Designing Fault Tolerant Systems into SRAM-based FPGAs

Fernanda Lima[1,2] Luigi Carro[1] Ricardo Reis[1]

[1]Universidade Federal do Rio Grande do Sul
PPGC – Instituto de Informática – DELET
Caixa Postal: 15064, Porto Alegre – RS – Brazil
+55 51 33 16 70 36
{fglima, carro, reis}@inf.ufrgs.br

[2]Universidade Estadual do Rio Grande do Sul
Engenharia de Sistemas Digitais
Estrada Santa Maria 2300, Guaíba – RS – Brazil
+55 51 491 40 42
fernanda-lima@uergs.edu.br

ABSTRACT
This paper discusses high level techniques for designing fault tolerant systems in SRAM-based FPGAs, without modification in the FPGA architecture. Triple Modular Redundancy (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 new technique proposed in this paper 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, low NRE (Non-Recurring Engineering) cost and fast turnaround time. In particular, SRAM-based FPGAs are very valuable for remote missions because of the possibility of being 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, with the aim of reducing the mission cost by correcting errors or improving system performance after launch. SRAM-based FPGA will be the focus of this work, more specifically the Virtex® family [17] from Xilinx.

DAC'03, June 2-6, 2003, Anaheim, California, USA.

Transient faults, also called Single Event Upset (SEU), are the major concern in space applications [3], 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 in order 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 assure reliability [11, 9].

SEU has a peculiar effect in 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 load of the cell. Accordingly, the use of SEU mitigation techniques for programmable architecture must take into account these peculiarities.

In order to mitigate SEU in Virtex® family [17] FPGAs, the Triple Modular Redundancy (TMR) with voting technique combined with bitstream scrubbing has been applied [6]. 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 [10] and radiation ground testing presented in [5] 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 significant increase in power dissipation.

In this work we propose an innovative 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 method combines duplication with comparison (DWC) and concurrent error detection (CED) based on time redundancy to detect permanent faults in the programmable matrix (SEUs in the programmable elements). To the author's knowledge, no references were published about high level concurrent error detection techniques for FPGAs that take into account the permanent effect of a SEU in the user's combinational logic, except for the TMR approach.

This paper is organized as follows. Section II shows some used SEU tolerant techniques applied to FPGAs and to ASICs. Section III introduces the new technique that combines time and hardware 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 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 by different moment of storage (time redundancy), or by a combination of both. An efficient SEU mitigation technique should cope with transient faults occurring in the combinational logic, and SEUs in the storage cells. In this way, transient faults in the combinational logic will never be stored in the storage cells, and bit flips in the storage cells will never occur or will be immediately corrected. Each technique has some advantages and drawbacks, and there is always a compromise between area, performance, power and fault tolerance efficiency.

Time and hardware redundancy techniques are largely used in ASIC [1, 2, 7]. They range from concurrent error detection (CED) to correction mechanisms. The use of full time or full hardware redundancy permits voting the correct value in the presence of single upsets. In the case of time redundancy, the goal is to take advantage of the transient pulse characteristic to compare signals at different moments. The output of the combinational logic is latched at three different moments, where the clock edge of the second latch is shifted by the time delay d and the clock of the third latch is shifted by the time delay 2d. A voter chooses the correct value. The scheme is illustrated in figure 1a. The area overhead comes from the extra sample latches and the performance penalty is measured by twice the duration of the transient upset pulse.

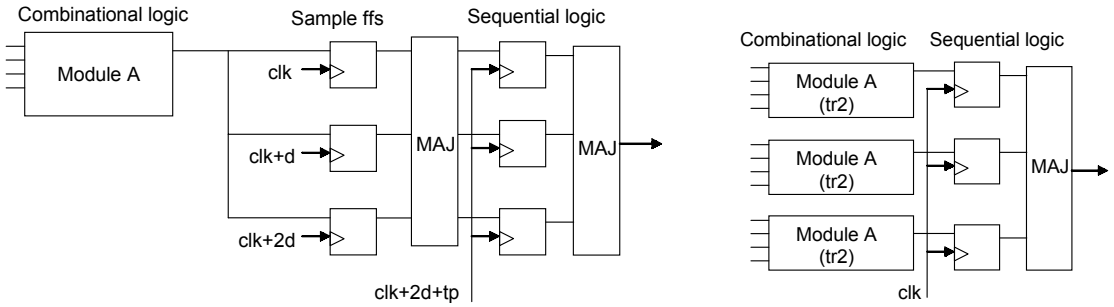
A full hardware redundancy, the well known TMR approach, can also be used to identify the correct value, as shown in figure 1b. Although it presents a larger area overhead compared to time redundancy, since it triplicates all the combinational and sequential logic, it does not have major performance penalties, just the voter propagation time, and it does not need different clock phases.

