View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositório Aberto da Universidade do Porto

Robust Configurable System Design with Built-In Self-Healing

Manuel Gericota, Gustavo Alves, José Ferreira

***Abstract*—The new generations of SRAM-based FPGA (Field Programmable Gate Array) devices, built on nanometre technology, are the preferred choice for the implementation of reconfigurable computing platforms. However, their vulnerability to hard and soft errors is a major weakness to robust system design based on FPGAs.**

In this paper, a novel Built-In Self-Healing (BISH) methodology, based on modular redundancy and on self-reconfiguration, is proposed. A soft microprocessor core implemented in the FPGA is responsible for the management and execution of all the BISH procedures. Fault detection and diagnosis is followed by repairing actions, taking advantage of the self-configuration features. Meanwhile, modular redundancy assures that the system still works correctly. This approach leads to a robust system design able to assure high reliability, availability and data integrity.

***Index Terms*—Built-In Self-Healing, reconfigurable computing, robust system design, reliability**

I. Introduction

THE incorporation of self-reconfiguration capabilities in recent SRAM-based Field Programmable Gate Arrays (FPGAs), allied to the inclusion of soft microprocessor cores, enabled the development of autonomous configurable computing platforms. By mapping compute-intensive sections of an application to reconfigurable hardware, these platforms tend to exhibit a significant speedup in performance over traditional microprocessors.

These developments were made possible by the introduction of Very Large Scale Integration (VLSI) technologies, which raised substantially the reliability of electronic systems, when compared with the previous use of discrete components. Hence, the use of fault tolerance techniques was confined only to specific applications requiring high levels of security or operating on harsh environments. The reduction in the size of transistors in each new generation of semiconductor technology led to a greater integration and to a per unit power reduction, enabling chips to grow in size and complexity.

However, new nanometre scales also brought some negative aspects, namely the vulnerability to soft errors, also called single-event upsets (SEUs), which are radiation-induced transient errors caused by neutrons from cosmic rays and alpha particles from packaging material. Until now, they used to be a major concern only for space applications. But, for designs manufactured at advanced technology nodes – such as 90 nm, 65 nm, and downward – system-level soft errors became an issue also at ground level. They are now much more frequent than in previous generations [1, 2].

Soft errors do not physically damage the chip, but the values stored in memory cells may be affected, causing incorrect data to be transmitted or an improper instruction to be retrieved by a processor. This problem has a particular impact on the reliability of SRAM-based FPGAs, because the structural definition of the functions implemented relies on memory cells. The exponential growing on the amount of reconfigurable logic available at each new FPGA generation implies also a similar increase on the amount of configurable memory cells, which makes FPGAs especially vulnerable to soft errors. Additionally, the amount of embedded memory blocks available for user's applications is also increasing.

Another negative aspect due to the smaller technological scales is the increased threat of electromigration, which may result in permanent physical damages to the chip. The number of defects related to small manufacturing imperfections that are not detected by production testing has been growing as scale goes down. These defects are especially prone to electromigration phenomena, resulting, after large periods of operation, in the emergence of permanent faults.

The recent addition of new features, such as dynamic reconfiguration and self-reconfiguration, the two most advanced forms of reconfigurability, may help to cope with the problems mentioned above, in particular when dealing with critical applications that require a high reliability level.

Dynamic reconfiguration involves the reconfiguration of a fraction of the configurable resources, without disturbing the operation of those functions that are not modified. This feature extends FPGA's flexibility, enabling multiple independent functions from different applications to share the same logic resources in the spatial and temporal domain [3]. More recently, and via self-reconfiguration [4], it became possible for functions currently implemented to control the dynamic reconfiguration of other areas of the same FPGA.

