Reducing Pin and Area Overhead in Fault-Tolerant FPGA-based Designs

Fernanda Lima Luigi Carro Ricardo Reis

Universidade Federal do Rio Grande do Sul
PPGC - Instituto de Informática - DELET
Caixa Postal: 15064, CEP 91501-970 - Porto Alegre - RS - Brazil
+55 51 33 16 70 36
<fglima, carro, reis>@inf.ufrgs.br

ABSTRACT

This paper proposes a new high-level technique for designing fault tolerant systems in SRAM-based FPGAs, without modifications in the FPGA architecture. Traditionally, TMR has been successfully applied in FPGAs to mitigate transient faults, which are likely to occur in space applications. However, TMR comes with high area and power dissipation penalties. The proposed technique was specifically developed for FPGAs to cope with transient faults in the user combinational and sequential logic, while also reducing pin count, area and power dissipation. The methodology was validated by fault injection experiments in an emulation board. We present some fault coverage results and a comparison with the TMR approach.

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance

General Terms

Design, Performance, Reliability.

Keywords

Fault-tolerance, FPGA.

1. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are increasingly demanded by spacecraft electronic designers, because of their high flexibility in achieving multiple requirements such as high performance, no NRE (Non Refundable Engineering) cost and fast turnaround time. In particular, SRAM-based FPGAs are uniquely suited for remote missions, because they can be reprogrammed by the user as many times as necessary in a very short period. As a result, SRAM-based FPGAs offer the additional benefits of allowing in-orbit design changes, thus reducing the mission cost by correcting errors or improving system performance after launch.

FPGA'03, February 23-25, 2003, Monterey, California, USA.

Transient faults, also called Single Event Upset (SEU), are the major concern in space applications [1], with potentially serious consequences for the spacecraft, including loss of information, functional failure, or loss of control. SEU occurs when a charged particle hits the silicon, transferring enough energy to provoke a bit flip in a memory cell or a transient logic pulse in the combinational logic. SEU on devices has become more frequent because of smaller transistor features achieved by the continuous technology evolution. As a result, not only space applications but also terrestrial applications that are critical, such as bank servers, telecommunication servers and avionics, are more and more considering the use of tolerant techniques to ensure reliability [2, 3].

SRAM-based FPGA will be the focus of this work, more specifically the Virtex® family [4]. SEU has a peculiar effect in SRAM-based FPGAs when a particle hits the user's combinational logic. In an ASIC, the effect of a particle hitting either the combinational or the sequential logic is transient; the only variation is the time duration of the fault. A fault in the combinational logic is a transient logic pulse in a node that can disappear, according to the logic delay and topology. In other words, this means that a transient fault in the combinational logic may or may not be latched by a storage cell. Faults in the sequential logic manifest themselves as bit flips, which will remain in the storage cell until the next load. On the other hand, in a SRAM-based FPGA, both the user's combinational and sequential logic are implemented by customizable logic cells, in other words, SRAM cells. When an upset occurs in the combinational logic, hitting either the logic or the routing, it has a transient effect followed by a permanent effect, because the SRAM cell that implements that logic or controls that routing has flipped. This means that a transient upset in the combinational logic in a FPGA will be latched by a storage cell, unless some detection technique is used. When an upset occurs in the user sequential logic, it has a transient effect, because the fault can be corrected in the next cell load. Accordingly, the use of SEU mitigation techniques for programmable architectures must take into account these peculiarities.

In order to mitigate SEU in Virtex® family [4] FPGAs, the Triple Modular Redundancy (TMR) with voting technique, combined with bitstream scrubbing has been applied [5, 7]. TMR is a suitable technique for SRAM-based FPGAs, because of its full hardware redundancy property in the combinational and sequential logic. Previous results from bitstream fault injection [6] and radiation ground testing presented in [7] showed that the use of

TMR in Virtex FPGAs has confirmed the efficacy of the TMR structure combined with scrubbing to recover upsets in the FPGA architecture. However, the TMR technique presents some limitations, such as area overhead, three times more input and output pins and, consequently, a meaningful increase in power dissipation.

In this work we propose a SEU tolerant technique for SRAM-based FPGAs to cope with both problems above described: power and pins overhead caused by TMR, and the permanent effect of an upset in the user's combinational logic. This paper is organized as follows. Section II shows some used SEU tolerant techniques applied to FPGAs and to ASICs, more specifically hardware redundancy and time redundancy techniques. Section III introduces the new technique that combines duplication with comparison (DWC) and time redundancy to reduce pin count penalties. Section IV presents the evaluation results performed by fault injection experiments developed using a prototype board, and a comparison to the TMR approach. Conclusions and ongoing works are discussed in section V.

2. PREVIOUS WORK

Several SEU mitigation techniques have been proposed in the past years in order to avoid transient faults in digital circuits, including those implemented in programmable logic. A SEU immune circuit may be accomplished through a variety of mitigation techniques based on redundancy. Redundancy is provided by extra components (hardware redundancy), by extra execution time or different time of storage (time redundancy), or by a combination of both. Each technique has some advantages and drawbacks, and there is always a compromise between area, performance, power and fault tolerance efficiency.

In the case of SRAM-based FPGAs, the problem of finding an efficient technique in terms of area, performance and power is very challenging, because of the high complexity of the architecture. As previously mentioned, when an upset occurs in the user's combinational logic implemented in a FPGA, it provokes a very peculiar effect, not commonly seen in ASICs. The SEU behavior is characterized as a transient effect, followed by a permanent effect. The upset can affect either the combinational logic or the routing. The consequences of this type of effect, a transient followed by a permanent fault, cannot be handled by the standard fault tolerant solutions used in ASICs, such as Error Detection and Correction Codes (Hamming code) or the standard TMR with a single voter, because a permanent fault in the encoder or decoder logic or in the voter would invalidate the technique provoking an error in the circuit.

Special techniques should be developed for FPGAs to cope with this type of effect. An example is the TMR proposed in [5]. Section 2.1 discusses this technique and some penalty points that could be improved if some SEU mitigation techniques for ASICs are adapted for FPGAs. Section 2.2 discusses some time and hardware redundancy proposed in the past for ASICs. It is important to notice that the studies presented in this paper do not change the FPGA matrix architecture of the FPGA. They are all applied in high-level design descriptions, such as VHDL or Verilog.

