On-Line Self-Healing of Circuits Implemented on Reconfigurable FPGAs

Manuel G. Gericota, Luís F. Lemos, Gustavo R. Alves
Dep. of Electrical Eng., LABORIS/ISEP
{mgg, lfl, gca}@isep.ipp.pt

José M. Ferreira
Dep. of Electrical and Computers Engineering, FEUP
jmf@fe.up.pt

Abstract

To boost logic density and reduce per unit power consumption SRAM-based FPGAs manufacturers adopted nanometric technologies. However, this technology is highly vulnerable to radiation-induced faults, which affect values stored in memory cells, and to manufacturing imperfections.

Fault tolerant implementations, based on Triple Modular Redundancy (TMR) infrastructures, help to keep the correct operation of the circuit. However, TMR is not sufficient to guarantee the safe operation of a circuit. Other issues like module placement, the effects of Multi-Bit Upsets (MBU) or fault accumulation, have also to be addressed. In case of a fault occurrence the correct operation of the affected module must be restored and/or the current state of the circuit coherently re-established.

A solution that enables the autonomous restoration of the functional definition of the affected module, avoiding fault accumulation, re-establishing the correct circuit state in real-time, while keeping the normal operation of the circuit, is presented in this paper.

1. Introduction

The reduction in transistors size experimented by new generation of semiconductor technology lead to a greater integration and to a per unit power reduction, enabling chips to grow in size and complexity. But new nanometer scales also bring some negative aspects, such as the vulnerability to soft errors, due to Single Event Upsets (SEU) having its origin in background radiation, and to electromigration. Despite soft errors do not physically damage chips, values stored in memory cells may be modified, causing incorrect data to be transmitted or an improper instruction to be retrieved by a processor. Furthermore, after large periods of operation, defects related to small manufacturing imperfections, not detected by production tests, may become exposed, due to, for example, electromigration phenomena, emerging as permanent faults.

These problems have a particular impact on the reliability of SRAM-based Field Programmable Gate Arrays (FPGAs). They have been enduring a considerable evolution in the last few years, both in terms of density and complexity, with nanometer technology being currently used in their manufacturing. Unfortunately, the related exponential growth in the amount of memory cells, needed for configuration purposes, has turned them especially vulnerable not only to physical imperfections but also to radiation-induced faults, such as SEU and Multi-Bit Upsets (MBU) [1-4]. Although these last set of faults do not physically damage the chip, their effects are permanent, since the functionality of the circuits mapped into the device is permanently altered.

Another problem are the transients (glitches) induced in combinatorial logic paths by the incidence of heavy ions (a phenomenon known as Single Event Transients (SETs)), which may be propagated to flip-flop inputs, where, as system clock speeds increase, they have a high probability to be registered, causing soft-errors in the user data. Besides, if a SET strikes a clock line, double-clocking may occur, leading to an extemporaneous update of part or all the flip-flops driven by that line (depending on the charge value and on line attenuation).

Full module redundancy, namely Triple Modular Redundancy (TMR), has been the preferred choice to improve the reliability of highly critical applications based on FPGAs since it does not require any architectural innovation and it is function-independent [4-10]. In a discrete implementation of a TMR system, if a defect affects the functionality of a single module the system will continue to work correctly. However, a

♦ This work is supported by an FCT program under contract POSC/EEA-ESE/55680/2004.

13th IEEE International On-Line Testing Symposium (IOLTS 2007)
0-7695-2918-6/07 $25.00 © 2007 IEEE

second failure in one of the remaining modules will lead to a system failure. Ideally, when a module fails, it should be replaced to restore the initial redundancy, but this action may not be possible immediately (or may even be impossible, like in space applications). In FPGA-based systems, in the event of a module failure, the initial redundancy may be restored by reconfiguration of the affected module. No physical replacement is therefore necessary, resulting in a significant improvement in reliability without a comparable rise in costs. Occasionally, even physical defects may be overturned by reconfiguration.