The advantages of using dynamic reconfiguration in the implementation of online structural test and fault tolerance strategies were largely explored in previous works [5-8]. However, those previous approaches relied on a rotate and test methodology, whose primary aim was the structural test

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

Manuel G. Gericota and Gustavo R. Alves are with the Department of Electrical Engineering, School of Engineering - Polytechnic Institute of Porto, Rua Dr. Antonio Bernardino de Almeida, 4200-072 Porto, Portugal (e-mail: {mgg, galves}@dee.isep.ipp.pt).

José M. Ferreira is with the Department of Electrical and Computer Engineering, Faculty of Engineering of the University of Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal (e-mail: jmf@fe.up.pt).

of the FPGA. Moreover, only a small fraction of the resources were configured to be tested at a time. These solutions create a test latency that must be taken into account since it degrades the performance of the test strategy. If a defect, caused by a soft error, affects the functionality of a given function, the resulting fault will be propagated until the test function reaches the defective resource. By then, the fault may already have caused the irreversible malfunctioning of the whole system, eventually interrupting its operation. In some cases, it may be impossible to recover from this situation.

This paper presents a new methodology that aims to increase the reliability of configurable computing platforms implemented in dynamically reconfigurable FPGAs. The drawback associated to previous approaches is avoided by the introduction of fault tolerance techniques.

The next section analyses traditional hardware redundancy techniques, and is followed by the presentation of the proposed methodology. Several aspects related to its practical implementation are then discussed, and future research lines are presented in the concluding section.

II. HARDWARE REDUNDANCY

The reliability of a system is defined as the probability of that system to be functioning correctly throughout an interval of time, [t0, t], given that it was performing correctly at time t0. Traditionally, highly critical applications relied on hardware redundancy to increase their reliability.

One of the best know of such approaches is Triple Modular Redundancy (TMR), a static redundancy technique that achieves fault tolerance without actually detecting any fault. In this method, extra components are used to instantaneously mask the effect of a faulty component, meaning that no propagation of the fault will occur. The concept of TMR was originally suggested by von Newmann [9], and is illustrated in figure 1. Each module may be a complete system, such as a computer, or a less complex unit, like a microprocessor or even an adder or a gate. The voting element accepts the outputs from the three sources and delivers the majority vote at its output. This concept can be extended to any number of redundant modules to produce an N-modular redundant (NMR) system, which can tolerate up to k module failures, where k=floor[(N-1)/2].

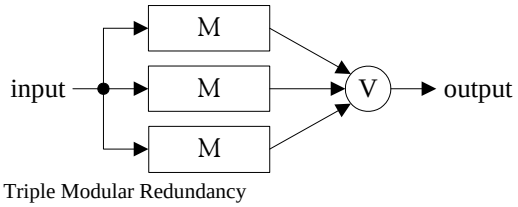

Fig. 1. Triple Modular Redundancy

The reliability equation for an NMR system is given by [10]:

$$R(t)_{NMR} = \sum_{i=0}^{n} \binom{N}{i} \cdot \left(1 - R(t)_M\right)^i \cdot R(t)_M^{(N-i)} \qquad (1)$$

In (1), R_M is the reliability of each individual module in the NMR system. In this case, it is assumed that the majority voter does not fail, which is an unrealistic principle. When this assumption is not verified, the reliability of the voter element will determine the reliability of the circuit, since it will fail if the voter fails. However, the reliability of a voter in a redundant system can be improved by replicating this element as well, in a scheme that is called N-NMR [10].

In new nanometre technology, the use of fault tolerance mechanisms is essential, not only due to soft errors, but because it is unrealistic to expect that a manufacturing test will cover all possible faults. In particular, delay faults emerging from defects of resistive type, or due to crosstalk or ground bounce, are almost impossible to foresee [11].

Hardware redundancy is also a preferred choice to improve the reliability of highly critical applications based on FPGAs [12-16]. Due to their inherent configurability, FPGAs are especially suitable for the implementation of modular redundancy, since it does not require any new architectural feature and it is function independent.