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 (EDAC), such as Hamming code, or the standard TMR with a single voter, because a fault in the encoder or decoder logic or in the voter would invalidate the technique.

Special techniques should be developed to FPGAs able to cope with this type of effect. The SEU mitigation technique used nowadays to protect designs synthesized in the Virtex® architecture is mostly based on TMR combined with scrubbing [6]. The TMR mitigation scheme uses three identical logic circuits (redundant block 0, redundant block 1 and redundant block 2), synthesized 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 [6], and more examples are also presented in [10, 4]. Figure 2 shows the TMR scheme.

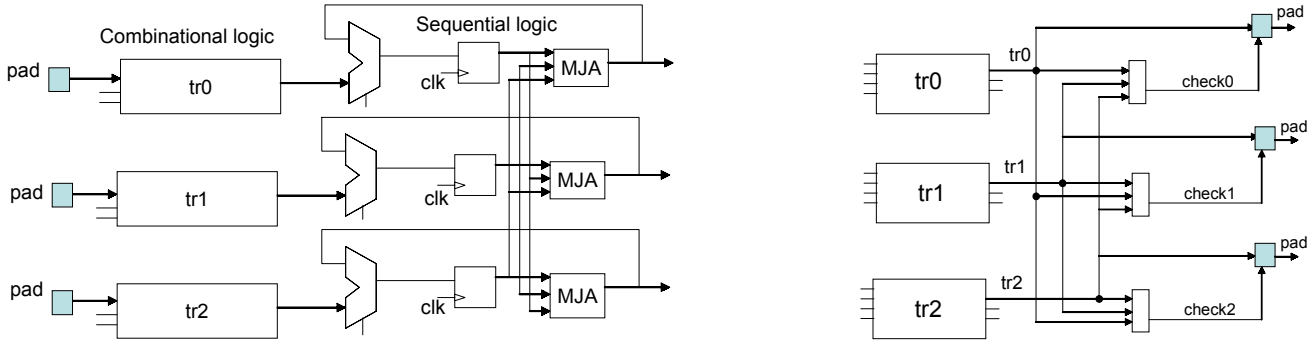
The user's flip-flop is replaced by a TMR structure composed of three flip-flops, three voters and multiplexors that are implemented using LUTs, one for each. The combinational logic and input and output pins are also triplicated to avoid any single point of failure inside the FPGA, as illustrated in figure 2. In this way, any upset inside the matrix can be voted out by the TMR structure assuring correct values in the output.

TMR is an attractive 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 design 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 [14, 16], EDAC techniques and standard TMR with single voter [13], these solutions are costly because they require modifications in the matrix architecture (NRE cost). Techniques to detect permanent faults in FPGAs have been discussed previously in [15]. However, the authors do not discuss the permanent effect of a single upset in the user's combinational logic, but physical faults that cannot be corrected by reconfiguration. When a physical fault occurs in the matrix, it is necessary to reconfigure the design, avoiding the faulty CLBs in order to continue using the part.



(a) Full Time Redundancy (b) Full Hardware Redundancy

Figure 1. Examples of SET and SEU Correction schemes



(a) User's combinational and sequential logic (b) Output Pads

Figure 2. Triple Modular Redundancy for Xilinx FPGAs

In this paper, we present a new high level technique that combines the time and hardware redundancy with some extra features able to cope with the upset effects in FPGAs to reduce the number of I/O pads and area overhead for the user's combinational logic. This method copes with the permanent effect of a SEU in the programmable matrix. In addition, this proposed method is also able to detect physical faults, which are permanent faults that are not corrected by reconfiguration.