A framework for implementing self-healing circuits in FPGAs, making them immune to radiation-induced faults and, within limits, also to structural defects, is herein explored. Its aim is to fully automate the procedure of confining, detecting, locating and mitigating structural and radiation-induced faults in the TMR modules, creating a self-healing mechanism fully contained in the same FPGA. The full proposal was implemented on a XC2V1500 from Xilinx.

2. Literature survey

The results of several radiation campaigns in SRAM-based FPGAs, carried out with the objective of understanding the effects of radiation-induced faults, were reported by several authors [2, 3, 7]. In general, these authors observed that radiation changes the correct functionality of the circuits, an effect defined as a Single Event Functional Interrupt (SEFI). A classification of SEFIs was proposed in [1-2].

Several fault injection approaches, proposed as alternatives to (expensive) radiation campaigns, may also be found in the literature [9-11]. The greatest advantage of these methods is the higher controllability of the experiments, in contrast to the unpredictability of radiation injection, which enables a better diagnostic of the effects of each SEU. A combination of both techniques, not only to increase the controllability of the experiments, but also to verify the accuracy of the emulation fault injection techniques used, may be found in [4, 5, 12, 13].

Lately, several hardening techniques have been proposed to avoid SEU effects on the functional behavior of circuits. Correcting techniques based on dynamic reconfiguration, known as scrubbing, like those presented on [14-16], periodically read back the configuration memory to detect bit flips caused by SEUs. If a bit flip is detected, the affected frame is reconfigured and the system is reset. However, the same authors recognized that a fault-free read-back of the configuration bitstream does not guarantee that a SEU did not occur. In fact, radiation-induced bit-flips in flip-flops occur without upsetting the bitstream. Another drawback is fault detection latency. The read-back operation of the whole configuration memory may take from several milliseconds to a few hundred milliseconds depending on the size of the FPGA and on the interface used to perform it. By then, the fault propagation may already have caused a system operation failure. Furthermore, those techniques do not cover emergent structural defects.

Alternative techniques based on hardware redundancy were proposed without the aim of identifying and correcting faults, but just to mask its existence. After extensive testing, several authors proved that SEU-induced failures can be properly controlled for the Virtex family of FPGA devices using TMR associated to a careful placement and routing [6, 7, 9, 10, 13, 17]. Fault tolerance is achieved using extra components to instantaneously mask the effect of a faulty component, meaning that no fault propagation will occur. Still, as no fault detection occurs, the faulty module is not replaced and therefore the initial redundancy (and reliability level) is not restored. Consequently, over time, cumulative faults will increase the probability of a system failure.

It is also important to take account of the results achieved during radiation campaigns concerning MBUs due to single charged particles, since they may potentially affect multiple redundant modules and produce incorrect values. These effects are intrinsically related to the architecture of the configuration memory. In earlier Virtex generations, configuration memory is divided into one bit wide vertical frames that span from the top to the bottom of the array. Each column of Configurable Logic Blocks (CLBs) comprises multiple frames, which combine internal CLB configuration and state information, with column routing and interconnection information. In [5] it is reported that MBUs in Virtex devices occurred all in the same configuration frame, while in the Virtex-II family, the percentage of MBUs that occurred in the same configuration frame decreases to 88%. However, no MBUs struck configuration frames from two different CLB columns [4]. Configuration memory organization changed in more recent generations of Virtex devices. Instead of spanning the array from top to bottom frames are now restricted to a fixed number of CLB rows, defining a grid of configuration regions.

In summary, although the association between dynamic reconfiguration and TMR seems to be the most effective way to mitigate the effects of radiation, extra care is required during the mapping of the circuits into the FPGA and a particular attention is required concerning the coherent re-establishment of the module state after reconfiguration or after the occurrence of bit-flips in flip-flops. Furthermore, none of these techniques is sufficient to re-establish the correct operation of a module if the FPGA is itself affected by a structural defect.