But other factors have to be considered, such as common--mode failures (CMFs). Despite the introduction of hardware redundancy, hardware redundant FPGA-based systems may fail because of the emergence of faults that affect more than one module of the redundant system at the same time, generally due to a common cause. These possible systematic defects in specific parts of the configurable logic space may eventually also lead to the simultaneous failure of more than one module of the same function (and consequently of the function itself), if all its modules are identical and implemented using equal resources [13]. Particularly, in FPGAs, soft errors affecting the configuration control mechanism of the device may cause the erroneous configuration of several modules, depending on the number of data frames affected during its partial or total reconfiguration. Another factor to be considered is multiple-event upsets caused by a source of radiation [17], which may also lead to multiple-module failure in a redundant system.

The use of design diversity, where each redundant module is synthesized using a different synthesis technique (which leads to different implementations of the same logic circuit), may help to prevent the consequences of CMFs, further enhancing reliability [16]. Additionally, the use of word-wise instead of bit-wise voters further averts the occurrence of failures due to CMFs [18].

Possible soft errors in the on-chip configuration memory cells may be recovered by simply performing a partial readback operation of the configuration of the faulty module. The retrieved bitstream is compared with its original configuration, and if a modification is detected, the correct bitstream can be re-established through partial reconfiguration. This technique is known as scrubbing, and defined as the process of re-writing the configuration memory during (and without disturbing) normal FPGA operation [19].

Readback and partial reconfiguration do not affect the data stored in flip-flop registers, and consequently soft-errors in data registers cannot be recovered using this method. However, due to the transient nature of upsets, the

error will be recovered by the circuit when the affected flip-flop is updated again. The propagation of soft-errors, either affecting data registers or the functionality of the circuits, is avoided by redundancy.

If it is clear that hardware redundancy increases the reliability of a system, it is also obvious that no single solution is able to cope with the whole universe of possible identifiable problems and their consequences. Moreover, any proposed methodology has also to take into consideration the cumulative impact of single errors, as their added effect may lead to the quick disruption of a system. The great advantage of using reconfigurability is that this issue may be solved without a significant rise in costs. In fact, in the event of a module failure, a diagnose-and-repair mechanism may be activated and the initial redundancy re--established. This may be done transparently and without human intervention, since physical component replacement is not needed. This means that a higher level of maintainability[1] is achieved, without even implying the inoperability of the affected circuit, since it is protected by TMR. This is both true to hard and soft errors, despite the different repair mechanisms that must be adopted to overcome them.

III. A METHODOLOGY TO IMPROVE THE RELIABILITY OF FPGA-BASED SYSTEMS

The reliability of a circuit implemented in an FPGA is intrinsically connected to the reliability of the component itself. Nevertheless, the maintainability of the same circuit is higher due to the dynamic and self-reconfiguration features of new FPGAs. In a discrete implementation of a TMR system, if a defect affects the functionality of one module, reliability decreases, but the system still works correctly. A second failure in one of the remaining modules may lead to a malfunctioning of the system. Ideally, when a module fails, it should be replaced to restore the original redundancy index. However, this action may not be possible immediately. In certain cases, like in space applications, it may even be impossible. Besides, it is not easy to detect a fault in a TMR implementation using traditional online test strategies, due to the inherent masking properties of redundancy.

In our approach we propose the implementation of a Built-In Self-Healing (BISH) methodology, which comprises built-in self-detection, built-in self-diagnosis and built-in self-repair. In an initial stage, this methodology is being applied only to soft-errors, but we plan to extend its usage to hard errors, making use of active replication techniques [3]. The methodology to be followed and several of the issues involved are already discussed in this paper.

The BISH approach can be divided into three tasks: detection, diagnosis and repair. These tasks are controlled by a soft microprocessor core implemented in the same FPGA, and having a compatible reliability index. Due to the usual long time interval between module failures [20], a generic soft microprocessor core that carries on other tasks related to the operation of the whole system may be used for this purpose.

