authors approach fault tolerance for cluster-based FPGA which group multiple LUT/FF pairs together in clusters. Their method that takes advantage of logical redundancy in such clusters by replacing faulty LUT inputs and logic resources unused in the original design mapping by defining methods for LUT Input Exchange and Basic Logic Element exchange. All these re-routing strategies that involve spare resources require the device to be offline, and the support of an external system to complete the re-routing procedure.

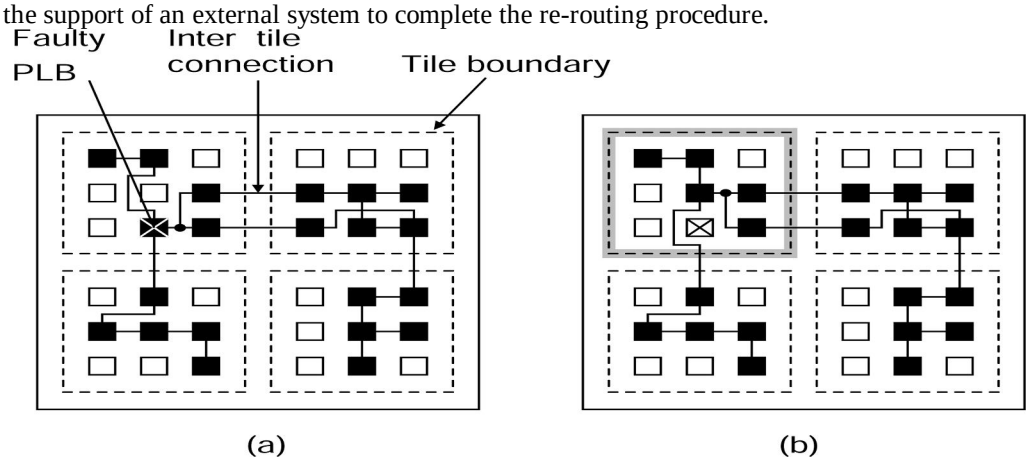

Fig 2[1]. (a) faulty PLB in the tile. (b) PLB is replaced with adjacent PLB

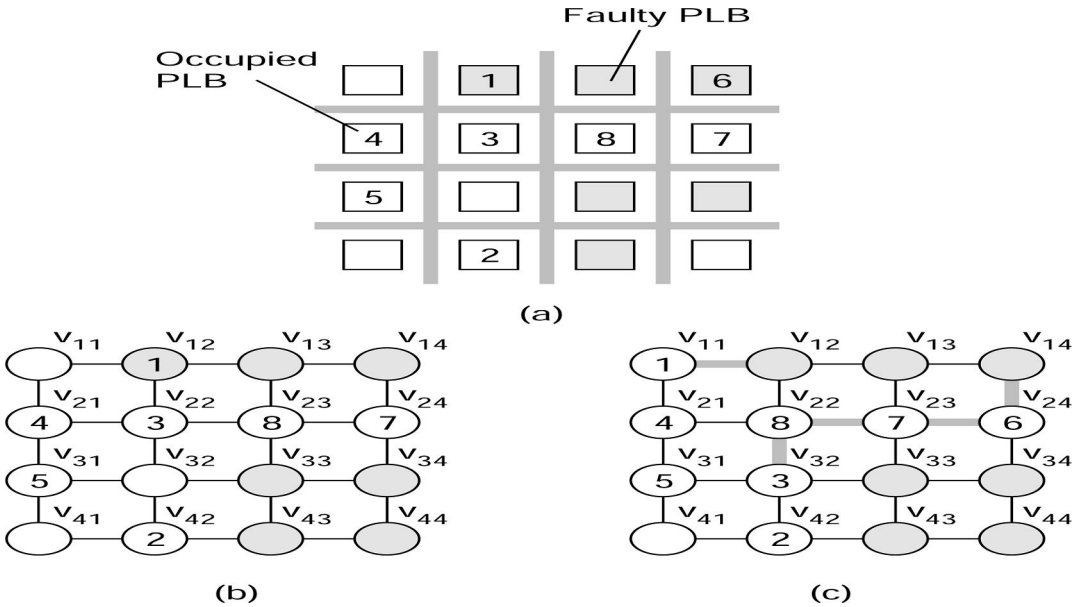

Fig.3[19] Pebble Shifting (a) placed circuit with a faulty PLB (b) graph created from the circuit; (c) reconfigured circuit graph, with shift paths represented by the thick grey lines.