The experimental results and conclusions reviewed above were taken into account when developing our proposed framework for the design and implementation of self-healing FPGA-based circuits.

3. The proposed framework

In a classic TMR implementation [18], the correct circuit output values are settled by voting elements that accept the outputs from three redundant sources and deliver the majority vote at their outputs. To ensure a consistent reliability index, voters have also to be replicated, in a schema known as T-TMR [18]. Only single faults are usually covered by T-TMR, but multiple faults may also be masked, providing that i) they affect only one of the redundant modules or voters, ii) if affecting different modules, they involve different signals and bitwise comparison is used. In these cases, faults are confined to the module or voter where they emerged, and are not visible from its outside.

To fully prevent functional problems caused by configuration upsets, each signal should enter the FPGA in triplicate, using three input pins [17]. Otherwise, a failure at the single input would cause the error to propagate through all the redundant modules, and thus it would not be masked. This same principle applies to clock signals. Each of the triplicate circuit modules should receive its own clock to avoid that spurious signals induced by SETs on a single clock line lead to an extemporaneous update of all the three-module registers.

Output signals should also leave the FPGA in triplicate, with minority voters monitoring each output [17], converging to a same node outside. When one output is different from the others, the correspondent pin is driven to high impedance avoiding contention.

Some of these implementation aspects were already addressed by a Xilinx tool called TMRTool [19]. However, the simple implementation of TMR is not sufficient to guarantee complete immunity to radiation effects or to emerging structural defects. Other issues, like the effects of MBUs or fault accumulation, have also to be addressed to guarantee the correct long term operation of the circuits implemented in the FPGA. Placement and routing considerations presented in [10] were also taken into account when developing the present framework, in conjunction with the results produced by the fault injection campaigns reported in [2-4, 7].

Our proposed framework divides the FPGA vertically into four areas: three for the user's circuit modules and a fourth area for placing a detection-and-fix controller.

The interconnections between a module and its own Input/Output Blocks (IOBs) should not cross other modules'area minimizing route networking share. The overall implementation schema proposed is illustrated in figure 1.

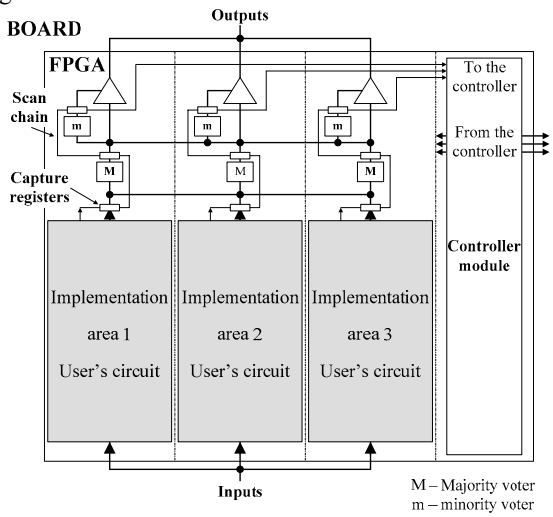

Figure 1. Proposed framework overview

When one or more faults appear in one of the modules or voters, the T-TMR implementation confines the fault and masks its existence, avoiding its propagation to the rest of the circuit. However, the cumulative effects of two or more faults induced over time may suppress the effectiveness of the confinement and masking mechanism, allowing fault propagation. With the aim of detecting data incoherencies, locating the faulty module and restoring its correct operation a detection-and-fix controller was implemented in the fourth area defined on the FPGA logic space. This repair procedure is done transparently, through partial reconfiguration of the affected module, without human intervention, since physical component replacement is not needed. As a result, a higher level of maintainability is achieved without implying the inoperability of the circuit. An overview of the controller structure is shown in figure 2.

