TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs

Cristiana Bolchini, Antonio Miele, Marco D. Santambrogio
Dip. Elettronica e Informazione – Politecnico di Milano
P.zza L. da Vinci, 32 – 20133 Milano – Italy
{*bolchini*, *miele*, *santambr*}@*elet.polimi.it*

Abstract

This paper presents the adoption of the Triple Modular Redundancy coupled with the Partial Dynamic Reconfiguration of Field Programmable Gate Arrays to mitigate the effects of Soft Errors in such class of device platforms. We propose an exploration of the design space with respect to several parameters (e.g., area and recovery time) in order to select the most convenient way to apply this technique to the device under consideration. The application to a case study is presented and used to exemplify the proposed approach.

1 Introduction

The adoption of Field Programmable Gate Arrays (FPGAs) as the platform for the development of embedded systems is nowadays common practice, to benefit from the limited costs, the highly flexible architecture, as well as the possibility to re-program it to modify its behavior or to correct it in case of problems. In particular, this opportunity to re-program the device has received a lot of attention and improvements have been introduced to support such a feature at run-time also, i.e., dynamically. Furthermore, advances in the FPGA technology led to support the re-configuration, at run-time, of only a portion of the device, thus being able both to reduce the time necessary to re-program the device and to avoid the necessity to halt the entire system [1, 2].

On the other hand, the flexibility of this specific platform provided by the reconfiguration capability constitutes also a problem, since the SDRAM memory elements storing the configuration bitstream (as well as the other memory elements of the device) are susceptible to radiation-induced temporary faults, also called soft errors, which corrupt the memory content causing the system to misbehave. In order to cope with this problem, particularly significant for the space environment but always more and more relevant also at ground level, fault detection and tolerance techniques need be introduced.

In this scenario several studies have been carried out to deal with the problem of radiation-induced faults in SRAM-based FPGAs from different points of view ([3, 4, 5, 6, 7, 8, 9]); some of them apply well-known techniques, traditionally adopted for other platforms too, focusing the attention on the peculiarities of the selected platform, while others exploit the opportunities of reconfiguration to mitigate faults effects. Given the available solutions, the design of a reliable system on SRAM-based FPGAs may be achieved in different ways, according to the designer's requirements and constraints.

The main contribution of this paper is the definition of an approach supporting the exploration of the solution space for the application of a passive hardware redundancy technique, using the partial dynamic reconfiguration to mitigate soft-error effects; the result is a hybrid approach, masking faults and ad-hoc recovering to cope with this class of faults. Since the technique can be applied at different levels of granularity, with different trade-offs between costs, performance and recovery time, it is important to be able to evaluate various solutions and the proposed methodology supports such an opportunity, offering the designer the chance to compare the available alternatives. While the work presented in [5] identified a limited set of parameters and techniques to achieve fault mitigation, we have generalized the approach to take into account a broader set of figures of merit.

The rest of the paper is organized as follows. Section 2 introduces the basic concepts of the adopted fault model and mitigation technique, along with a brief overview of related work in this area. Section 3 presents the proposed approach, discussing the exploration of the solution space; the application to a filter circuit used as a case study is also reported. Final considerations and future work conclude the paper.

2 Background and Motivations

In this section background and motivations of this study will be presented; at first we describe the adopted fault model, focusing on issues typical of FPGA devices. Then, the peculiarity of FPGAs supporting partial dynamic reconfiguration will be illustrated, together with a brief discussion on the adopted fault mitigation technique, i.e., Triple Modular Redundancy (TMR).

2.1 The Adopted Fault Model

Digital devices, such as ASICs or FPGAs, are commonly affected by transient faults caused by radiations and alpha particles. The occurrence of these phenomena is increased by technological scaling and noise margin reduction and causes the manifesting of their effects not only in the harsh space but also at ground level [10]. These types of faults may affect both combinatorial and sequential logic: in the first case, the effect of the fault, called Single Event Transient (SET), is a pulse on the signal that if latched in a register produces an erroneous value; in the second one, the fault, called Single Event Upset (SEU), causes a state transition in a elementary storage cell. These types of faults are commonly referred to as *soft error*, i.e., a transient fault that affects the devices causing a bit-flip of a value stored in a memory cell; the error can be recovered by re-writing the correct value.