2.1 Full Hardware Redundancy (TMR) for FPGAs

Virtex devices consist of a flexible and regular architecture, composed of an array of configurable logic blocks (CLBs), surrounded by programmable I/O blocks, all interconnected by a hierarchy of fast and versatile routing resources [4]. The CLBs provide the functional elements for constructing logic, while the I/O blocks provide the interface between the package pins and the CLBs. The CLBs are interconnected through a general routing matrix (GRM), that comprises an array of routing switches located at the intersections of horizontal and vertical routing channels. The Virtex matrix also has dedicated memory blocks of 4096 bits each, clock DLLs for clock-distribution delay compensation and clock domain control, and two 3-State buffers (BUFTs) associated with each CLB.

Virtex devices are quickly programmed by loading a configuration bitstream (collection of configuration bits) into the device. The device functionality can be changed at anytime by loading in a new bitstream. This continuous bitstream loading is called scrubbing. The bitstream is divided into frames, and it contains all the information needed to configure the programmable storage elements in the matrix located in the Look-up tables (LUTs), CLBs flip-flops, CLBs configuration cells and interconnections, as shown in figure 1. All these configuration bits are potentially sensitive to SEU and consequently, they must be protected by a SEU tolerant technique.

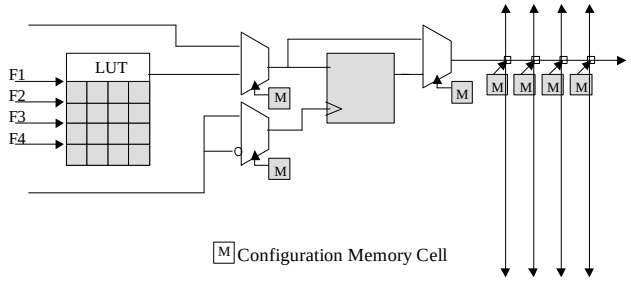


Figure 1. Example of SEU sensitive bits in the CLB tile (shaded area)

The SEU mitigation technique used nowadays to protect designs synthetized in Virtex® architecture is mostly based on TMR combined with scrubbing. The TMR mitigation scheme uses three identical logic circuits (redundant block 0, redundant block 1 and redundant block 2), synthetized in the FPGA, performing the same task in tandem, with corresponding outputs being compared through a majority vote circuit. The TMR technique for Virtex® is presented in details in [5], and more examples are also presented in [6, 8]. Figure 2 shows the TMR scheme. The user's flip-flop is replaced by a TMR structure composed of three flip-flops, three voters and multiplexors that are implemented using LUTs, one for each. The combinational logic and input and output pins are also triplicated to avoid any single point of failure inside the FPGA, as illustrated in figure 2. In this way, any upset inside the matrix can be voted out by the TMR structure assuring correct values in the output. Figure 3 shows the output pad scheme of the TMR approach.

The upsets in the FPGA matrix can be corrected by loading the original bitstream (scrubbing). Scrubbing allows a system to

repair SEUs in the configuration memory without disrupting its operations. It is performed continuously, without interruption of the system operation. However, because scrubbing does not interrupt the normal device operation, the original bitstream values cannot be loaded in the flip-flops and embedded memory structures, as these cells are constantly being updated by the application. Consequently, if an upset occurs in the CLB flip-flop, the TMR structure (figure 2) is responsible for voting and correcting the fault by itself. If an upset occurs in the user combinational logic, TMR votes out the logic block no longer functioning, while the reconfiguration (scrubbing) repairs the upset before more upsets accumulate and overcome the TMR.

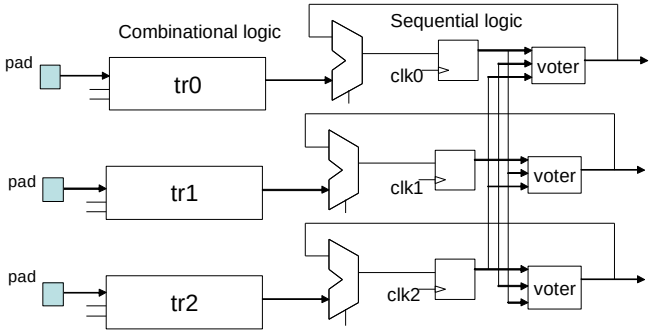


Figure 2. Triple Modular Redundancy Scheme for Combinational and Sequential blocks [5]

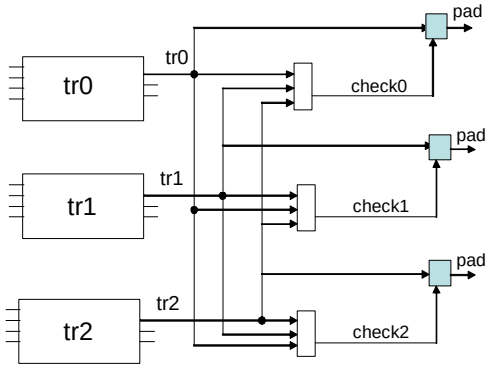


Figure 3. Triple Modular Redundancy Scheme for output pads [5]

The effect of an upset in the FPGA matrix is unique for each type of programmable structure, such as LUT cells, routing cells, flip-flop cells and embedded memory cells. In order to assure reliability, the effects of an upset should be isolated inside a single redundant block, and must not propagate, affecting two or more blocks. For example, upsets in LUTs and CLB flip-flops are restricted to the redundant block affected by the upset, because they provoke an error only in the combinational logic defined by that LUT. Upsets in the embedded memory are isolated in the memory. However, the rate of the memory scrubbing circuit implemented by the user in the FPGA is very important to avoid accumulation of upsets that could overcome the TMR reliability. Upsets in the routing can connect or disconnect signals causing a short circuit or an open circuit, respectively. The effect of an open circuit is only harmful for the redundant block to which the signal belongs. However, the effect of a short circuit can be destructive for the entire design, if two signals from two different redundant blocks are connected. Fault injection studies show that a very low percentage of short circuits (upsets in the routing), less than 1%, can provoke error in the TMR output [6]. The use of assigned area constraints for each redundant block can reduce the probability of short circuits between signals from distinct blocks.