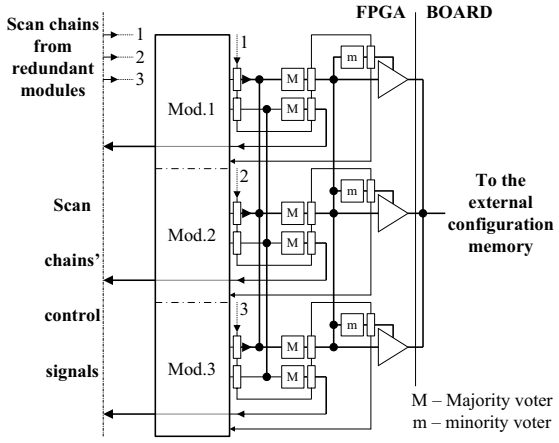

Figure 2. Structure of the detection-and-fix controller

4. Fault detection, location and mitigation

It is very hard to detect the emergence of a fault in a T-TMR implementation using traditional online test strategies, since the redundancy of the circuit masks its effect. Therefore, fault detection has to occur before its masking. In our approach, this is done via three scan chains that regularly capture the values at the outputs of modules and voters.

A Boundary-Scan (BS)-like register [20] is used to implement the scan chains, composed of simpler cells comprising only a capture / shift stage. The absence of the latch stage means that no delay is introduced in the signal's path by the scan chain. To avoid capturing undefined values, the scan chain is updated synchronously with the system clock (assuring that modules or voters outputs will be in a steady state when they are captured).

The scan chain control signals are generated by the detection-and-fix controller. This controller regularly updates the scan chains and shifts their contents, comparing the output values. Three parallel scan chains are used, each covering a different user's module, with comparisons executed "on-the-fly". Thus, there is no need to hold the shifted values. This approach makes it easier for the controller to accurately diagnose which of the three module areas is affected by a fault, and to trigger its reconfiguration. Since the shifting time is divided by the number of parallel chains more frequent capture operations are enabled, contributing to decrease fault detection latency.

The problem with this approach is that if a fault affects the content of a flip-flop, the output of its module will exhibit a wrong value, which will be captured by the scan chain, triggering the reconfiguration procedure of the module. However, reconfiguration will not correct the value stored in the flip-flop, and so, the error will persist. A feedback schema to correct faults in flip-flops, based on a majority voter that compares the outputs of the three replicated flip-flops and returns the correct value to the inputs is proposed in [17]. Furthermore, this approach results in the partition of the module in smaller logic blocks with voters in-between, which increases the robustness of the TMR in the presence of routing upsets without being of concern to floorplanning [9]. Yet, if a fault affects one of the majority voters, this voter will return to the input of its flip-flop a faulty value. This fault will be permanent and will be propagated to the inputs of the remaining voters, increasing the risk of fault accumulation and consequent circuit failure. A detection mechanism placed only at the outputs of the circuit may fail the detection of these faults if a fault occurs deep in the module logic.

To solve this problem the scan chain was extended to cover the inputs and outputs of each one of the flip-flops in the circuit. Thus, it is not only guaranteed that in case of a bit-flip in the flip-flop it will be correctly updated in the next clock cycle, but also that any functional fault affecting the majority voter will be detected through the scan chain, enabling to determine which module should be reconfigured. Furthermore, extending the scan chain to inside the module and wrapping on it the different combinational blocks and registers enables a more precise location of the fault. Considering the organization into regions of the configuration memory of the Xilinx Virtex-4 and -5 families [21], a more precise fault location enables the controller to activate the partial reconfiguration of smaller areas of the FPGA.

The original partial configuration files of the four defined areas are stored externally. Due to the volatility of the FPGA configuration memory, an external memory is already necessary to hold the FPGA configuration bitstream (to be uploaded during system power up).