The detection of faults is done through a scan chain that regularly captures the values at the outputs of all the modules and voters, including those of the soft microprocessor core, as shown in figure 2. Optionally, other information may be included. For example, if word-wise voters were used, apart from individual bit capture, the error signal produced by the word-voter can also be included in the scan chain.

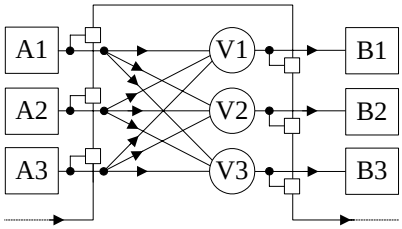

Fig. 2. Example of a T-TMR implementation with a scan chain

Upsets can also affect the values shifted through the scan chain, thus leading to wrong fault diagnosis and consequently to the extemporaneous activation of a repairing mechanism. However, despite representing an additional unnecessary task for the reconfiguration mechanism, it does not affect system operation. A more complicated situation happens if the structural configuration of the scan chain is affected by a fault, either due to a hard or soft error. In this case, several neighbouring bits in the scan chain will be disturbed, indicating that a simultaneous general failure in all modules of one or more functions is taking place. If this happens, and since the probability of a general failure is very low, the scan chain must be checked first. The Boundary Scan (BS) chain may also be used to capture each FPGA output [21]. As a hard-wired implementation, this scan chain is less prone to soft errors.

The captured bitstream is shifted to the internal microprocessor where it is analyzed. All bits at the outputs of the same set of redundant modules and at the outputs of their respective voters must be equal. If word-wise voters, or any other fault tolerance mechanisms, are used, the error signal produced has also to be checked. In the event of error detection, a diagnosis phase follows-up.

Since the scan chain cells completely wrap the modules and voters, it is possible to confine the origin of an error to the space between them, corresponding to the module or voter where the value was captured, and to the interconnections in-between [22].

Three possible causes for a fault to appear may be considered:

1. the faulty value is due to a soft error affecting one of the circuit registers;

2. the faulty value is due to a soft error affecting a configuration memory cell, which may lead to a change in the functionality of the module or voter or in the routing of signals;

3. the faulty value is due to a permanent physical defect affecting the structure of the FPGA.

The first case may be immediately excluded if the error is captured at the output of a voter, since voters are typically

[1] The probability that an inoperable system will be restored to an operational state in case of failure within the time t.

implemented using combinational logic only. If it has its origin in a module, one can expect that the fault will be automatically corrected at the next register update. A new scan chain capture operation may show that the error has already been fixed and no further action is needed. If not, the second situation may have occurred.

In this case, a background task is launched to readback part of the configuration bitstream of the area where the affected module is implemented. Comparison with the original bitstream may be done by bit comparison or Cyclic Redundancy Check (CRC). If an incoherency is found, the microprocessor performs a partial reconfiguration of the area where the supposedly affected module is implemented, restoring the original configuration and eliminating the cause of the failure. The output of the module should now be captured again and its correctness verified.

If no error on the configuration bitstream is detected after the readback-and-compare operation, but the fault persists, the most probable reason is the existence of a physical defect in the array. Therefore, in order to restore the reliability index, the affected module has to be reconfigured in a fault-free area and its input and output connections re-established. Then, the resources occupied by the faulty module are released and subsequently tested to detect and diagnosis the origin of the fault [8]. This procedure is controlled by the microprocessor. When the defect location is identified, the defective resource is “marked down”, to avoid its use in future reconfigurations. A list of faulty resources is maintained in memory by the microprocessor. This memory must also be protected against upsets using error checking and correction techniques based on Hamming or Hsiao codes [11].

The remaining resources that are tested OK can be reused in later replacements of any other faulty module. In this way, the available spare resources are almost entirely restored for future replacements. Figure 3 shows the diagram flow of the proposed methodology.