TMR is an attractive solution for SRAM based FPGAs because it only changes the high level design description. It does not require changes at the mask level. On the other hand, because it does not change the FPGA design by itself, it presents some limitations. Many applications can accept these limitations but some can not. The main limitations are:

- The number of I/O pads available for designers is reduced by three, because each input and output of each TMR redundant block (tr0, tr1, tr2) should have its own input and output pads, as shown in figure 2. The number of dedicated clock resource segments for the routing available for designers is also reduced by three, because each input and output of each TMR redundant block (tr0, tr1, tr2) should have its own clock.
- The size of the combinational logic in the design is multiplied by three, and this also happens in the sequential logic, where each storage cell must be replaced by three storage cells, with three voters and multiplexors, as shown in figure 2.
- The embedded memory also needs to be triplicated and refreshed using extra logic.
- There is a delay overhead inserted by the voters.
- The power consumption is increased by three times as all input and output pins as well as the combinational and sequential logic are triplicated.

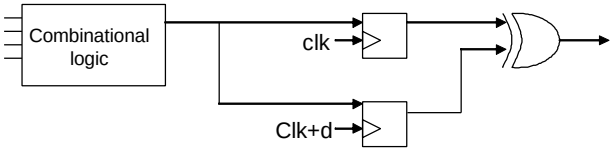
In conclusion, TMR is a suitable solution for FPGAs because it provides a full hardware redundancy, including the user's combinational and sequential logic, the routing, and the I/O pads. However, it comes with some penalties because of its full hardware redundancy, such as area, I/O pad limitations and power dissipation. Although these overheads could be reduced by using some SEU mitigation solutions such as hardened memory cells [9, 10], EDACs techniques and standard TMR with single voter [11], these solutions are costly because they require modifications in the matrix architecture of the FPGA (NRE cost). This paper will show a high level technique that combines TMR with time and hardware redundancy, with some extra features able to cope with the upset effects in FPGAs, allowing the reduction of the number of I/O pads and consequently power dissipation in the interface.

2.2 Time and Hardware Redundancy for ASICs

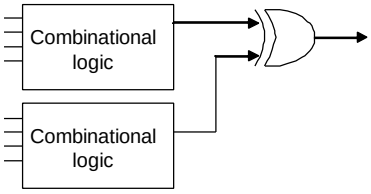
Time and hardware redundancy techniques are largely used in ASICs [12]. The techniques range from simple upset detection to upset voting and correction. There is a wide possibility of techniques according to the user's application requirements. Sometimes it is just necessary to warn the presence of upset with an interruption in the system functionality, while sometimes it is required to completely avoid interruptions, assuring full reliability. There is a set of techniques that can present reliability in between

these two extremes, each one producing more or less overhead according to its fault reliability.

The use of time and hardware redundancy has already been proposed in the past [13, 14, 15] for the detection of transient upsets and upset tolerance. The goal is to take advantage of the transient pulse characteristic to compare signals at two different moments. The output of the combinational logic is latched at two different times, where the clock edge of the second latch is shifted by time *d*. A comparator indicates an upset occurrence (error detection). The scheme is illustrated in figure 3a. Hardware redundancy as duplication with comparison (DWC) can also be used for upset detection, as shown in figure 3b. Although it presents a larger area overhead compared to time redundancy, its detection scheme for a full coverage of upset detection is simpler to apply.



(a) Time Redundancy



(b) Hardware Redundancy

Figure 4. Examples of Upset Detection schemes

The possibility of applying time redundancy combined with hardware redundancy technique for FPGAs looks interesting to reduce the costs of using full hardware redundancy: TMR. Potentially the use of DWC with time redundancy may reduce area and pin count in comparison to TMR. But there are two problems to be solved. First, previous techniques can only be used to detect transient upsets, and not transient upsets that become permanent, as in the case of SRAM based FPGAs. Second, in the FPGA, it is not only sufficient to detect an upset, but one also must be able to vote the correct value in order to assure the correct output. In the next section, we present a technique based on time and hardware redundancy for SRAM-based FPGA that takes into consideration the above problems to reduce pin count, area and power dissipation.

3. REDUCING PIN AND AREA OVERHEARD BY USING TIME AND HARDWARE REDUNDANCY

There are always some kinds of penalties to be paid when designing fault tolerant systems. The penalties come from the redundancy. Full hardware redundancy approach can produce larger area, pin and power overheads. Time redundancy can provoke interruption of the system in the presence of fault, and performance penalties. The question is: what would happen if some hardware redundancy blocks are replaced by time redundancy?

Aiming to reduce the number of pins overhead of a full hardware redundancy implementation (TMR), and at the same time coping with permanent upset effects, we present a new technique based on time and hardware redundancy, where triple modular redundancy (TMR) and double modular redundancy with comparison (DWC) are combined with time redundancy to detect transient faults in the programmable matrix (SEUs in the programmable elements). The upset detection and voter block is a state-machine able to detect upsets and to identify the fault-free redundant block (correct value) to allow continuous operation. The main objective is to reduce input and output pins, and consequently power dissipation in the I/O pads, the main drawbacks of the TMR approach. Moreover, it can present some area reduction for some designs composed of large combinational logic structures.

Time redundancy by itself can only detect transient faults. The same occurs with duplication with comparison (DWC), which can also only detect faults. However, the combination of time redundancy and DWC can provide an interesting upset evaluation, which can not only detect the presence of a fault, but also recognize in which redundant block the upset has occurred. Figure 5 shows the detailed scheme. There are two redundant blocks: dr0 and dr1. In this way, upsets in the combinational logic can be detected and voted before being latched.

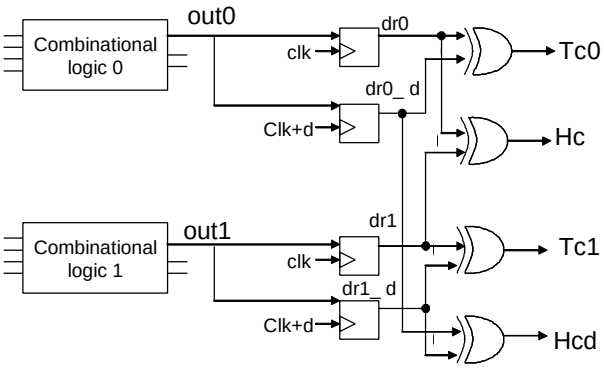


Figure 5. Time and Hardware Redundancy Schematic for Upset Detection

Four values are stored in the auxiliary latches (dr0, dr0_d, dr1 and dr1_d), two from each redundant block collected in different instant. Two latches store the dr0 and the dr1 outputs at the clock edge and two latches store the dr0 and dr1 outputs at the clock edge plus a delay *d*. As a consequence, there are four outputs of comparators in the scheme: Tc0 is the time redundancy comparator from redundant block 0, Hc is the hardware redundancy comparator at the clock edge, Tc1 is the time redundancy comparator from redundant block 1 and Hcd is the hardware redundancy comparator at the clock+*d* edge. Analyzing the sixteen possibilities of output combinations of dr0, dr0_d, dr1 and dr1_d, eight different syndromes are recognized, as presented in table 1. Analyzing the syndromes from table 1, it is possible to see the temporal effect of an upset in the FPGA. The steps are basically no fault, transient upset effect in redundant block 0 (dr0) or block 1 (dr1), permanent effect in redundant block 0 (dr0) or block 1 (dr1), recovery upset effect in redundant block 0 (dr0) or block 1 (dr1), and no fault.