If one of the modules is affected by an emerging structural fault the detection and location procedure works exactly the same way. However, reconfiguration of the affected area using the same partial configuration file won't solve the problem. Ideally, and after locating the fault, the system should be able to recreate the placement process avoiding the faulty resource. However, the amount of processing and the indispensable resources needed to implement such procedure are not practical.

One way of trying to overcome this problem is to use design diversity [22], where each module is synthesized using different synthesis techniques (which leads to different implementations of the same logic circuit), associated with placement diversity inside the module area. Instead of only one partial configuration file, several configurations, implementing exactly the same functionality but using different resources, are stored into the external memory

Firstly, when a fault is detected, the controller tries to correct it by reconfiguring the area using the original partial configuration file. If the fault persists it indicates the probable emergence of a structural fault. Therefore, the controller reconfigures the area again but using the next partial reconfiguration file available for that area. The controller also increments the reconfiguration value flag of that area, keeping trace of the current partial reconfiguration file being in use.

However, it should be noticed that, being a fairly good solution, this is not a perfect one. A structural fault located in one of the I/O blocks used by the module has a very low probability of being corrected by using an alternative placement, because the I/O placement is dependent on the pin location which is fixed. The effectiveness of this approach decreases with higher

occupancy rates, because the number of spare resources is obviously smaller, limiting placement diversity.

Obviously, the controller and the scan chains may also be affected by SEUs or structural faults. To ensure their correct operation, the controller is equally implemented using a T-TMR design and its combinational logic and voter output signals are also covered by the scan chains, creating a self-verifiable circuit. The option of concentrating the controller in only one area, despite being implemented in T-TMR, was taken to reduce complexity and the number of occupied CLB columns. However, since it occupies fewer slices than those available in each column, modules are conveniently separated.

The first bits of the scan chain belong to the outputs of the controller. If an incoherency is detected in those first bits, the controller will be fully reconfigured and reset at once. This procedure guarantees that the controller is working properly. While not being a critical component (concerning the functionality of the system), a fault-free controller is mandatory to maintain the reliability level of the whole system.

The algorithm executed by the detection-and-fix controller is the following:

```
While
    Update scan chains & shift-and-compare each set of bits;
    If an incoherence is detected on the bits belonging to the
       controller then
          Update scan chains & shift-and-compare each set of
          bits;
       If incoherence persists then {
          Reconfigure controller module;
          Update scan chains & shift-and-compare each set of
          bits }
          If incoherence persists then
             Reconfigure controller module using a diverse
             partial configuration file;
    ElseIf an incoherence is detected on the bits belonging to
          one of the user's module then
          Update scan chains & shift-and-compare each set of
          bits;
       If incoherence persists then {
          Reconfigure user's module;
          Update scan chains & shift-and-compare each set of
          bits }
          If incoherence persists then
             Reconfigure user's module using a diverse
             partial configuration file;
    ElseIf an incoherence is detected on multiple bits then
          Reconfigure all FPGA;
       If incoherence persists then
          Reconfigure all FPGA using diverse partial
          configuration files;
    End If
End While
```

Of course, if an upset affects the values shifted through the scan chain, this will falsify fault diagnosis and consequently trigger an extemporaneous partial reconfiguration of the supposedly faulty area. This operation, although unnecessary, will not affect the operation of the system. In the case of a structural fault affecting scan chains several neighboring bits will be disturbed, falsely indicating that a general failure in one or more modules occurred. Additionally, it won't be possible to locate the place where the fault or faults emerged. In this situation, the controller undertakes a full dynamic reconfiguration of the FPGA, completely restoring the structural integrity of the scan chains.

