A Novel Approach for Providing Fault Tolerance to FPGA-Based Reconfigurable Systems

Upasana Sharma, *Member, IACSIT* and Shampa Chakraverty

***Abstract*—The dynamically reconfigurable Field Programmable Gate Arrays (FPGAs) are most frequently employed for developing adaptive embedded systems. They are also being increasingly used as co-processors in high performance computing applications. For these systems to be fielded in harsh environments such as those encountered in space, extra- terrestrial locations and regions of extreme conditions on the earth, one must adopt fault tolerant design techniques to ensure uninterrupted and reliable operation despite the occurrence of faults. Commercial Off-the-Shelf (COTS) FPGA components offer a cost effective design trajectory where the designer can choose among a rich variety of FT approaches and techniques. This paper compares the various FT techniques and proposes a novel method in which these techniques can work together to provide a synergetic approach for fault tolerant FPGA design.**

***Index Terms*—FPGA, fault tolerance techniques, dynamic partial reconfiguration.**

I. Introduction

Reconfigurable Computing (RC) is an upcoming field that bridges the gap between hardware and software. The principal benefits of RC are the ability to execute larger hardware designs with fewer gates and to realize the flexibility of a software-based solution while retaining the execution speed of a more traditional, hardware-based approach.

Field Programmable Gate Arrays (FPGAs) are a general class of RC hardware which contain an array of programmable logic blocks (PLBs), with programmable interconnect between PLBs, as well as programmable I/O cells. The configuration bits used to program the FPGA determine the function of the device.

Advances in digital technology provide the means to make each generation of FPGAs significantly more attractive and useful than their predecessors. Each generation introduces additional benefits and utilities besides the expected larger size and faster processing. One of the advancements of significant importance is the ability to reconfigure a portion of an FPGA, leaving the others unchanged. This feature is called Partial Reconfiguration (PR). It is much faster than the reconfiguration of the entire FPGA board, especially when only a small part of the FPGA logic needs to be changed. Dynamic PR is a step ahead as it allows PR to be executed at runtime. It therefore can potentially reduce the number of devices or the device size, thereby reducing both size and power consumption. As an example, a system that requires either transmit or receive capabilities at any given time, but not both, can switch between the two modes in a fraction of a second using “partial reconfiguration”.

FPGAs are excellent candidates for applications that require hardware software co-design. They can also serve as hardware acceleration coprocessors for building high performance computing applications. Practical experience with FPGA-based coprocessors shows at least a ten-fold improvement in the execution speed of algorithms as compared to processors alone [1]. FPGAs especially find applications in any area or algorithm that can make use of the massive parallelism offered by their architecture. Some examples of the applications areas of FPGAs are code breaking (in particular brute-force attack, of cryptographic algorithms), digital signal processing, ASIC prototyping, image processing, encryption/decryption engines, space missions, speech recognition, defense, avionics, industrial control, automotive, medicine and a growing range of many other areas.

In all these applications, and, specifically applications deployed in mission critical environments, there is a strong need for availability and reliability. This is achieved through the provision of Fault Tolerance (FT) to such systems. As a result a lot of academic research has been done in the recent past in this area and various methods/techniques of providing FT to reconfigurable platforms have been developed. This paper emphasizes on the need of a flexible FT approach by comparing the important proposals/techniques that have been devised for the provision of FT on RC platforms. It proposes a new technique that incorporates hierarchical levels of online FT for applications running on reconfigurable platforms.

The paper is organized as follows – Section II talks about the need of FT for FPGAs. Section III discusses some key FPGA features and tools that are used in providing FT. Section IV discusses the related work. Section V provides a comparative analysis of the FT techniques. Section VII describes the proposed FT technique. Finally, we conclude in Section VIII.

II. Need for FT for FPGAs

Fault tolerance is defined as the ability of a system to operate normally given the presence of malfunctioning resources, or faults [2]. A fault tolerant system, therefore,

Manuscript received June 22, 2012; revised July 28, 2012. This work was funded by the Department of Science and Technology, Government of India under the Women Scientist Scheme.

Upasana Sharma was with the Division of Computer Engineering, Netaji Subhas Institute of Technology, New Delhi, India -110078. She is now with Safenet Inc., Noida, (U.P.), India - 201301 (e-mail: upasanash@yahoo.com).