An upset in redundant block 0, syndrome 1001, is characterized by a transient variation in the output (Tc0=1) with no changes in dr1 outputs (Tc1=0), and in addition Hc=0 and Hcd=1. An upset occurrence in dr1 is recognized in an equivalent way, where Tc1=1 and Tc0=0. There are many other syndromes that are not commonly seen in an ASIC environment, only in FPGAs. One example is the permanent effect of an upset, syndrome 0101. By analyzing this syndrome, it is not possible to conclude which redundant block has the correct value and which does not. The previous syndrome characterized by the transient effect detection is necessary to vote the correct path. This phenomenon characterizes the necessity of a state machine to vote the correct value. This technique considers only one upset per design at once, either in redundant block 0 or in redundant block 1. An implementation with an assigned area constrain may avoid the occurrence of a fault in the redundant block 0 at the same time of a fault in redundant block 1 (syndrome 1010). The identification of this syndrome can be used as a flag to show that upsets have overcome the DMR scheme.

The DWC with time redundancy proposed technique for the combinational blocks, illustrated in figure 6, combines duplication with comparison and time concurrent error detection to identify combinational upsets in FPGAs. DWC with time redundancy tolerates upsets without system interruptions. The combinational logic is duplicated and there are detection and voter circuits able to detect an upset and to identify which redundant block should be connected to the CLB flip-flops. The upsets in the combinational logic are corrected by scrubbing, while upsets in the CLB flip-flops are corrected by the TMR scheme. It is important to notice that for upset correction, scrubbing is performed continuously, to assure that only one upset has occurred between two reconfigurations in the design.

Table 1. Syndrome Analysis in the Double Modular Redundancy Approach

dr0	dr0_d	dr1	dr1_d	Tc0	Hc	Tc1	Hcd	Syndrome
0	0	0	0	0	0	0	0	No fault
0	0	0	1	0	0	1	1	Fault dr1 (stage 1, transient)
0	0	1	0	0	1	1	0	Fault recovery dr1
0	0	1	1	0	1	0	1	Fault dr0 or dr1 (stage 2, permanent)
0	1	0	0	1	0	0	1	Fault dr0 (stage 1, transient)
0	1	0	1	1	0	1	0	Fault dr0 and dr1 (stage 1, transient)
0	1	1	0	1	1	1	1	Fault dr0 or dr1, recovery dr0 or dr1
0	1	1	1	1	1	0	0	Fault recovery dr0
1	0	0	0	1	1	0	0	Fault recovery dr0
1	0	0	1	1	1	1	1	Fault dr0 or dr1, recovery dr0 or dr1
1	0	1	0	1	0	1	0	Fault dr0 and dr1 (stage 1, transient)
1	0	1	1	1	0	0	1	Fault dr0 (stage1, transient)
1	1	0	0	0	1	0	1	Fault dr0 or dr1 (stage 2, permanent)
1	1	0	1	0	1	1	0	Fault recovery dr1
1	1	1	0	0	0	1	1	Fault dr1 (stage1, transient)
1	1	1	1	0	0	0	0	No fault

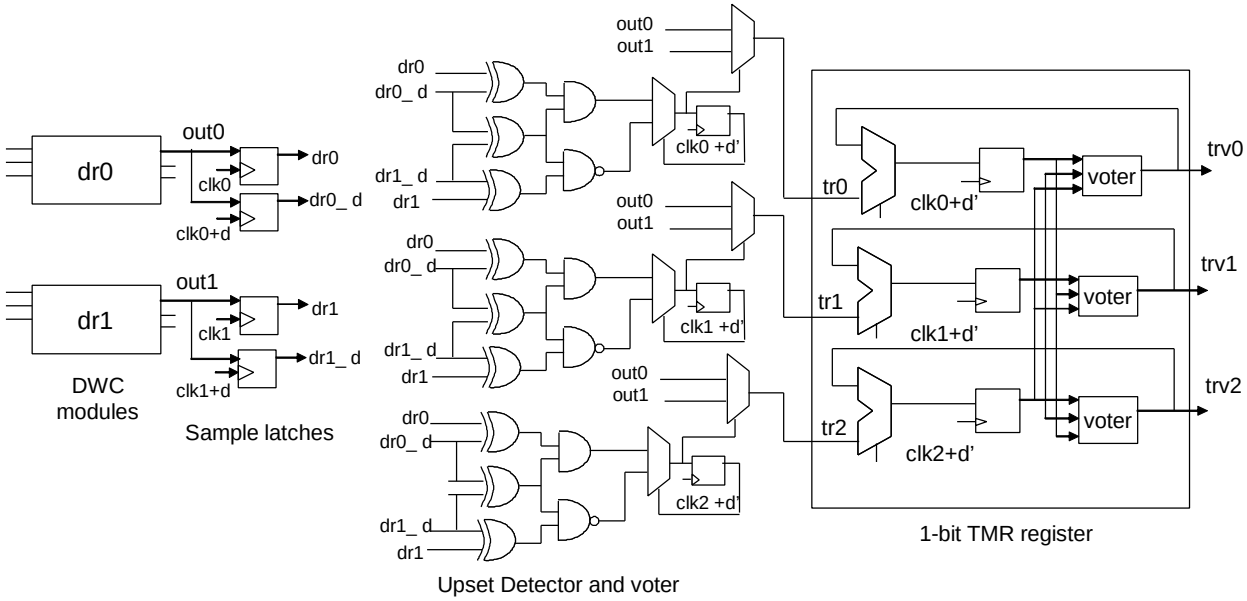


Figure 6 DWC with time redundancy proposed technique scheme for one bit output

When the circuit is reset, the state machine starts in state 0 and it is persistently monitoring the redundant block 0, while the redundant block 1 is the spare. At this point, an upset in the redundant block 1 will not affect the system, and it will be corrected by the periodic scrubbing. If an upset occurs in the redundant block 0, the state machine recognizes the fault, and the operation switches to the spare path, the redundant block 1. The upset in the redundant block 0 will be soon corrected by the scrubbing, while now the system is operating with the redundant block 1. At this point, the state machine is constantly monitoring the redundant block 1, looking for upsets and the redundant block 0 is the spare. Upsets in the redundant block 0 will be corrected by scrubbing.