The detection, repair and test proceedings are controlled by the internal microprocessor. To not affect the performance of the system, these tasks should be executed in background during normal operation exploiting regular idle cycles. Meanwhile, the TMR implementation ensures the correct operation of the system, and therefore the extra overhead time has no critical influence over its operation.

This methodology extends the reliability of each function and enables a smoother degradation of the global reliability index. Despite being a static T-TMR implementation, a faulty module or voter is dynamically repairable n times, where n depends on the cause of the failure. If the origin is not a permanent physical defect, then n is infinite. Otherwise, n depends on the initial amount of spare resources and on the location of the defects that affect the structure of the FPGA.

The microprocessor is also implemented using T-TMR to ensure a reliability index compatible with the remaining blocks. The microprocessor is divided in small functional modules, facilitating replacement in case of fault detection, and reducing the spare space needed for replication. If the defective module is part of one of the three implemented processors, the remaining two will be responsible for the replication of the malfunctioning module. Subsequent test procedures will already be assumed by the whole three.

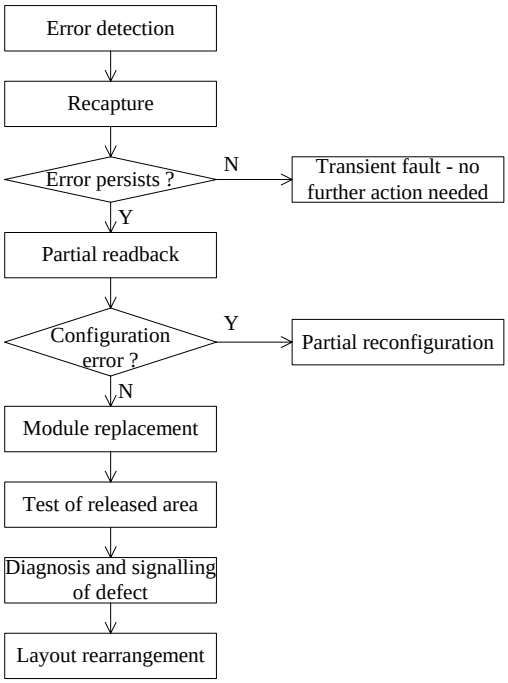

Fig. 3. Flowchart of the detection-diagnose-and-repair methodology proposed

Self-reconfiguration is necessary to embed the whole system in a single FPGA, including the BISH features. The Virtex-II and Virtex-II Pro families have an Internal Configuration Access Port (ICAP) [23]. The ICAP enables a soft microprocessor core to control its own dynamic reconfiguration or the reconfiguration of any external modules, without stopping or disturbing the operation of the whole system.

IV. Implementation Issues

Despite the apparently easy steps necessary to implement the proposed BISH methodology, many problems will have to be overcome.

When the system is synthesized, each one of the different modules from the various functions is configured in contiguous resources so as to enable lower interconnection delays and better performance. When a defect is found in an area previously occupied by a module, the remaining resources of that area may still be used in future reconfigurations. However, when a new defective module needs to be replicated, the presence of this “defective island” may disperse the components of the module, and thus lead to a degradation of performance. To avoid this effect, a rearrangement of the modules in an area with “defective islands” may be necessary [3].

Additionally, the spare area where a defective module is replicated may be relatively far from its previous location, resulting in longer path delays. Despite not leading to an erroneous output from the voter that collects the results from all the three modules of the function (if the remaining two modules still working fine), in practice it reduces module redundancy, making fault tolerance ineffective. To prevent the perpetuation of this situation, after the test the

module may return approximately to the same area where it was originally located (avoiding the defective resource). However, since the area became probably smaller, a rearrangement of the contiguous modules may be needed to fit in the returning module, without dispersing its components by different and possibly distant areas. Another possibility, depending on the defect, may be a diverse resynthesize of the module, avoiding the defective resource.