Pebble shifting technique [19]: A method developed for yield improvement. It begins with a circuit that has been placed and routed in an array containing logic faults. The circuit is converted in to a graph with logic blocks being vertices and routing resources being the edges as in fig3. Their algorithm estimates the cost of shifting as a function of additional routing and congestion in routing channels. In this approach an entire faulty cluster is replaced with a spare cluster.

Emmert and Bhatia [20] present a similar Incremental Rerouting approach that does not require design-time allocated spare resources. The fault recovery method assumes an FPGA to contain resources not utilized by the application, thus exploiting unused fault-free resources to replace faulty resources. Upon detecting and diagnosing a logic or interconnection fault by some other detection method, Incremental Rerouting calculates the new logic netlist to avoid the faulty resource. The method reads the configuration memory to determine the current netlist and implements the incremental changes through partial reconfiguration.

Emmert et al. [21] present an approach that pseudo-exhaustively tests, diagnoses, and reconfigures resources of the FPGA to restore lost functionality due to permanent faults. This technique merged the concepts of cluster reconfiguration , pebble shifting and incremental routing and tried to reuse faulty cluster also as a spare for logic expressions which do not require faulty component. The application logic handles transient faults through a concurrent error-detection technique and by periodically saving and restoring the system's state through check pointing.

Chain shifting [22]: this technique is also called as node covering method. In this clusters are arranged as chains with one or more spare clusters provided at the end of chain.

Stroud and Garimella [23] targeted multiple regular structure cores including memories and multipliers and developed a diagnostic procedure based on the extension of the MULTICELLO algorithm. The diagnostic procedure is performed in five steps. They presented a BIST approach in which neighboring blocks are compared by a set of ORAs. Thus, each core is observed by two sets of ORAs and is compared to two different cores.

Sarvi et al [24] developed a diagnostic method to detect and locate faulty embedded cores in FPGAs using BIST. The technique configures the device twice in order to complete fault isolation. The method partitions the cores on an FPGA into two groups and conducts BIST on each of these groups. Fault isolation is achieved by comparing the results of the two tests. Under this scheme the two configurations are constructed to enable isolation by comparison. In post-processing, defectives are identified by analyzing the results of comparisons among blocks enclosed within the same group.

The BIST methods presented by Shantanu Dutt et al [25] are mainly targeted at detection and diagnosis of permanent faults that model fabrication defects as well as other physical defects arising during the lifetime of the FPGA chip. They present 1- and 2-diagnosable BISTer designs that make up a Roving Tester (ROTE). Due to their provable diagnosabilities, these BISTers can avoid time-intensive adaptive diagnosis without significantly compromising diagnostic coverage. Manuel et al [26] describe an active replication procedure that enables a non-intrusive online relocation of active functions implemented in dynamically reconfigurable FPGAs. Their technique allows concurrent structural test of the FPGA. They use Boundary-Scan infrastructure to apply the test vectors and to capture the test responses.

Partial reconfiguration: partial reconfiguration (PR) is an option provided in modern FPGAs. This feature allows a user to modify some part of the design partially without affecting other parts of the design. Accessing configuration memory is necessary for implementing Partial reconfiguration designs on FPGAs. This is done through reconfigurable interfaces like JTAG, select map or ICAP. In order to recover from faults, the failed module may be dynamically repaired, either by reloading the *bitstream* in the FPGA frames that contained the failed module or, if the fault was permanent, by moving the module into other free frames of the FPGA.

Many researchers have utilized PR for developing FT techniques for FPGAs.[27 - 34] . Bolchini, C et al [27] have developed a highly reliable system using TMR concept and partial dynamic reconfiguration. It mainly concentrated on single event upsets. Methodology in [28] divides an application in to small modules called as partial reconfigurable modules (PRMs). partial reconfigurable controller was designed for supervising the reconfiguration process. This technique resulted in good fault localization. But it faces synchronization issues after reconfiguration. In order to provide FT they [29] used two copies of the entire design and defined them as reconfigurable modules as in fig4. Whenever one unit is detected with faults, other copy is selected. Even though area overhead is less compared to TMR technique, it also faces synchronization issues after reconfiguration.

Technique [30] is based on self repair capacity. In order to provide self repair capability they have used two FPGAs. One FPGA is designed as a controller to take care of fault management. Whenever fault occurs in one FPGA, the other will do the reconfiguration process using spare columns. [31] Their work was developed for detecting and tolerating transient faults(SEUs) considering space applications. It was designed to work in two modes as high performance mode and power safe mode. Depending on the density of SEUs reconfiguration switches between these two modes.