As the fault detection technique used for this method is able to identify only transient faults, it is necessary to have an observation period to detect the fault occurrence and consequently the faulty module (dr0 or dr1). The size of the observation period of a fault occurrence is referred to as the clock delay *d*. As a result, the transient fault observability occurs between clock and clock+*d*. Outside this observation period, the fault is seen as permanent and the faulty module can not be recognized, only the presence of a fault can be detected. The percentage of faulty module detection is related to *d*. As the observation period (*d*) becomes greater, the probability of faulty module detection becomes higher. One can use *d* as half of one clock cycle. The performance penalty of this method is related to the time duration of the fault observability (*d*).

The registers from the sequential logic store the combinational logic outputs at clock plus *d*, plus the delay from the upset detection circuit, totaling a new delay d'. The latches from the concurrent upset detection state machine will also store the next state at clock+*d'*. In order to simplify the number of clocks in the design, one possibility is to reduce the frequency of the design by two. In this way, the combinational output is stable at the clock falling edge. At this time, the value is captured for the future comparison. The fault observability period is until the next clock rising edge where the correct redundant logic is voted. Figure 7 illustrates two fault effects, one occurring during the propagation time and one occurring during the observation time.

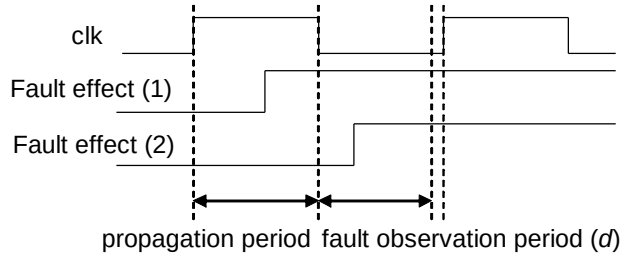


Figure 7. Fault effect in the clock period

If a fault effect occurs during the propagation period, the DWC with time redundancy scheme will detect an error but will not be able to recognize which redundant block (dr0 or dr1) is faulty. However, some fault effects occurring during the propagation period can be tolerated, if they affect the spare redundant logic that is not being observed at that time. The fault can be corrected in the next scrubbing and no error may occur. If a fault effect occurs during the observation time, the DWC with time redundancy scheme will be able to detect the output variation and vote the fault-free redundant module (dr0 or dr1). Faults in the observation time will always be correctly voted, except for those whose effect will not be manifested at the time. For example, a stuck at one fault in a node that is already one because of the input vectors.

Some constrains must be observed for the perfect functioning of the technique. There constrains are the same as TMR:

- There must be only one upset per dual modular redundancy (DMR) combinational logic, including the state machine detection and voting circuit, consequently it is important to use some assigned area constraints to reduce the probability of short circuits between redundant block 0 and 1 (dr0 and dr1).
- The scrubbing rate should be fast enough to avoid accumulation of upsets in two different redundant blocks.

Upsets in the detection and voting circuit do not interfere with the correct execution of the system, because the logic is already triplicated. In addition, upsets in the latches of this logic are not critical, as they are refreshed in each clock cycle. Assuming a single upset per chip between scrubbing, if an upset alters the correct voting, it does not matter as long as there is no upset in both redundant blocks.

This technique can be used as an additional option for the TMR technique for designing reliable circuits in FPGAs with pads and power reduction. Because the combinational circuit is just duplicated, inputs and outputs can be duplicated instead of triplicated, as in the TMR approach. However, it is important to notice that the TMR inputs related to the user's sequential logic used in the CLB flip-flops are not changed as triple input clocks, reset and user vdd and gnd [8].

The upset detector and voter circuit can be optimized in terms of area. In figure 6, the upset detector and voter circuit are represented for only one bit. However, it is possible to use the circuit for groups of bits. In this way, only one state-machine per TMR redundant part for each group of bits is necessary, as presented in figure 8.

Another possible optimization is to use a single state machine to vote just the input of the redundant block 2 of the TMR register, as presented in figure 9. In this way, a fault in one of the combinational redundant blocks (dr0 or dr1) is voted to the tr2 input, assuring the correct operation. A fault in this upset detection and voter block will corrupt just the redundant block 2 of the TMR (tr2), consequently, tr0 and tr1 will still vote the correct value. The scheme presented in figure 9 also shows the clock optimizations, where the sample storage cells are latched at the clock falling edge (clk0, clk1) and the state machine of the upset detection block is latched at the clock rising edge (clk2). The three clocks are the same, and are all connected outside the FPGA chip.

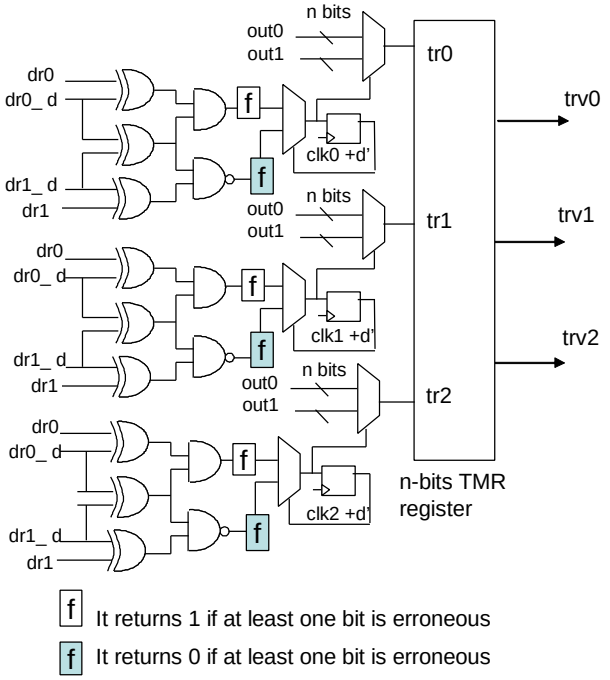


Figure 8. Upset detector and voter circuit area optimization using group of n bits