3. REDUCING PIN AND AREA OVERHEAD 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 large area, pin and power overheads. Time redundancy can provoke interruption of the system in the presence of faults and performance penalties. In order to explore the design space, one could wonder 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 to protect the user's combinational logic, where the double modular redundancy with comparison (DWC) is combined with a time redundancy upset detection machine, able to detect upsets and to identify the correct value to allow continuous operation. The sequential logic continues to be protected by TMR to avoid accumulation of faults as previously described, since the scrubbing does not change the content of a user's memory cell.

The possibility of applying concurrent error detection based on time redundancy combined with hardware redundancy technique for FPGAs looks interesting for upset detection and upset voting, but there are two problems to be solved. First, the DWC technique 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 without interruption to the operation.

The insertion of a CED block in each redundant module of the duplication approach can help to identify which module is faulty. Figure 3 presents the DWC approach able to perform concurrent error detection (CED), called hot backup DWC-CED. The important characteristic of this approach is that there is always a correct value in the output of the scheme in the presence of a single fault, because the mechanism is able to detect the faulty module and to select the correct output of the two.

There are many methods to implement a CED, basically based on time or hardware redundancy. It is apparent that hardware redundancy is not the appropriate choice for CED, because it would increase the area overhead, and in the end it would be larger than the original TMR. Consequently, time redundancy is the only way to detect a fault without area overhead. The basic concept of time redundancy is the repetition of computation in a way that allows the errors to be detected. The problem of detecting faults in SRAM-based FPGAs is that common time redundancy methods able to detect transient faults are not applicable in this case, because an upset in the programmable matrix has a permanent effect, changing the construction of the user's combinational logic. Consequently, after the occurrence of an upset in the programmable matrix, the upset will only disappear after a reconfiguration that can be

complete or partial. The upset can occur at any time, even the propagation time, and it must be detected.

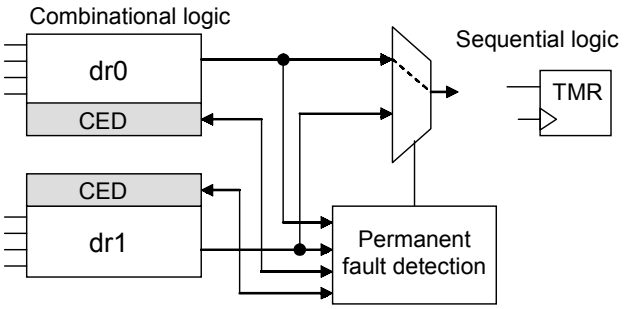


Figure 3. DWC combined with CED for the user's combinational logic

To allow redundancy to detect permanent faults, the repeated computations are performed differently. During the first computation at time t0, the operands are used directly in the combinational block and the result is stored for further comparison. During the second computation at time t0+d, the operands are modified, prior to use, in such a way that errors resulting from permanent faults in the combinational logic are different in the first calculation than in the second and can be detected when results are compared. These modifications are seen as encode and decode processes. The scheme is presented in figure 4.

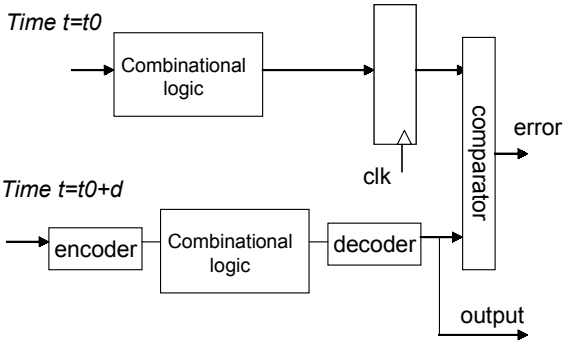


Figure 4. Time redundancy for permanent fault detection

Many techniques to encode and decode were proposed in the literature to detect permanent faults [8], such as bit-wise inversion for stuck at fault problems, parity prediction for logic functions, re-computing with shift operands (RESO) for faulty arithmetic slices and re-computing with swapped operands (REWSO) for faulty chips. This paper will show the first case study, a multiplier. For this circuit, RESO technique can be successfully used, as presented in [12].

The concurrent error detection method RESO uses the principle of time redundancy where the coding function is the left shift operation and the decoding function is the right shift operation. Thus, in the first computation, the operands are computed and stored in a register. At the second computation, the operands are shifted k bits to the left, computed and the result is shifted k bits to the right (2k bits, if a multiplier or divider). In the proposed application, the operands are shifted by 1 bit. The result of the second step is compared to the previous result stored in the register. A mismatch indicates the presence of a permanent fault in the module.