5. Case study

To evaluate the effectiveness of our approach, we developed an experimental circuit based on a 32-bit counter and on a cascade of add/subtractor blocks. The use of a cascaded configuration enabled the building of a large circuit, able to maximize FPGA occupation, with a medium level of complexity but where the addition of new add/subtractor blocks does not imply a decrease in its maximum frequency of operation, which enable to confirm the validity of the approach even with high performance circuits. This circuit was implemented in a XC2V1500-based prototyping board, according to the rules defined in our proposed framework. The detection-and-fix controller used a total of 254 slices, distributed across two of the 40 available CLB columns, representing an area overhead of 5%. This overhead is constant and independent of the size or the complexity of the circuits implemented on the FPGA. The maximum speed of operation achieved by the detection-and-fix controller alone (obtained by simulation) was 221 MHz.

The remaining 38 columns were divided in three areas of 12 columns each, leaving a total of 2304 slices available for the implementation of each user's module. Of these, 2184 slices were occupied by the experimental circuit. The extra two columns (remainder of the division of 38 by 3) were placed among the three areas of 12 columns, to reinforce protection against (improbable) column-spanning MBUs [4]. The overhead introduced by the scan chain depends on the number of internal flip-flops covered by the scan-chain and on the number of outputs of the user's circuit. Our experimental circuit had 100 cells by scan-chain, running at a frequency of 125 MHz, given a maximum fault detection latency of 800 ns. To mention an estimation, in the Cibolla flight experiment it has been anticipated that memory cells upset at a rate ranging from 0.13 SEUs per hour in a quiet sun environment to 4.2 SEUs per hour during the peak upset rate [12]. Comparatively, the complete

reconfiguration of the FPGA takes 300 ms. Therefore, shifting time is negligible when compared to reconfiguration time. Furthermore, our method enables to reduce fault latency by, at least, a third, since in most cases it will be required to reconfigure only a third of the FPGA to correct the fault. In case of fault detection, the detection-and-fix controller initiates the partial reconfiguration by resolving the location address of the partial file to be configured. Our prototyping board uses System ACE [23] to keep trace of the partial configuration files and to configure the FPGA.

Tests performed continuously over several days based on the insertion of faults through partial reconfiguration proved the effectiveness of the proposed concept. This process was automated using JBits [24]. Each fault insertion takes sixteen seconds. After several thousand fault insertions the circuit registered no system failures. It was able of autonomously recovered from the inserted faults. The occurrence of structural faults was not tested.

6. Conclusion

This paper presented a framework to support the on-line self-healing of circuits implemented on SRAM-based FPGAs. Several issues addressing the effectiveness of TMR to cope with radiation-induced faults in FPGAs were reviewed and discussed to support the option of associating T-TMR to a careful placement and routing and to dynamic reconfiguration as the most effective approach to mitigate radiation-induced faults in FPGAs. To avoid system failure due to fault accumulation a complementary detection-and-fix controller mechanism was proposed, with the aim of restoring the proper operation of the modules when a fault is detected. Doing it selectively decreases fault latency and limits power consumption comparatively to repetitive *blind* scrubbing.

A complementary procedure was used to increase module robustness to the emergence of physical defects.

A practical case-study enabled the partial quantification of the overhead of our proposed solution and the assessment of its effectiveness. Further work is being done to improve the evaluation methodology.

References

[1] L. Sterpone, M. Violante, "Analysis of the Robustness of the TMR Architecture in SRAM-Based FPGAs", *IEEE Trans. on Nuclear Science*, Vol. 52 No. 5, pp. 1545-1549, Oct. 2005.

[2] M. Ceschia et al., "Identification and Classification of Single-Event Upsets in the Configuration Memory of SRAM-Based FPGAs", *IEEE Transactions on Nuclear Science*, Vol. 50, No. 6, pp. 2088-2094, Dec. 2003.

[3] M. Bellato et al., "Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA", *Proc. Design, Automation and Test in Europe*, pp. 584-589, 2004.

[4] H. Quinn et al., "Radiation-Induced Multi-Bit Upsets in Xilinx SRAM-Based FPGAs", *Proc. Military and Aerospace Applications of Prog. Logic Devices Conf.,* 2005.