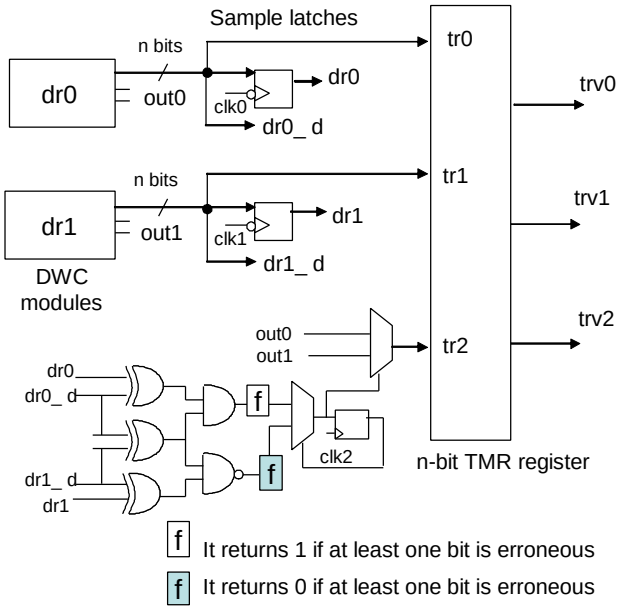


Figure 9. Upset detector and voter circuit area optimization using a single state machine for a group of n bits

In summary, the final DWC with time redundancy scheme is composed of:

- Two redundant blocks of the combinational logic.
- A set of sample latches related to the number of output bits of each redundant block, which is used to capture the value at the clock falling edge.
- Upset detection block, which is continuously monitoring a variation between the captured value and the combinational output during the observation period (clock low level).

The corrected redundant part is voted just before the next clock rising edge, where the TMR redundant part 2 from the register stores the fault-free redundant logic (dr0 or dr1).

4. RESULTS

The DWC with time redundancy scheme was validated by fault injection methodology in a prototype board using VHDL. The fault injection system described in VHDL was specifically developed to test the proposed technique. Results were emulated in an AFX-PQ249-110 board using a XCV300 part. Some area comparisons between the proposed approach and TMR were also performed using Xilinx implementation tools. We use multipliers as combinational circuit case studies, and FIR canonical filters as sequential circuit case studies.

Fault injection in VHDL was used to characterize and validate the technique. The fault injection system is able to randomly choose the fault time, the fault node and the redundant block. The faults are injected in a SRAM cell located in the fault node and its effect is permanent. The fault can be a stuck at one or a stuck at zero. There is a reset fault signal that works as a scrubbing, cleaning up the fault. The circuit chosen for this first evaluation was a 2x2 bit multiplier with a register at the output. It was possible to inject a set of faults in all the user's combinational nodes of the example circuit, covering several time intervals in the clock cycle, and to emulate the scrubbing between faults. The multiplier input vectors were also randomly generated.

Figure 10 shows the simulation of random faults in a node of the multiplexor emulated in the XCV300 FPGA. The fault is a stuck at one and it was inserted in the redundant block 0 during the observation time. There is one point of data acquisition at the clock falling edge, just after the combinational output has stabilized. The fault must be detected before the next clock rising edge (clock+*d*), as shown in figure 6. The fault effect between these two points can be easily detected, and the correct redundant block can be voted. However, upset effects located extremely near the clock rising edge of the register, or during the propagation time cannot be voted, but they can be detected. This limitation on the detection of a fault is due to the impossibility of distinguishing a data disparity coming from a fault or from the input variations in the redundant block 0 and redundant block 1. As the effect of an upset in the user's combinational logic in a FPGA is permanent, all the results from the redundant block 0 after the fault effect are erroneous until the next scrubbing takes place.

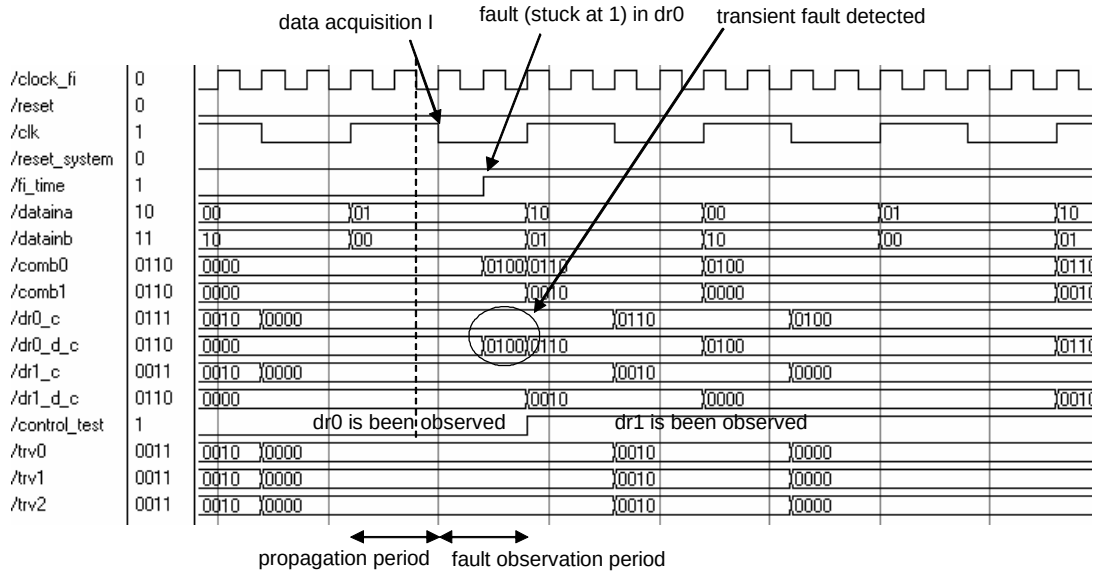


Fig. 10. Simulation Analysis of a fault injection in the DMR with time redundancy scheme implemented in a 2x2 bits multiplier

Table 2. Example of combinational circuit: Multiplier Implemented in XCV300-PQ240 FPGA

	Standard			TMR			DWC with time redundancy		
Multipliers	2x2	8x8	16x16	2x2	8x8	16x16*	2x2	8x8	16x16
Combinational Input Pins	4	16	32	12	48	96	8	32	64
Sequential Input Pins	2	2	2	12	12	12	12	12	12
Output Pins	4	16	32	12	48	96	12	48	96
Number of 4-input LUTs	4	156	705	16	514	2002	33	440	1504
Number of ffs	4	16	32	12	48	96	21	81	161

*I/O pins were out of range for the TMR approach, the part XCV300-BG432 was used.