The combination of DWC technique and CED blocks enables one to detect permanent faults provides a new high-level SEU mitigation technique for FPGAs. In the scheme presented in figure 4, two clock cycles are necessary to identify a permanent fault in the combinational logic module. However, this extra time does not occur at every clock operation in our approach. Using DWC combined with CED for permanent faults, it is possible to take advantage of the simple comparison at the output of the duplication scheme to inform whether it is necessary to re-compute the data for permanent fault detection. The re-computation is needed only when a mismatch of the outputs occurs.

If an output mismatch occurs, the output register will hold its original value for one extra clock cycle, while the CED block detects the permanent fault. After this, the output will receive the data from the fault free module until the next reconfiguration (fault correction). The important characteristic of this method is that it does not incur performance penalty when the system is operating free of faults or with a single fault. The method just needs one clock cycle in hold operation to detect the faulty module, and after that it will operate normally again without performance penalties. The final clock period is the original clock period plus the propagation delay of the output comparator, as presented in figure 6. Sample registers (dr0 and dr1) are latched at the rising clock edge and the user's TMR registers are latched at rising clock+d edge.

Figure 5 shows the scheme proposed for an arithmetic module, in the present case study: a multiplier. There are two multiplier modules: mult_dr0 and mult_dr1. There are multiplexors at the output able to provide normal or shifted operands. The computed output computed from the normal operands is always stored in a sample register, one for each module. Each output goes directly to the input of the user's TMR register. Module dr0 connects to register tr0 and module dr1 connects to register tr1. Register tr2 will receive the module that does not have any fault. For default, the circuit starts passing the module dr0. A comparator at the output of register dr0 and dr1 indicates an output mismatch (Hc). If Hc=0, no error is found and the circuit will continue to operate normally. If Hc=1, an error is characterized and the operands need to be re-computed using the RESO method to detect which module has the permanent fault. The detection takes one clock cycle. While the circuit performs the detection, the user's TMR register holds its previous value. When the free faulty module is found, register tr2 receives the output of this module and it will continue to receive this output until the next chip reconfiguration (fault correction).

Let's consider two different fault situations. In one, the fault occurs in module dr0 (Mult_dr0), Hc indicates that there is an output mismatch; Tc0 indicates that module dr0 is faulty and Tc1 indicates that dr1 is fault free. This analysis takes one clock cycle. Consequently, the permanent fault detection block selects dr1 for the tr2 input. Note that the value stored in the user's TMR register is held for one cycle while the scheme identifies the free faulty module. In the second case, a fault occurs in the module dr1 (Mult_dr1), similarly to the previous example, Hc indicates that there is an output mismatch; Tc0 indicates that module dr0 is fault free and Tc1 indicates that dr1 is faulty. The permanent fault detection block selects dr0 for the tr2 input.