[5] M. French et al., "Radiation Mitigation and Power Optimization Design Tools for Reconfigurable Hardware in Orbit", *Proc. Earth-Sun System Technology Conference*, 2005.

[6] C. Carmichael et al., "SEU Mitigation Techniques for Virtex FPGAs in Space Applications", *Proc. Military and Aerospace Applications of Prog. Logic Devices Conf.*, 1999.

[7] E. Fuller et al., "Radiation Testing Update, SEU Mitigation, and Availability Analysis of the Virtex FPGA for Space Reconfigurable Computing", *Proc. Military and Aerospace Applications of Prog. Logic Devices Conf.*, 2000.

[8] M. G. Gericota et al., "A self-healing real-time system based on run-time self-reconfiguration", *Proc. 10th IEEE Emerging Technologies and Factory Automation Conf.*, pp. 19-22, 2005.

[9] F. Kastensmidt et al., "On the Optimal Design of Triple Modular Redundancy Logic for SRAM-Based FPGAs", *Proc. Design, Automation and Test in Europe*, pp. 1290-1295, 2005.

[10] L. Sterpone, M. Violante, "A New Reliability-Oriented Place and Route Algorithm for SRAM-Based FPGAs", *IEEE Trans. on Computers*, Vol. 55, No. 6, pp. 732-744, June 2006.

[11] M. Rebaudengo et al., "Simulation-based analysis of SEU effects on SRAM-based FPGAs", *Proc. 12th Intl. Conf. on Field-Prog. Logic and Applications*, pp. 607-615, 2002.

[12] M. Wirthlin et al., "The Reliability of FPGA Circuit Designs in the Presence of Radiation Induced Configuration Upsets", *Proc. 11th IEEE Symp. on Field-Prog. Custom Computing Machines*, pp. 133-142, 2003.

[13] G. M. Swift et al., "Dynamic testing of Xilinx Virtex-II field programmable gate array (FPGA) input/output blocks (IOBs)", *IEEE Trans. on Nuclear Science*, Vol. 51, No. 6, pp. 3469-3474, Dec. 2004.

[14] M. Gokhale et al., "Dynamic reconfiguration for management of radiation-induced faults in FPGAs", *Proc. 18th Intl. Parallel and Distributed Proc. Symp.*, pp. 145-150, 2004.

[15] M. Abramovici et al., "Using Roving STARs for On-Line Testing and Diagnosis of FPGAs in Fault-Tolerant Applications", *Proc. Intl. Test Conference*, pp. 973-982, 1999.

[16] M. G. Gericota et al., "Active Replication: Towards a Truly SRAM-based FPGA On-Line Concurrent Testing", *Proc. 8th IEEE Intl. On-Line Testing Workshop*, pp. 165-169, 2002.

[17] Triple Module Redundancy Design Techniques for Virtex FPGAs, *XAPP 197 Application Note*, Xilinx, 2001.

[18] P. K. Lala, *Self-Checking and Fault-Tolerant Digital Design*. San Francisco, CA: Morgan Kaufman Publishers, 2001.

[19] TMRTool, available at: http://www.xilinx.com/ise/optional_prod/tmrtool.htm

[20] IEEE Standard Test Access Port and Boundary Scan Architecture (IEEE Std 1149.1), IEEE Std. Board, June 2001.

[21] Virtex-5 FPGA Configuration User Guide 2006, available at: http://direct.xilinx.com/bvdocs/userguides/ug191.pdf

[22] N. R. Saxena et al., "Dependable Computing and Online Testing in Adaptive and Configurable Systems", *IEEE Design and Test of Computers*, Vol. 17, No. 1, pp. 29-41, Jan.-March 2000.

[23] System ACE MPM Solution, *Product Specif.*, Xilinx, 2003.

[24] S. A. Guccione et al., "JBits Java based interface for reconfigurable computing", *Proc. 2nd Military and Aerospace Appl. of Prog. Devices and Technologies Conf.*, 1999.