Fault injection results show the reliability of the presented method. There were inserted 128 stuck at one and 128 stuck at zero faults in a random single node (ranging from 0 to 7) at a random instant of the clock cycle in a 2x2 bit multiplier that could occur during the propagation or the observation time. Among the stuck at one faults, 113 of them were detected and tolerated, either because they were correctly voted or because the fault did not affect the correct design output. Among the stuck at zero faults, 121 of them were also detected and tolerated, either because they were correctly voted or because the fault did not affect the correct design output.

The injected faults during the observation time that generated an error were the ones where the effect could not be observed by the input vectors at that time. Faults occurring during the propagation time were detected and some of them were also tolerated. The tolerated faults are the ones that occurred in the spare redundant block. When the upset effect happens during the propagation time, the scheme presented in figure 6 is not capable of detecting in which redundant block the fault has occurred, only detecting that the system is in error. Consequently, after fault detection with no correction (syndrome 0101), the system should be reinitialized or some results should not be considered.

Table 2 presents area results of 2x2, 8x8 and 16x16 bit multipliers, implemented in the XCV300 FPGA using no tolerance technique, TMR technique and DWC with time redundancy in order to reduce pin count. All of the multipliers were synthesized with a register at the output. Table 2 results show that it is possible not only to reduce the number of I/O pins but also the area, according to the size of the combinational logic block. Note that the 16x16 bit multiplier protected by TMR could not be synthesized in the prototype board that uses a Virtex part with 240 I/O pins (166 available for the user). However, the same multiplier implemented by the proposed technique could fit in the chip, also occupying less area.

There is a constant area in this proposed method, resulting from the upset detection and voter block. Consequently, the proposed approach will only show a smaller area than TMR when the area of the combinational logic related to the third redundant part of the TMR that is suppressed is larger than this constant cost. However, this technique can be used in I/O circuitry, to assure pin count reduction in critical pin count designs.

A canonical filter circuit was chosen as a sequential case study circuit for the proposed technique. Figure 11 shows the scheme of a canonical filter of 5 taps. The multipliers were designed with constant coefficients, resulting in an optimized area. The upset detection and voter block is placed at the outputs, and it votes the correct pad output from dr0 or dr1, as shown in figure 12. The registers are protected by TMR, while the combinational logic (multipliers and adders) is protected by DWC with time redundancy technique.

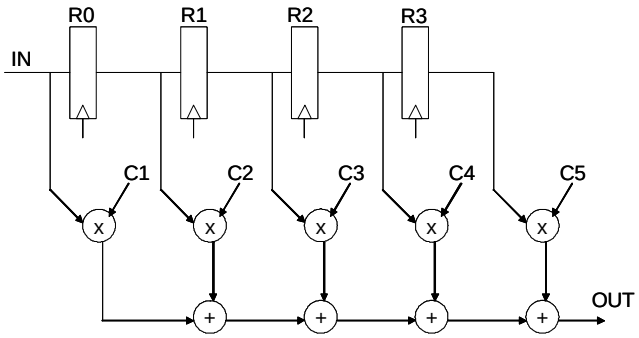


Fig. 11. Example of FIR Canonical Filter of 5 taps scheme

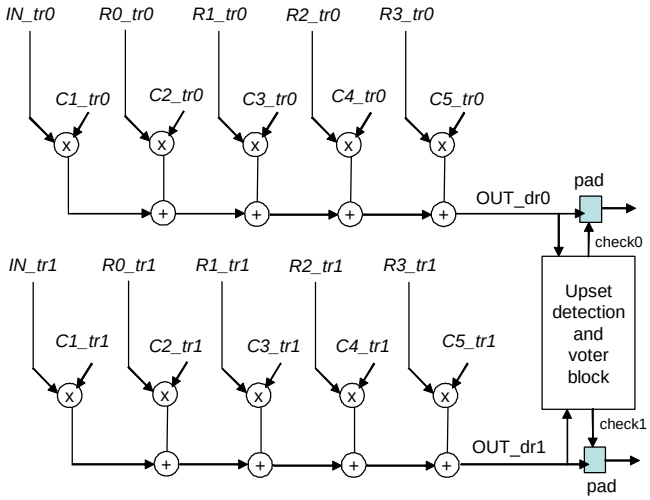


Figure 12. Filter adders and multipliers protected by DWC with time redundancy

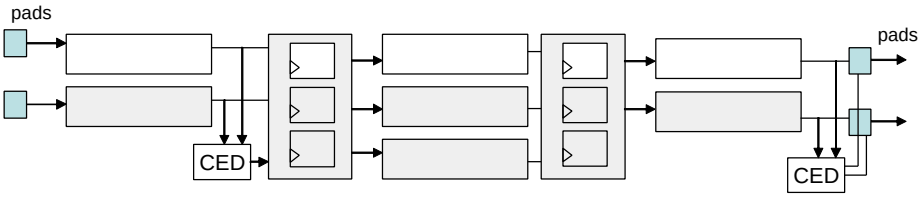
An 8-bit FIR canonical filter of 9 taps was synthesized in a XCV300 FPGA to evaluate area and pin count. The multiplier coefficients are: 2, 6, 17, 32 and 38. Table 3 presents area results of this filter using no tolerance technique, TMR technique and the proposed technique. Results show that the 9 taps FIR canonical filter occupies 22% less area in the FPGA if protected by DWC and time redundancy instead of by TMR. The results also present a reduction of 20% in the pin count compared to TMR.