Shampa Chakraverty is with the Division of Computer Engineering, Netaji Subhas Institute of Technology, New Delhi, India - 110078 (e-mail: apmahs@gmail.com).

consists of two parts: fault detection and fault repair (if possible) or fault bypass.

The provision of FT gains significant importance for mission critical applications like space, automotive, avionics and medicine. Since FPGAs are being used in such mission critical systems, there is a strong need for the provision of fault tolerance (FT) to the reconfigurable platforms as well. The provision of FT for an FPGA-based system is typical because it involves reconfiguration of the FPGA. This might hamper the performance of the FPGA-based system. Hence FT does not just entail overcoming the faults, but also overcoming the faults with minimum impact to performance and delays.

There can be various kinds of faults that can occur in an FPGA-based system:

- Aging faults: These types of faults are caused due to the degradation of the components, which can be attributed to a number of mechanisms [3].
- Manufacturing defects: These can be exhibited as circuit nodes which are stuck-at 0 or 1 or switch too slowly to meet the timing specification. Defects also affect the interconnect network and can cause short or open circuits and stuck open or closed pass transistors [4].
- Single Event Upsets (SEUs) and Single Event Transients (SETs): These faults occur when an energetic particle (typically a proton, neutron or a heavy ion) collides with atoms in the silicon lattice and leaves electric charge in its wake. SEUs can cause state bits to change and logic outputs to evaluate incorrectly.
- Software Faults: FPGAs are being used in applications that require hardware software co-design. In such applications, software faults are always a possibility that must be catered.

III. FPGA Features Used for FT

The recent technological development in FPGAs has led to the availability of some very useful FPGA features and tools. These features/tools are often coupled along with the traditional FT techniques discussed in section IV, to develop a robust fault-tolerant FPGA-based system. The features/tools are:

- Placement and Routing Tools: These tools physically place and map the application design to the physical resources of the FPGA [5], [6].
- Readback: Configuration is the process of loading a design bit stream into the FPGA internal configuration. Readback is the process of reading that data [7].
- Scrubbing: It involves periodical reloading of the contents of the configuration memory [8].
- Dynamic Partial Reconfiguration: It is the ability to reconfigure only a specific portion of an FPGA at runtime [9].

IV. Related Work

Most of the earlier works focused on tolerating the manufacturing defects and aging faults and manufacturing defects at either device level or configuration level using offline FT methods [2], [10]-[15]. However, provision of online FT is required to cater SEUs and software faults. With the advent of the dynamic partial reconfiguration capabilities of the new generation of FPGAs, the recent focus is on the provision of online FT, which is certainly the need of the hour.

Ambramovici et al [16]-[18] have demonstrated how self testing of small portions of a circuit can be carried out simultaneously with normal functioning of other parts. In [17], [18], a Self Testing Area, so called STAR, is first offloaded by reconfiguring its functionality on another area so that normal functioning is not affected at all, thus enabling online self testing using BIST (Built In Self Testing).

The TMR technique has been exploited for systems requiring high reliability, safety-critical applications, such as space missions [19]-[22]. The Xilinx Triple Modular Architecture [23], [24] replicates three identical copies of the same circuit and generates their voted outputs.

A volume of work has been done for providing FT to space applications built using SRAM-based FPGAs [19]- [22], [25]. The case of space applications is typical because here the FPGAs are exposed to high radiation environments. Radiation can cause both short term and permanent device failures. In short term they can cause transient upsets in circuits (SEUs and SETs). Additionally, radiation can cause permanent damage to silicon devices over time, rendering all or part of the device unusable. Authors of [21] have concluded that the TMR technique is not able to mask all the faults induced by SEUs. TMR is often coupled with scrubbing [8] to avoid fault accumulation. The authors of [6] have proposed coupling of reliability oriented place and route algorithms in order to make complete SEU immune circuits.

The concept of partial TMR has been introduced in [19]. Using device level TMR and a separate radiation hardened voter ASIC with integrated configuration management logic is another FT technique [20]. D. Fay et al [22] talk about device level TMR accompanied with adaptive FT for the distributed memory system. The authors of [25] have proposed and demonstrated a framework for Reconfigurable Fault Tolerance (RFT) that enables FPGA-based systems to change at run-time the amount of fault tolerance being used.