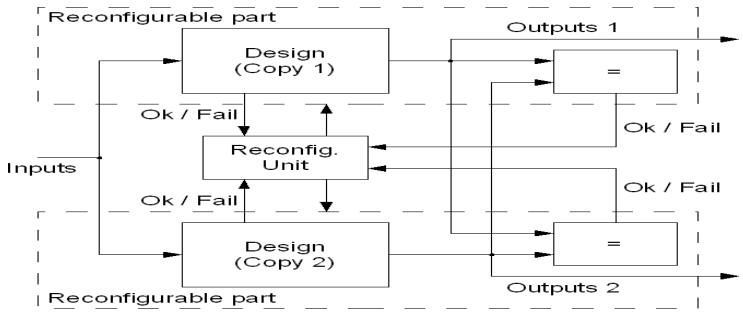

Fig 4. Two copies of the same design configured as reconfigured modules.

A reliable system design was presented in [32]. SEUs causing bit –flips in SRAM memory were detected. Partial dynamic reconfiguration was used for mitigating the effects of faults. Whenever faults were identified recompilation for that particular part was initiated. FT methodology developed by[33] was based on circuit level modifications and was implemented at high level description. They merged again TMR and a combination of duplex with CED techniques. Faults considered were SEUs occurring in combinational and sequential logic. Technique given by [34] is about the design of on-line checkers for detection of errors and used TMR or duplex system for fault recovery. Whenever a fault occurs is detected by on-line checker is made to initiate reconfiguration process in the faulty unit.

4. CONCLUSION

In this paper we have presented a brief explanation about different fault tolerant techniques for SRAM- based FPGAs. Classification of detection methods is explained. Some of the techniques based on partial reconfiguration are also briefly mentioned. However, we feel that more and more On-line testing and fault tolerant techniques should be developed so that FPGAs can be used in complex computing systems (system on chip) and highly reliable applications.

REFERENCES

[1] N. R. Shnidman, W. H. Mangione-Smith, and M. Potkonjak, “On-line fault detection for bus- based field programmable gate arrays,” IEEE Trans. VLSI Syst., vol. 6, pp. 656– 666,Dec. 1998.
[2] Atul maheshwari, Israel koren and Wayne Burleson, “ Techniques for transient fault sensitivity analysis and reduction in VLSI circuits”.
[3] E.Stott, P. Sedcole and P. Cheung “ Fault tolerance and reliability in FPGAs” IET Comput.Digit.Tech., 2010, Vol.4, Iss.3 PP.196-210.
[4] Abderrahim Doumar and Hideo Ito,“Detecting, Diagnosing, and Tolerating Faults in SRAM-Based Field Programmable Gate Arrays: A Survey” in IEEE Trans. VLSI Syst., vol. 11, no. 3, June 2003 implementation,” in Proc. Int. Test Conf., 1997, pp. 471–478.
[5] NaveedImranand Ronald F. DeMara1 “ Cyclic NMR-based Fault Tolerance with Bitstream Scrubbing via Reed-Solomon Codes “ Respace/MAPLD conference, Albuquerque, NM, August 2011.
[6] JASON A. CHEATHAM and JOHN M. EMMERT “A Survey of Fault Tolerant Methodologies for FPGAs” ACM Transactions on Design Automation of Electronic Systems, Vol. 11, No. 2, April 2006.
[7] KELLY J., IVEY P.: ‘A novel approach to defect tolerant design for SRAM based FPGAs’. Int Workshop on FPGAs,1994
[8] HATORI F., SAKURAI T., NOGAMI K., ET AL.: ‘Introducing redundancy in field programmable gate arrays’. Custom Integrated Circuits Conf., May 1993, pp. 7.1.1–7.1.4
[9] DURAND S.: ‘FPGA with self-repair capabilities’. Int. Workshop on Field Programmable Gate Arrays, 1994,pp. 1–6
[10] KELLY J., IVEY P.: ‘Defect tolerant SRAM based FPGAs’. Int. Conf. on Computer Design, 1994, pp. 479–482
[11] HOWARD N.J., TYRRELL A.M., ALLINSON N.M.: ‘The yield enhancement of field-programmable gate arrays’, IEEE Trans. VLSI Syst., 1994, 2, (1), pp. 115–123
[12] W.-J. Huang and E. J. McCluskey, "Column-Based Precompiled Configuration Techniques for FPGA," in The 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01), 2001, pp. 137-146.
[13] LACH J., MANGIONE-SMITH W.H., POTKONJAK M.: ‘Enhanced FPGA reliability through efficient run-time fault reconfiguration’, Trans. Reliab, 2000, 49, (3), pp. 296–304
[14].J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Low overhead fault-tolerant FPGA systems," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 6, pp. 212-221, 1998
[15] LACH J., MANGIONE-SMITH W.H., POTKONJAK M.: ‘Algorithms for efficient runtime fault recovery on diverse FPGA architectures’. Int. Symp. on Defect and Fault Tolerance in VLSI Systems, 1999

[16]S. Dutt, V. Shanmugavel, and S. Trimberger, "Efficient incremental rerouting for fault reconfiguration in field programmable gate arrays," International Conference on Computer Aided Design: Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design, vol. 7, pp. 173-177, 1999.