Table 3. Example of Sequential circuit: FIR canonical filter of 9 taps implemented in XCV300-PQ240 FPGAs

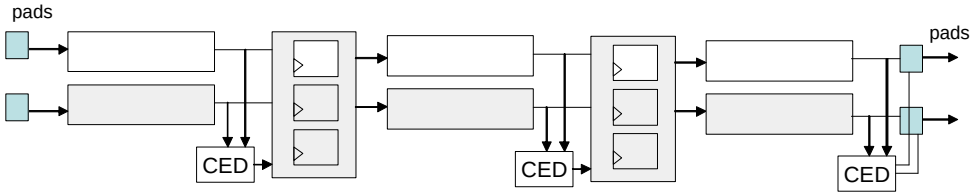
	Standard	TMR	DWC with time redundancy
Combinational Input Pins	8	24	24
Sequential Input Pins	3	15	15
Output Pins	16	48	32
Number of 4-input LUTs	265	948	741
Number of ffs	64	192	225

According to the user's application requirements, the designer will be able to choose between a full hardware redundancy implementation (TMR) or a mixed solution where duplication with comparison is combined to concurrent error detection to reduce pins and power dissipation in the interface, as well as area, as shown in previous examples. Figure 12 shows some implementations combining TMR and DWC with time redundancy. It is possible to use this new technique only in the interface of the FPGA, in this way reducing pins, as shown in figure 12a. DWC with time redundancy can also be used along the design as presented in figure 12b to reduce the number of I/O pads and also area for large combinational circuits, as presented in table 2 and table 3.

Sequential circuits such as counters and state machines are more suitable to be protected by TMR, as the combinational logic is small compared to the sequential logic. The proposed technique is an alternate method to protect combinational circuits, as it is necessary to insert a concurrent error detection block. On the other side, large combinational logic blocks can be easily found in many applications. For example, microprocessors are composed of combinational logic such as the Arithmetic and Logic Unit, multipliers and the micro-instruction decoder.



(a) DMR with time redundancy implementation in the interfaces



(b) DMR with time redundancy implemented in the entire circuit

Figure 12. Evaluation schemes of the TMR and the DWC with time redundancy approach

5. CONCLUSIONS

This work presents a new technique for upset detection and voting that combines duplication with comparison (DWC) with time redundancy for the user's combinational logic in SRAM-based FPGAs. This technique reduces the number of input and output pins of the user's combinational logic when compared to TMR technique. In addition, it can also reduce area, when large combinational blocks are used. The proposed approach was validated by fault injection in a Virtex prototype board using VHDL. Upsets were randomly inserted in the user's combinational logic nodes to emulate faults in the logic. The fault injection procedure was developed in VHDL, and it represents the effect of a SEU in a SRAM-based FPGA, where it has a transient effect followed by a permanent effect. Experiments in a 2x2 bit multiplier showed that 100% of the faults can be detected and 234 of the 256 injected stuck at zero and stuck at one faults (91%) were tolerated, either because they were correctly voted before being captured by a CLB flip-flop or that specific faults did not affect the correct design output.

Although the time redundancy technique can be successfully used to reduce pin count and area overhead over a full hardware redundancy, the transient concurrent error detection technique is not able to correct 100% of the faults occurring in FPGAs. Another penalty of this method is performance overhead because of the observation time. The evolution of this work investigates the use of modified time redundancy technique based on permanent fault detection to improve fault correction and to reduce the performance penalty at each clock cycle.

6. ACKNOWLEDGMENTS

The authors wish to thank Xilinx and CNPq Brazilian Agency for their support of this work and Renato Hentschke from the Federal University of RGS in Brazil (UFRGS) for his contribution to the case study VHDL circuit description.

7. REFERENCES

[1] J. Barth, "Radiation Environment", IEEE NSREC Short Course, July, 1997.

[2] E. Normand, "Single Event Upset at Ground Level", IEEE Transactions on Nuclear Science, VOL. 43, NO. 6, Dec. 1996.

[3] A. Johnston., "Scaling and Technology Issues for Soft Error Rates", 4th Annual Research Conference on Reliability, Stanford University, Oct. 2000.

[4] Xilinx Inc. Virtex™ 2.5 V Field Programmable Gate Arrays, Xilinx Datasheet DS003, v2.4, Oct. 2000.

[5] C. Carmichael, "Triple Module Redundancy Design Techniques for Virtex Series FPGA", Xilinx Application Notes 197, v1.0, Mar. 2001.

[6] F. Lima, C. Carmichael, J. Fabula, R. Padovani, R. Reis, "A Fault Injection Analysis of Virtex® FPGA TMR Design Methodology", Proc. of Radiation and its Effects on Components and Systems (RADECS), Sept. 2001.

[7] C. Carmichael, E. Fuller, J. Fabula, F. Lima, "Proton Testing of SEU Mitigation Methods for the Virtex FPGA", Proc. of Military and Aerospace Applications of Programmable Logic Devices MAPLD, 2001.

[8] M. Caffrey, P. Graham, E. Johnson, M. Wirthlin, "Single Event Upsets in SRAM FPGAs", Proc. of Military and Aerospace Applications of Programmable Logic Devices (MAPLD), Sept. 2002.

[9] L. R. Rocket, "A design based on proven concepts of an SEU-immune CMOS configuration data cell for reprogrammable FPGAs", Microelectronics Journal, VOL. 32, 2001, pp. 99-111.

[10] R. Velazco, D. Bessot, S. Duzellir, R. Ecoffet, R. Koga, "Two Memory Cells Suitable for the Design of SEU-Tolerant VLSI Circuits", IEEE Transactions on Nuclear Science, VOL. 41, NO. 6, Dec. 1994.

[11] W. Wesley Peterson, Error-correcting codes. Ed. 2.ed. Cambridge : The mit Press, 1980. 560 p. ISBN 0262160390.

[12] E. Dupont, M. Nicolaidis, P. Rohr, "Embedded Robustness IPs for Transient-Error-Free ICs", IEEE Design and Test of Computers, May-June, 2002, pp. 56-70.

[13] L. Anghel, M. Nicolaidis, "Cost Reduction and Evaluation of a Temporary Faults Detecting Technique," Proc. 2000 Design Automation and Test in Europe Conference (DATE 00), ACM Press, New York, 2000, pp. 591-598.

[14] L. Anghel, D. Alexandrescu, M. Nicolaidis, "Evaluation of a Soft Error Tolerance Technique based on Time and/or Hardware Redundancy," Proc. of IEEE Integrated Circuits and Systems Design (SBCCI), Sept. 2000, pp. 237-242.

[15] B.W. Johnson, J.H. Aylor, H.H. Hana, "Efficient Use of Time and Hardware Redundancy for Concurrent Error Detection in a 32-bit VLSI Adder," IEEE Journal of Solid-State-Circuits, pp. 208-215, Feb. 1988.