When considering FPGA devices, the occurrence of soft errors is a more problematic issue, due to the high density of SRAM memory cells used for storing application information (constituted by data and control information for the running application) and application configuration. In the first case, faults may corrupt the content of an application register causing an error in the data or in the control of the running application, implying an error in the computation and, therefore, an erroneous result. In another case, faults may affect a configuration register, causing either the functionality performed by a Look-Up Table (LUT) or a routing cell between Combinatorial Logic Blocks (CLBs) to change, resulting in a functionality alteration. While the first situation may be recovered by a re-computation or the application reset, the second one requires a reconfiguration of the FPGA.

2.2 Partial Dynamic Reconfiguration

Complete reconfiguration is the most immediate and easiest way to reconfigure an FPGA; the bitstream, containing the FPGA configuration data, provides information regarding the complete chip and by downloading it on the board, the entire FPGA is programmed. This kind of reconfiguration introduces no particular constraints. The main disadvantage of such an integral approach is the performance overhead and all the elements necessary to support computation suspension and recovery. In order to overcome this limitation, *partial reconfiguration*, has been proposed [1]; in fact, modern FPGAs such as VirtexII, VirtexIIPro, Virtex4 and Virtex5 [2], offer the capability to be reconfigured at run-time. By exploiting this feature, a part of the configuration can be changed while the FPGA is working, avoiding reloading the whole application on the board before re-starting it.

FPGA configuration memory may be seen as a rectangular array of bits where the smallest part that can be reconfigured is a *frame*, i.e., a portion of area 1 bit wide and extended from the top of the array to the bottom. According to its position into the array, the frame provides the configuration of a specific part or component of the device, i.e., CLBs, BRAMs, IOBs. The idea of the partial reconfiguration is to define partial bitstreams, each one to update information of a specific part of array implying the modification of the specific device area. To exploit partial dynamic reconfiguration, the application must be implemented fulfilling a set of constraints, that are often device dependent. Several flows have been proposed in the past for partial dynamic reconfiguration ([11]); more recently, *Early Access Partial Reconfiguration* (EAPR) design flow based on the module-based approach has been introduced proposing new features. The first one is the possibility to allow signals (routes) in the base design to cross through a partially reconfigurable region without the use of a bus-macro. The second interesting characteristic is the possibility of defining rectangular reconfigurable regions (2D placement), since there are no constraints requiring the use of the whole height of the reprogrammable device for each reconfigurable region. This feature is very useful to implement designs, based on particular devices such as Virtex-4 [2], where reconfigurable components can be defined to maximize resource usage; in this paper, though, we will consider 1D placement only.

2.3 Triple Modular Redundancy

The TMR is the most commonly adopted passive hardware redundancy technique achieving fault masking properties [12]. It uses three replicas of the whole system and adds a voter that identifies the correct result among the three ones on the basis of a majority vote. The technique may be applied at different levels of abstraction, from the whole system to the single component offering in this second case a reduced detection latency together with an higher area ovehead with respect to the first one; moreover, due to the fact that the voter may be affected by a fault, it must be Totally Self-Checking (TSC) itself, i.e., it must be able to detect its own faults, although it still remains a single point of failure. For our voters we have referred to literature [13, 4] and are currently working on a modified version, to achieve additional localization functionality.

TMR is applied to mitigate the effects of soft errors occurring in the memory elements storing temporary data used during computation, as well as in those storing the configuration bitstream; in the first situation the masking effect of the voting system provides an autonomous recovery from the error, while in the latter situation a re-configuration of the

FPGA device is in order. In our approach, as discussed in the next section, we propose to monitor the voters' error outputs to determine which kind of memory element is affected by the soft error, thus triggering a partial dynamic reconfiguration in case the bitstream has been corrupted.

Since TMR can be applied at different granularity levels, and different area overheads, performance degradation, and reconfiguration times characterize the various solutions, we estimate these parameters and provide the designer with the most interesting solution according to his/her requirements.

2.4 Related work

The adoption of the TMR as a fault mitigation technique has been proposed in recent years, such as in [3, 4], focusing the attention on the important task of the optimal design of the TMR logic (e.g., how to place and route the voters, interconnection protection) to ensure robustness, analyzing and comparing different solutions. These analyses take into account significant aspects of TMR implementation and provide guidelines for adopting the approach; nevertheless, they do not consider the opportunity to reconfigure only a portion of the device in order to recover from the soft error. As a result, other aspects, such as data buffering between the different frames to be independently reconfigured, need be taken into account. Furthermore, in this context the exploration of the design space must also include parameters related to reconfiguration, to be able to compare different solutions.