In this way, the previously occupied spare area is almost entirely released for future replications of other modules. Owing to lack of enough contiguous free resources, this procedure avoids the fragmentation of the logic space, which would eventually prevent the replication of new defective modules. That rearrangement shall be achieved without affecting the functionality or disturbing the operation of the rearranged functions, by a dynamic relocation mechanism [3]. Furthermore, to facilitate this process, only one module of each function shall be replicated at a time. This step-by-step procedure also avoids a rise in the path delays, which would lead to degradation of performance, and eventually (as referred above) to a false indication of a defective module.

The initial replication of the module and the possible subsequent rearrangement of the functions imply that the microprocessor shall be able to manipulate directly the FPGA configuration bitstream. This is necessary to create partial reconfiguration files for the replication procedures and to perform re-routing. To support this feature, a software tool is being developed, based on the JBits software – a set of Java classes that provide an Application Programming Interface (API) to access the Xilinx FPGA bitstream [24]. This tool will create the partial configuration files and will carry out the partial and dynamic reconfiguration of the FPGA through the ICAP interface. Consequently, the microprocessor shall be prepared to run this software. Two solutions are being considered: the use of a generic microprocessor; or the use of a Java processor. In the first case, a generic soft processor core will run a Java Virtual Machine (Java VM) developed specifically for that microprocessor and to support the set of Java classes used by the reconfiguration tools. The disadvantage of this solution is the amount of memory necessary to hold the JAVA VM.

The second hypothesis, the use of a Java processor, seems to be the most adequate solution for the inclusion of the BISH feature, since the software necessary to its implementation is developed using Java. These will also speed up its execution, reducing time latency between detection and correction of any fault. The disadvantage of this solution is that any other applications concerning the operation of the system, not related to the BISH feature, have to be rewritten in JAVA, which, in some cases, may not be feasible. However, this may not be a problem if a new product is being developed from scratch.

Therefore, the option between the two proposals must take into consideration not only the BISH implementation but also the purpose of the whole system and the current stage of its development cycle.

Since the partial configuration files that implement the rearrangements defined by the repair procedures are generated automatically (without designer intervention), the inclusion of this methodology is expected to be quite straightforward, and completely transparent for the final user. Its incorporation at the design level will also be automated in a later project phase.

V. Conclusions

The methodology presented in this paper is at an early stage of implementation. Therefore, apart from the description of the project and of the consolidated parts of its implementation, the proposal presents a set of issues that are being studied and must be sorted out to ensure its complete success.

Excluding the already mentioned issues and in spite of the generalized idea that TMR makes FPGAs virtually immune to hard or soft errors, further research is necessary and several issues related to their use in reconfigurable systems have yet to be considered:

- the probability of total failure due to a single-event-functional-interrupt (SEFI), caused by an upset in the device Power-On Reset (POR), which leads to the total clearing of the configuration memory and causes the loss of state data;
- the probability of a fault in a function output due to bridging faults between modules;
- the possibility of a false module or voter failure diagnosis caused by defects or upsets affecting the scan chain that captures their outputs;
- the vulnerability of the configuration control mechanism, of the ICAP and of the BS infrastructure, to defects or upsets;
- the influence that the position of replicated modules has over the effectiveness of fault tolerance features (due to a variation on the path delay between modules and voters);
- the vulnerability of the memory holding the original or current configuration file, which must also be protected against upsets using error checking and correction techniques;
- the probability of an upset to change the content of the memory block holding the microprocessor program, since TMR does not obviously offer any protection in case of software errors.

Current work is being done towards the resolution of these various issues and their integration into the proposed methodology.

References

[1] S. Mitra, N. Seifert, M. Zhang, Q. Shi, K. S. Kim, “Robust System Design with Built-In Soft-Error Resilience,” *Computer*, vol. 38, no. 2, pp. 43-52, February 2005.

[2] R. Baumann, “Soft Errors in Advanced Computer Systems,” IEEE Design & Test of Computers, vol. 22, no. 3, pp. 258 – 266, May-June 2005.