V. COMPARISION of FT Techniques Used for FPGAS

The design exploration process of an application must be able to weigh the pros and cons of different FT methodologies on factors such as system performance, cost, power etc. Table I provides a comprehensive summary of the comparative analysis between the various FT techniques that can be devised for an FPGA based system.

VI. The Proposal

The comparison amongst the various FT techniques for RC platforms in Section 5 clearly indicates that choosing a single FT technique for a given application shall require a tradeoff between the robustness of FT and factors like cost, system performance and power. The TMR technique is one of the most robust online FT techniques; however, it incurs significant penalties in terms of area utilization, so much so, that, for certain designs it's not even feasible to use this technique.

The partial dynamic reconfiguration feature of the latest generation of FPGAs has opened gates for a very effective and an entirely new methodology for the provision of FT for FPGA-based systems. This methodology allows the choice of any of the available FT techniques as per the application and environment requirements. Two very recent works have been based on this approach [25], [27]. In the case of [25], an application is run in different modes, with each mode having a different FT level. In the case of [27], the demonstrator application autonomously improves its FT features based on the current workload.

TABLE I: COMPARISON OF VARIOUS FAULT-TOLERANT TECHNIQUES FOR FPGA BASED SYSTEM DESIGN.

FT Parameter	**Built-in Self Test(BIST) [16][17][18]**	**Duplicate & Compare(D&C) [25][27]**	**Triple Modular Redundancy (TMR) [19][25][27]**	**Concurrent Error Correcting Codes(CEC) [25][26]**
Mode of checking: Offline or Online	Generally, testing is carried out in offline mode. [17][18] overlaps testing of portions of FPGA with online functioning of remaining portions	Online provision for error security.	Online error correction by majority voting.	Online error detection / correction with information redundancy.
Ability to handle transient faults	No. In [17][18], time to detect fault is upper bounded by time to reconfigure and test full FPGA area.	Yes: Can detect transient faults	Yes: Can correct transient faults.	Yes: Can detect /correct transient faults.
Space overhead due to A) Added FT functions	BIST circuitry.	Duplication circuitry and Comparator	Triplication, And Voter circuitry	Circuitry for error code generation, checker and code converters.
Space overhead due to B) Communications requirement	Routing test clock, test data input and output through scan chain path.	Routing input signals to both modules and error signal to other parts of system	Routing input signals to triplicate modules and their outputs to voter.	Routing error signal to dependant circuits and diagnosis/ recovery system
Memory space requirement	Stores signatures of the circuit as in [17] [18]	None*.	None*.	*For data storage integrity, CRC codes are stored with data.
		*To diagnose permanent fault an *n*-bit history needs to be recorded		
Time overhead	Time for offline testing. In [17] [18], time incurred for reconfiguring and roving the STARS	*Time for comparison and propagating error signal	*Time for voting on multiple results.	*Time to generate coded values, detect non-code values and rectify errors.
		*In all cases, pipelining techniques increase throughput at the cost of initial latency and extra space required by latches.		
Diagnostic capability	Can diagnose all stuck-at logic faults at test clock speed. Provides very high fault coverage.	Can detect faulty pair.	Identifies minority unit as faulty.	Can identify faulty module at run time and additionally locate faults.
Recovery mechanism	Bypass faulty blocks by reconfiguring their functionality.	*Avoid unsafe operation by blocking output of erroneous modules.	*Use majority output as correct. Can switch to *error secure* mode when a faulty module is removed.	*Avoid unsafe operation by blocking output of erroneous modules. For error-correcting circuits, rectify erroneous data.
		*By monitoring past results, permanently faulty module can be identified and isolated. Its functionality can then be reconfigured on spare logic cells and switched in.		
Comments	Gives best diagnosis. But continuous self testing incurs time overhead and extra power dissipation	Eliminates unsafe operation. However, it is not possible to identify which of the pair is the faulty.	Highest hardware redundancy incurs high cost but gives the best reliability	Hardware redundancy is usually less than duplicated. Using extra redundancy in coding data, errors can also be located and corrected.

Our proposal is also based on this novel methodology and intends to provide hierarchical levels of FT. Unlike [25] and [27], we intend to attach an FT level with each task comprising of the application that is being run on the FPGA. Thus, our approach combines scheduling along with the provision of FT. FPGA Scheduling is itself a very typical and wide topic that attracts the interest of a lot of researchers and various techniques are being devised and proposed for the same [28]-[32].