Another recent paper adopts error detection codes to cope with SEU in LUTs [6], analyzing costs and performance with respect to hardware redundancy approaches, not exploiting though partial dynamic reconfiguration; this opportunity is used in [5], coupled with Duplication With Comparison (DWC) and thus not providing fault tolerance properties.

On the other hand, in literature there are approaches adopting partial reconfiguration to mitigate the effects of SEU faults [7, 8, 9]; such techniques use *readback* and *scrubbing* to determine if an error has occurred and eventually trigger a re-download of the correct portion of bitstream. These techniques though, work only for the bitstream corruption, being unable to determine if a SEU has occurred in the memory used for computation. Furthermore, the constant *readback* period implies a not optimal error detection latency.

This work aims at coupling the benefits of the two strategies, evaluating different designs in order to meet the designer's requirements and to limit overheads; the next section presents the mitigation technique and the supported design space exploration.

3 The Proposed Mitigation Technique

The proposed approach consists in applying the TMR on the system specification, in order to identify and mitigate the occurrence of a bit-flip; the voter takes in input the three replicated results and computes the correct data also in the case of a single module failure, also signaling which one of the replicas – or the voter itself – computes an erroneous value. This information will be used to control the re-configuration process in case the fault is identified as having affected the configuration memory, and thus exposing a "permanent" erroneous behavior. More precisely, the controller receiving the outputs of the voters and managing the reconfiguration, determines whether the fault occurred in a temporary register used for computation or in the configuration memory by monitoring the voter outputs in

consecutive cycles, and in the latter case it triggers a re-configuration of the portion of the FPGA affected by the fault.

In order to obtain this kind of system, the initial specification is partitioned into modules $m_1, m_2, \ldots, m_n$, which to apply TMR to. A portion f_i may host one or more modules $\{m_j\}$, eventually the voter, and it is possible to recover from a fault in the configuration bitstream controlling the functionality performed by the modules placed in f_i by applying partial bitstream b_i. As a consequence, when the system is being designed, the complete bitstream is generated, together with all the partial bitstreams $b_1, b_2, \ldots, b_k$ used to re-configure only the single portions of the FPGA.

The possible alternatives in placing modules and their replicas into the same f_i or in different ones (together with all the additional information and structures to correctly implement them avoiding a stand-by of the part of the system not affected by the fault) are explored evaluating advantages and costs, as described in the next subsection.

3.1 Solution Space Exploration

This approach, although simple in principle, requires a set of a complex design tasks, from the partitioning of the entire system into independently controlled sub-systems (a trade-off has to be identified since the partition size impacts onto the control section, the error latency and, the reconfiguration time) to the placing of this sub-specification into different FPGA areas. When considering the partitioning task, on one hand there is the solution where the system is not partitioned and is monitored only on the primary outputs (corresponding to a complete reconfiguration solution), and on the opposite hand there is a partitioning with comparison of every single functionality. In between there are several alternatives, each one characterized by its costs and benefits. More precisely, the size of such subsystems determines the precision of fault localization and some considerations need be introduced: i) the localization capability grows with the number of subsystems independently controlled; ii) the area overhead grows with the number of subsystems independently controlled; and iii) it is necessary to be able to control the process so that the identified subsystems are placed in separately reconfigurable portions of the device. Different partitioning solutions for the given device lead to alternative designs characterized by different costs due to the varying number and size of the comparators and the amount of shared logic. Figure 1 shows (some solutions of) the application of the TMR at different levels of granularity:

a) the three replicas are grouped on the same reconfigurable portion, the voter occupies the adjacent frame and based on the detection one of the two portions is re-configured;
b) the three replicas are placed on different frames and the corrupted one is re-configured;
c) each frame hosts three replicas and their voter and the corrupted stage is re-configured.