[17] ABRAMOVICI M., EMMERT J.M., STROUD C.E.: 'Roving STARs: an integrated approach to on-line testing, diagnosis, and fault tolerance for FPGAs'. NASA/DoD Workshop on Evolvable Hardware, 2001, p. 73

[18]V. Lakamraju and R. Tessier, "Tolerating operational faults in cluster-based FPGAs," Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays, pp. 187-194, 2000

[19]. NARASIMHAN J., NAKAJIMA K., RIM C.S., DAHBURA A.T.: 'Yield enhancement of programmable ASIC arrays by reconfiguration of circuit placements', IEEE Trans. CAD Integ. Circuit Syst., 1994, 13, (8), pp. 976–986

[20] EMMERT J.M., BHATIA D.K.: 'A fault tolerant technique for FPGAs', J. Electron. Test., 2000, 16, (6), pp. 591–606

[21] J. M. Emmert, C. E. Stroud, and M. Abramovici, "Online Fault Tolerance for FPGA Logic Blocks," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 15, pp. 216-226, 2007

[22] HANCHEK F., DUTT S.: 'Node-covering based defect and fault tolerance methods for increased yield in FPGAs'. Int. Conf. on VLSI Design, January 1996, pp. 225–229

[23]C. Stroud and S. Garimella, "Built-In Self-Test AND Diagnosis OF Multiple Embedded Cores IN So Cs," Proceedings of The 2005 International Conference on Embedded Systems and Applications, pp. 130-136.

[24]A. Sarvi and J. Fan, "Automated BIST-based diagnostic solution for SOPC," Design and Test of Integrated Systems in Nanoscale Technology, 2006. DTIS 2006. International Conference on, pp. 263-267, 2006.

[25]. Shantanu Dutt, Vinay Verma and Vishal Suthar "Built-in -Self-Test of FPGAs with Provable Diagnosabilities and High Diagnostic Coverage with Application to On-Line Testing" in proceedings of 2004 Design Automation conference, 2004.

[26]. Manuel G. Gericota, Gustavo R. Alves, Miguel L. Silva, and José M. Ferreira "Reliability and Availability in Reconfigurable Computing: A Basis for a Common Solution" Very Large Scale Integration (VLSI) Systems, IEEE Transactions on vol .16, no 11, Nov 2008.

[27].Bolchini, C., Miele, A., and Santambrogio, M. 2007. TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs. 2007. International Symposium on Defect and Fault-Tolerance in VLSI Systems (Rome, Italy, Sep. 2007).

[28]. Martin Straka, Jan Kastil "Fault tolerant structure for SRAM based FPGA via PDR. 2010. 13th Euromicro Conference on Digital system design: architectures ,methods and tools. Pp 365-372.

[29]. Borecky J., Kohlik M., Kubatova H., Kubalik P., "Fault Coverage Improvement based on Fault Simulation and Partial Duplication"13th Euromicro conference on DSD, 1-3 Sept. 2010, PP: 380 - 386

[30]. S. Mitra, W.-J. Huang, N. R. Saxena, S.-Y. Yu, E.J. McCluskey, "Recongurable Architecture for Autonomous Self Repair" IEEE Design & Test of Computers, May-June 2004, Vol 21, PP:228-240

[31].Jacobs, A., George, A.D., Cieslewski, G.,"Recongurable fault tolerance: A frame-work for environmentally adaptive fault mitigation in space" IEEE con on FPLA- Sept 2009, pp 199-204.

[32]. C. Bolchini, D. Quarta, and M. D. Santambrogio, "Seu mitigation for sram-based fpgas through dynamic partial reconfiguration," in GLSVLSI '07: Proceedings of the 17th ACM Great Lakes symposium on VLSI. New York, NY, USA: ACM, 2007, pp. 55–60.

[33]. M. G. Gericota, L. F. Lemos, G. R. Alves, and J. M. Ferreira, "Online self-healing of circuits implemented on reconfigurable fpgas," in IOLTS '07: Proceedings of the 13th IEEE International On-Line Testing Symposium. Washington, DC, USA: IEEE Computer Society, 2007, pp.217–222.

[34]. 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.vol. 25, no. 6, pp. 30–39, 2005