The three major functional components of our proposed design along with their roles are presented below:

- Task Analyzer

This module analyzes the task graph of the given application that is to be run on the FPGA. Depending on the criticality of a task, it assigns a level of FT to it. As an example, the most critical tasks are provided the highest level of FT using the TMR technique; the tasks with a defined second level of criticality are duplicated and the tasks that are not critical at all are not provided any FT.

- Scheduler

This is an enhanced version of the scheduler discussed in [32]. When a task is being scheduled, the FT level assigned to that task also plays an important role along with the other factors. For example, if a task has been assigned the highest level of FT, the scheduler will have to schedule three copies of that task.

- FT Manager

This module is responsible for the provision of FT using the framework created by the other two modules. It has a list of the executing tasks along with their assigned FT level. Depending on the FT technique assigned to a task, it checks for the proper functioning of that task. In case of a fault, this module is responsible for taking corrective actions.

VII. Conclusions

We have presented the work being done by the various research groups for the provision of FT for reconfigurable platforms. A comparative analysis of the various FT techniques has also been done. We have proposed a novel FT technique that shall integrate scheduling to provide an optimum, flexible and hierarchical FT to an application. The implementation part of the proposed approach is in progress and our future work shall focus on deriving significant experimental results out of the same.

Acknowledgement

The authors would like to thank the Department of Science and Technology, Govt. of India, for funding this work under the *Women Scientist Scheme*.

References

[1] Accelerating high-performance computing with FPGAs. [Online]. Available: http://www.altera.com/literature/wp/wp-01029.pdf.

[2] J. A. Chetham, J. M. Emmert, and S. Baumgart, "A survey of fault tolerant methodologies for FPGAs," *ACM Transactions on Design Automation of Electronic Systems*, vol. 11, no. 2, April 2006.

[3] S. Srinivasan *et al.*, "FLAW: FPGA lifetime awareness," *Design Automation Conference*, pp. 630-635, 2006.

[4] I. G. Harris *et al.*, "Testing and diagnosis of interconnect faults in cluster-based FPGA architectures," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 21, no. 11, pp. 1337-43 Nov. 2002.

[5] L. Sterpone and M. Violante, "A new reliability-oriented place and route algorithm for SRAM-based FPGAs," *IEEE Transactions on Computers*, vol. 55, Issue 6, pp. 732 – 744, June 2006.

[6] L. Sterphone and N. Battezzati, "A novel design flow for the performance optimization of fault tolerant circuits on SRAM-based FPGAs," *NASA/ESA conference on Adaptive Hardware Systems, IEEE* 2008.

[7] Xilinx Application Notes XAPP015, "Using the XC4000 readback capability".

[8] J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb, "FPGA partial reconfiguration via configuration scrubbing," *International Conference on FPL*, 2009.

[9] Correcting single event upset through Virtex partial reconfiguration, *Xilinx Application Notes XAPP216*, 2000.

[10] H. Ito Doumar, "Detecting, diagnosing, and tolerating faults in SRAM-based Field Programmable Gate Arrays; a survey," *IEEE Transactions on VLSI Systems*, vol. 11, no. 3, June 2003.

[11] N. Goel and K. Paul, "Hardware controlled and software independent fault tolerant FPGA architecture," *15th International Conference on Advanced Computing and Communications, IEEE* 2007.

[12] N. Campregher, P. Y. K. Cheung, G. A. Constantinides, and M. Vasilko, "Yield modeling and yield enhancement for FPGAs using fault tolerant schemes," *International Conference on Field Programmable Logic and Applications*, 2005.

[13] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Efficiently supporting fault-tolerance in FPGAs," *International Symposium on Field Programmable Gate Arrays*, pp. 105-15, 1998.

[14] F. Hanchek and S. Dutt, "Methodologies for tolerating cell and interconnect faults in FPGAs," *IEEE Transactions on Computers*, vol. 47, no. 1, pp. 15 – 33, 1998.

[15] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Low overhead fault-tolerant FPGA systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 2, pp. 212 – 221, 1998.

[16] J. Emmert, C. Stroud, B. Skaggs, and M. Abramovici, "Dynamic fault tolerance in FPGAs via partial reconfiguration," *Field-Programmable Custom Computing Machines, 2000 IEEE Symposium* on 17-19 April 2000, pp.165 – 174.