The proposed methodology aims at estimating the costs and benefits deriving from the possible different partitioning solutions. This exploration of the solution space is based on the evaluation of the following elements: i) size of the subsystems (used to estimate area and ro-indices), ii) size of the data widths to be compared (used to derive the size of the comparators), and iii) amount of the minimal reconfiguration portion of the FPGA (used to derive available area and reconfiguration times). A preliminary synthesis of the initial device partitioned into subsystems is performed, to identify the basic costs to perform the estimations. Starting from this finer-grain partitioning, incremental aggregation of modules is performed to find a satisfying trade-off between area overheads and partial reconfigurable

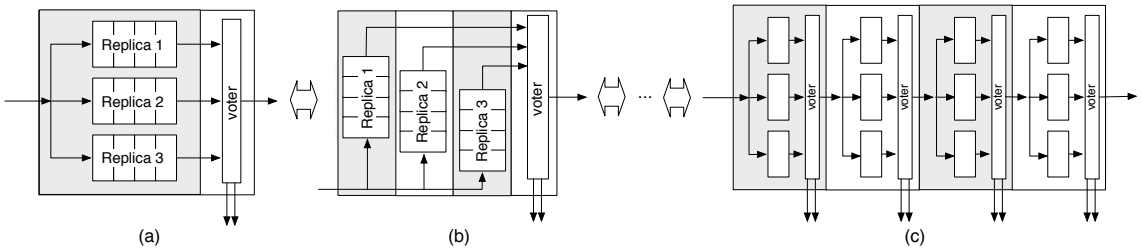

Figure 1. TMR applied with different levels of granularity (separately re-configurable adjacent frames have different background color).

times. The controller is pre-defined, since its behavior depends only on the number of error signals pairs to be monitored and the portions of FPGA to be separately managed.

Some solutions may not be interesting; considering the situation depicted on the left of Figure 2 the benefits of knowing precisely which module ($f_1 i$ or $f_2 j$) is faulty is not significant since the entire portion will be re-configured. As a result the first voter could be avoided allowing a more limited area overhead with the same coverage and partial reconfiguration control.

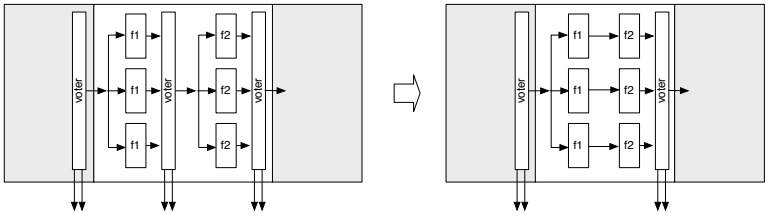

Figure 2. Disadvantageous partitioning solution.

Another aspect that has to be considered refers to the constraints which impose a horizontal placement multiple of four slices [11]; thus, we have to define a placement-area that can be characterized by the presence of a number of available resources greater than the ones needed by the modules to be placed in. This consideration leads to the definition of different approaches to reduce such a waste of resources. The *resources optimization index* parameter, *ro-index* [5], is defined as the ratio between the number of the required slices to implement a component and the number of resources (slices) available in the placement-area computed to implements such design. We can consider this parameter as the measurement of the fragmentation computed on each pair (component-area, placement-area), having a value between 0 and 1, where all the available resources in the placement-area are not used at all and all used to implement the functionality, respectively.

In such a scenario a support for the semi-automatic exploration of the solution space is in order, due to the relevant number of alternatives and aspects to be taken into account. The proposed methodology provides such a support, taking in input the initial design specification and estimating the raw costs of the implementation of the modules and the voters (Figure 3). Given these values it generates the various solutions and computes the various parameters characterizing it; the designer can thus select the most promising solution and proceed with its actual realization.

The next section presents the application of the proposed approach to a case study, a filter, evaluating three possible designs with fault mitigation properties.

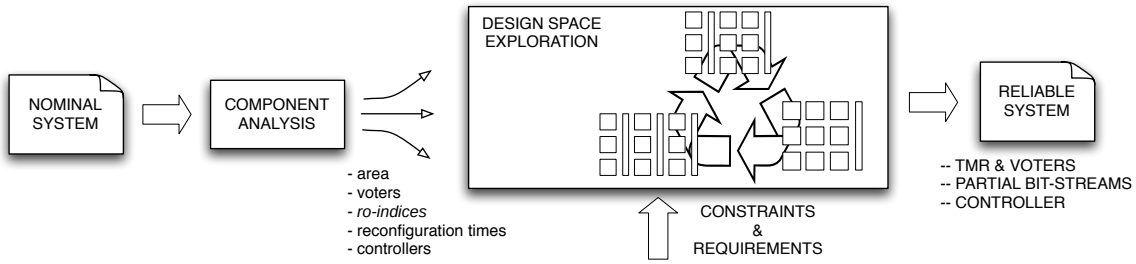

Figure 3. Design Space Exploration w.r.t to TMR and Partial Reconfiguration.