[3] M. G. Gericota, G. Alves, M. L. Silva, J. M. Ferreira, “Run-Time Management of Logic Resources on Reconfigurable Systems,” *Proc. of the Design, Automation and Test in Europe*, pp. 974-979, 2003.

[4] B. J. Blodget, S. P. McMillan, P. Lysaght, “A lightweight approach for embedded reconfiguration of FPGAs,” *Proc. of the Design, Automation and Test in Europe Designers' Forum*, pp. 399-400, 2003.

[5] M. Abramovici, C. Stroud, S. Wijesuriya, C. Hamilton, V. Verma, “On-Line Testing and Diagnosis of FPGAs with Roving STARs,” *Proc. of the 5th IEEE Intl. On-Line Testing Workshop*, pp. 2-7, 1999.

[6] M. Abramovici, C. Stroud, C. Hamilton, S. Wijesuriya, V. Verma, "Using Roving STARs for On-Line Testing and Diagnosis of FPGAs in Fault-Tolerant Applications," *Proc. of the Intl. Test Conference*, pp. 973-982, 1999.

[7] M. Abramovici, C. Stroud, B. Skaggs, J. Emmert, "Improving On-Line BIST-Based Diagnosis for Roving STARs," *Proc. 6th IEEE Intl. On-Line Testing Workshop*, 2000.

[8] M. G. Gericota, G. Alves, M. L. Silva, J. M. Ferreira, "Active Replication: Towards a Truly SRAM-based FPGA On-Line Concurrent Testing," *Proc. of the 8th IEEE Intl. On-Line Testing Workshop*, pp. 165-169, 2002.

[9] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata Studies*, vol. 34, pp. 43-98. Princeton Univ. Press, 1956.

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

[11] M. Nicolaidis, L. Anghel, "Concurrent checking for VLSI," *Microelectronics Journal*, Vol. 49, Nos. 1-2, pp. 139-156, November 1999.

[12] J. Moore, "Design Security in SRAM-based FPGAs," *MAAPLD Conf.*, 2003.

[13] C. Carmichael, "Triple Module Redundancy Design Techniques for Virtex FPGAs," *XAPP 197 Application Note*, Xilinx, Inc., 37 p., 2001.

[14] TMRTool. Information available at: http://www.xilinx.com/products/milaero/tmr/

[15] C. Carmichael, E. Fuller, P. Blain, M. Caffrey, "SEU Mitigation Techniques for Virtex FPGAs in Space Applications," *MAAPLD Conf.*, 1999.

[16] N. R. Saxena, S. Fernandez-Gomez, Wei-Je Huang, S. Mitra, Shu-Yi Yu, E. J. McCluskey, "Dependable Computing and Online Testing in Adaptive and Configurable Systems," *IEEE Design and Test of Computers*, Vol. 17, No. 1, pp. 29-41, January-March 2000.

[17] R.A. Reed et al., "Heavy Ion and Proton-Induced Single Event Multiple Upset," *IEEE Transactions on Nuclear Science*, Vol. 44, No. 6, pp. 2224-2229, December 1997.

[18] S. Mitra, E. J. McCluskey, "Word-Voter: A New Voter Design for Triple Modular Redundant Systems," *Proc. of the 18th IEEE VLSI Test Symposium*, pp. 465-470, 2000.

[19] C. Carmichael, M. Caffrey, A. Salazar, "Correcting single-event upsets through Virtex Partial Configuration", XAPP 216 Application Note, Xilinx, Inc., 12 p., 2000.

[20] P. L. Murray, "Re-Programmable FPGAs in Space Environments". Available at: http://www.seakr.com/ data/Unsorted/reprogrammable_fpga_in_space1.doc

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

[22] J. H. Lala; R. E. Harper, "Architectural principles for safety-critical real-time applications," *Proceedings of the IEEE*, Vol. 82, No. 1, pp. 25-40, January 1994.

[23] Virtex-II and Virtex-II Pro Data Sheets available at: www.xilinx.com

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