[17] M. Ambramovici, C. Stroud, C. Hamilton, S. Wijesuriya, and V. Verma, "Using roving STARs for on-line testing and diagnosis of FPGAs in fault-tolerant applications," in *Proc. International Test Conference*, 1999.

[18] J. Emmert, C. Stroud, and M. Ambramovici, "Online fault-tolerance for FPGA logic blocks," *IEEE Transactions on VLSI*, 2007.

[19] Pratt, M. Caffery, P. Graham, K. Morgan, and M. Wirthlin, "Improving FPGA design robustness with partial TMR," *MAPLD* 2005.

[20] G. L. Smith and L. de la Torre, "Techniques to enable FPGA based reconfigurable fault tolerant space computing," *IEEEAC*, vol. 5, pp. 1592, 2006.

[21] P. Bernardi, M. S. Reorda, L. Sterpone, and M. Violante, "On the evaluation of SEU sensitiveness in SRAM-based FPGAs," in *Proceedings of the 10th International On-Line Testing Symposium* 2004.

[22] D. Fay, A. Shye, S. Bhattacharya, D. A. Connors, and S. Wichmann, "An adaptive fault-tolerant memory system for FPGA-based Architectures in the space environment," *Second NASA/ESA Conference on Adaptive Hardware Systems* 2007.

[23] *Xilinx Inc., XTMR Tool User Guide*, UG156, August 2006.

[24] Carmichael, "Triple module redundancy design techniques for Virtex FPGAs," Xilinx Corporation, November 2001, pp. 197.

[25] A. Jacobs, A. D. George, and G. Cieslewski, "Reconfigurable fault Tolerance: A framework for environmentally adaptive fault Mitigation in space," *IEEE* 2009.

[26] W.-J. Huang, S. Mitra, and E. J. McCluskey, "Fast run-time fault location in dependable FPGA-based applications," in *Proceedings of the 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'01).*

[27] J. Soto Vargas, J. M. Moreno, J. Madrenas, and J. Cabestany, "Implementation of a dynamic fault-tolerance scaling technique on a self-adaptive hardware architecture," *International Conference on Reconfigurable Computing and FPGAs*, 2009.

[28] A. Ahmadinia, C. Bobda, S. P. Fekete, J. Teich, and J. C. van der Veen, "Optimal free-space management and routing-conscious dynamic

placement for reconfigurable devices," *IEEE Transactions on Computers*, vol. 56, no. 5, May 2007.

[29] R. Cordone, F. Redaelli, M. A. Redaelli, M. D. Santambrogio, and D. Sciuto, "Partitioning and scheduling of task graphs on partially dynamically reconfigurable FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 5, May 2009.

[30] C. Steiger, H. Walder, and M. Platzner, "Operating systems for reconfigurable embedded platforms: online scheduling of real-time tasks," *IEEE Transactions on Computers*, vol. 53, no. 11, November 2004.

[31] Y. Lu, T. Marconi, K. Bertels, and G. Gaydadjiev, "A communication aware online task scheduling algorithm for FPGA-based partially reconfigurable systems," *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, 2010.

[32] J. A. Clemente, C. González, J. Resano, and D. Mozos, "A hardware task-graph scheduler for reconfigurable multi-tasking systems," *International Conference on Reconfigurable Computing and FPGAs*, 2008.

Upasana Sharma received the M.Sc degree in Computer Science from the G.B. Pant University of Agriculture and Technology, Uttarakhand, India in 2000. She has worked on a research project on the devising of fault-tolerance techniques for reconfigurable platforms at Netaji Subhas Institute of Technlogy, New Delhi, India. She also carries a rich experience in software development and is currently a senior developer at Safenet (http://www.safenet-inc.com/) at its Noida (India) office.

Shampa Chakraverty is professor in the Computer Engineering Division at Netaji Subhas Institute of Technology, New Delhi, India. She completed her Bachelor's degree in Electronics and Communication ('83) from Delhi College of Engineering, MTech in Integrated Electronics and Circuits from IIT-Delhi ('92) and PhD in Computer Engineering from Delhi University, India. Her research interests include reconfigurable computing, design exploration for multiprocessor embedded systems, fault tolerant design, Information retrieval and complex adaptive systems.