3.2 Experimental Results

Two experimental sessions have been carried out to evaluate the proposed approach. The purpose of the first one is to test the application of the fault mitigation strategy based on the TMR coupled with the partial reconfiguration; the test architecture is a simple adder on an FPGA, communicating with a PC by means of the serial port. The final system is composed by a fixed area, where a control FSM is placed together with an UART controller and a TMR voter, and three reconfigurable areas, each one containing an adder replica. The controller receives operands from the serial port and issues an add operation; when results are ready, the voter computes the output by majority vote and eventually signals the occurrence of a failing module. A program, executed on the PC, feeds the system on the FPGA and on the basis of the error detection response, performs a reconfiguration of the faulty area without suspending the execution. The described architecture has been implemented by exploiting the EARP on a Spartan3 FPGA connected to a standard desktop computer. A specific tool designed for manipulating bitstreams has been used for inserting faults into the reconfigurable area bitstream; it is worth noting that in this experimental scenario reliability issues related to the controller have not been taken into account.

In the second experimental session a FIR architecture has been taken into account for an exploration of the solution space where several different partitionings of the system have been analyzed and evaluated. The FIR algorithm is based on the $y(t) = \sum_{i=0}^{10} c_i \cdot x(t–i)$ formula; its architecture has a simple data-path working on 27 bits and with no control FSM; it is composed by a sequential chain of 10 registers storing input values for a 10 clock cycle time window and a combinatorial net constituted by multipliers and adders. In each clock cycle, the value stored in the i-th register is multiplied with the a i-th coefficient and the result is added to the result of the i–1-th stage. It is worth noting that the regular and modular structure of the considered testing architecture is particularly suited for proposing and evaluating several solutions, each one constituted by a different partitioning.

We have implemented several variants; the first solution applies the TMR at system level, where each replica is placed on a different reconfigurable area, as well as the voter, obtaining an architectural solution such as the one represented in Figure 1b (solution labeled "Sys-Level"). On the opposite side of the solution space, there is the solution characterized by the triplication of each stage of the data-path with a majority voter, placing each one of these *reliable* stages on a different reconfigurable area (similarly to Figure 1c); this solution is identified as "10-Stages" solution. In between there are various alternatives given by the aggregation of two or more adjacent stages, to evaluate the trade-offs, identified with the number of resulting stages placed on independently reconfigurable portions of the FPGA.

All implementations have been carried out on the Virtex-II Pro VP7 platform, due to the

	FIR implementation					
	Base	Sys-Level	6-Stages	7-Stages	9-Stages	10-Stages
# slices (out of 6144)	325	1368	1761	1830	1968	2043
Part. bitstream size (Kbyte)	N/A	100 (11)	51 ÷ 82	49 ÷ 68	38 ÷ 61	27 ÷ 60
Part. reconf. time (ms)	N/A	66 (7)	34 ÷ 55	33 ÷ 45	25 ÷ 40	17 ÷ 40
ro-index		0.69 (0.15)	0.46 ÷ 0.83	0.46 ÷ 0.83	0.53 ÷ 0.83	0.37 ÷ 0.83

Table 1. Experimental results

relevant area requirements; area as well as reconfiguration data (bitstream size and timing) are reported in Table 1, where the nominal solution is compared with the two proposed ones. For the System Level solution, the information between brackets refers to the portion of FPGA hosting the voter).

TMR is expensive in terms of area, as expected, requiring 4 times the nominal circuit area in the first solution and almost 5 times in the 10-Stages one (characterized by 10 voters comparing 18-bit and 9-bit intermediate signals rather than a single one comparing the 18-bit output); furthermore, the impact of the intermediate voters is quite relevant and the grouping of stages has proven to be effective to limit such overhead. When considering recovery timing, the complete reconfiguration of the board requires in all situations 364 ms (for the complete 548 Kbytes bitstream), whereas partial reconfiguration to recover the specific corrupted portion of the FPGA is characterized by an interesting timing. The 10-Stages version has varying times; some stages are quite big and thus the required time grows to 60 ms, but for several stages there is a significant improvement.

As for the intermediate solutions, depending on aggregated stages, there are different costs and benefits; based on the designers' preferences (s)he can select the most interesting solution identified during the design space exploration.

When considering other approaches applied to the FIR case study, *scrubbing* would require a reconfiguration time of 364 ms, corresponding to a complete reconfiguration, without accounting the execution suspension during bitstream downloading. *Readback* is characterized a high fault detection latency, caused by the periodic check of the bitstream, since a complete readback for the considered architecture takes approximately the same time for downloading the bitstream. In addition, the two approaches cannot detect errors in data and control registers. Finally, when comparing the proposed approach with the one based on duplication with comparison and partial reconfiguration [5], such an approach offers a lower area overhead but is not able to distinguish between faults affecting data and control registers and faults affecting configuration registers, requiring, therefore, a reconfiguration each time an error is detected and thus suspending the execution.

4 Conclusions and future work

This paper presents an approach allowing the exploration of the solution space in the design of reliable system on FPGAs, based on the application of the TMR passive hardware redundancy technique, coupled with partial dynamic reconfiguration to recover from the occurrence of soft errors, affecting either the FPGA configuration memory or its temporary

memory elements, used for the application computation. Experimental results show the feasibility of the approach as well as the support for design space exploration. This approach for achieving SEU fault mitigation properties, added to the one previously presented in [5] contributes in our long-term goal of defining a complete framework for selecting the most convenient redundancy technique, applied with a granularity level driven by the various parameters, while pursuing the design of reliable systems on FPGAs. Current ongoing work consists in studying new voter architectures able to achieve additional localization functionality and in applying the proposed methodology to complex designs. Future work will focus also on the introduction of additional parameters, such as power or performance, for driving the exploration of the design space.

References

[1] M. Dyer and M. Wirz. *Reconfigurable System on FPGA*. Diploma Thesis, Winter Term, 2001/2002.

[2] Xilinx Inc. http://www.xilinx.com.

[3] F. Lima Kastensmidt, L. Sterpone, M. Sonza Reorda, and L. Carro. On the Optimal Design of Triple Modular Redundancy Logic for SRAM-Based FPGAs. In *IEEE Proc. Design, Automation and Test in Europe Conference*, pages 1290–1295, 2005.

[4] C. Carmichael. *Triple Module Redundancy Design Techniques for Virtex FPGAs*. Xilinx Application Notes 197, 2006.

[5] C. Bolchini, D. Quarta, and M. Santambrogio. SEU Mitigation for SRAM-Based FPGAs through Dynamic Partial Reconfiguration. In *Proc. ACM/IEEE Great Lake Symposium on VLSI*, pages 55–60, 2007.

[6] H. R. Zarandi, S. G. Miremadi, C. Argyrides, and D. K. Pradhan. Fast SEU Detection and Correction in LUT Configuration Bits for SRAM-Based FPGAs. In *Proc. 14th IEEE Reconfigurable Architecture Workshop*, 2007.

[7] C. Carmichael, E. Fuller, P. Blain, and M. Caffrey. SEU mitigation techniques for Virtex FPGAs in space application. In *MAPLD99 Poster*, page 24, 1999.

[8] C. Carmichael, M. Caffrey, and A. Salazar. *Correcting Single-Event Upsets Through Virtex Partial Configuration*. Xilinx Application Notes 216, 2000.

[9] M. Gokhale, P. Graham, E. Johnson, N. Rollinsand, and M. Wirthlin. Dynamic reconfiguration for management of radiation-induced faults in FPGAs. In *Proc. 18th Intl. Parallel and Distributed Processing Symposium*, pages 145–150, 2004.

[10] F. Ziegler et al. Terrestrial cosmic rays and soft errors. *IBM Journal of Research and Development*, 40(1), 1996.

[11] D. Lim and M. Peattie. *Two Flows for Partial Reconfiguration: Module Based or Difference Based*. Xilinx Application Notes 290 (v.1.2), May 2004.

[12] D. P. Sieworek and R. S. Swarz. *The Theory and Practice of Reliable System Design*. Digital Press, 1982.

[13] S. D'Angelo, C. Metra, S. Pastore, A. Pogutz, and G.R. Sechi. Fault-tolerant voting mechanism and recovery scheme for TMR FPGA-based systems. In *Proc. 13th IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems*, pages 233–240, 1998.