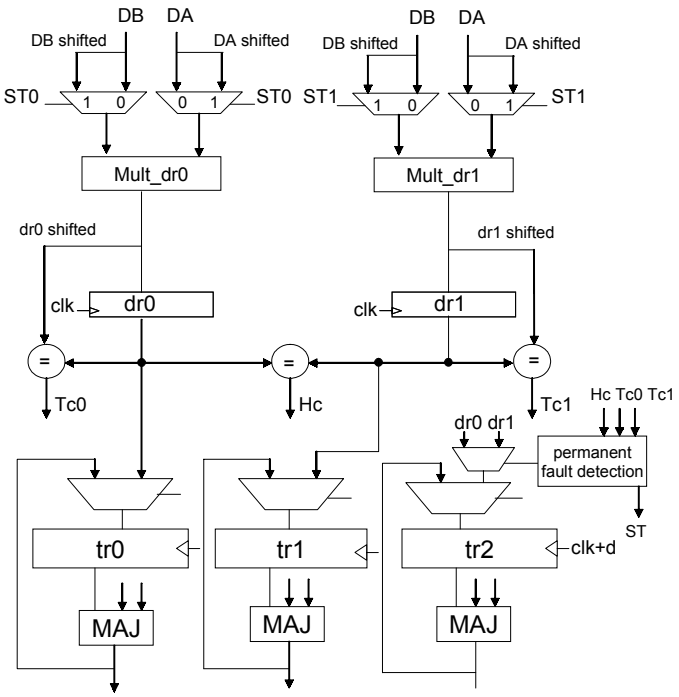


Figure 5. DWC-CED proposed scheme

Note that in both methods, TMR and the proposed technique, the upsets in the user's combinational logic are corrected by scrubbing, while upsets in the user's sequential logic are corrected by the TMR scheme used in the CLB flip-flops. It is important to notice that for upset correction the scrubbing is performed continuously, to assure that only one upset has occurred between two reconfigurations in the design. Some constraints must be observed for the perfect functioning of the technique, same as TMR: there must not be upsets in more than one redundant module, 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 module 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 in 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 far as there is no upset in both redundant blocks. Upsets in the routing are also considered in this approach as far as there is no fault in the routing that can connect two signals from dr0 and dr1 modules. In order to assure that faults in the routing will not short cut signals from dr0 and dr1, it is necessary to use a dedicated placement for each module.

In the proposed method, the area reduced by the design compared to the TMR is the area of one user's combinational logic module and the number of inputs that is reduced from 3 times to 2 times the original number. 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 clocks, reset, etc.

In addition, the advantage of using this technique is not only focused in reducing the pin count and the number of CLBs, but also in other type of radiation effects such as total ionization dose as this method has the important characteristic of detecting permanent faults. So far, we have mentioned only SEUs that happen in the SRAM programmable cells that are permanent until the next reconfiguration. However, a circuit operating in the space environment can suffer from total ionization dose and other effects that can provoke permanent physical damages in the circuit.

4. RESULTS

The proposed DWC-CED technique for permanent fault detection was first validated by fault injection methodology in a prototype board using emulation. 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 case studies.

Fault injection in VHDL was used to characterize and validate the technique. The fault injection system is able to randomly choose the instant of insertion of the fault, the faulty node and the redundant module (mult_dr0 or mult_dr1). There is a reset fault signal that works as a scrubbing cleaning up the fault. The circuit chosen for this first evaluation was an 8x8 bits multiplier with a register in the output. In this way it was possible to inject a set of faults in each user's combinational nodes of the example circuit (193 nodes in total) covering all time span of the clock cycle and to emulate the scrubbing between faults. The multiplier input vectors were also randomly generated. The faults are injected in a SRAM cell located in the fault node and its effect is permanent until the chip reconfiguration (scrubbing). Fault injection results show the reliability of the presented method.

The fault coverage was exhaustively measured by exercising all possible combination of vectors and faults inside the multipliers. For an 8-bit multiplier, there are 2^{16} (combination of vectors) x 193 (nodes) x 2 (type of faults, stuck at 0 and stuck at 1) combinations of faults that where emulated in prototype board. The data has been analyzed by Chipscope Software [18]. Final results show that the implemented RESO method can detect and corrected on the fly 99.97% of the faults within one clock cycle delay before being captured by the register.

Table 1 presents area results of 8x8 and 16x16 bits multipliers, implemented in the XCV300 FPGA using no fault tolerance technique, TMR technique and the proposed technique (DWC-CED for permanent faults). There are two versions of multipliers: one is synthesized with a register at the output and the other is pure combinational. Results show that according to the size of the combinational logic block, it is possible to not only to reduce the number of I/O pins but also area. Note that the 16x16 bits multiplier protected by TMR could not be synthesized in the prototype board that uses a Virtex part with 240 I/O pins (196 available for the user); while the same multiplier, implemented by

the proposed technique could fit in the chip, and also occupying less area.

Table 1. Evaluation of pin count and area of the DWC-CED technique implemented in a XCV300-PQ240 part.

	No protection		TMR		DWC-CED	
Multipliers (reg)	8x8	16x16	8x8	16x16	8x8	16x16
Total of I/O pads	34	66	108	204*	92	172
Number of 4-LUTs	159	741	584	2285	534	1791
Number of ffs	16	32	48	96	82	162
Multipliers (no reg)	8x8	16x16	8x8	16x16	8x8	16x16
Total of I/O pads	32	64	96	192	66	130
Number of 4-LUTs	156	711	551	2159	425	1442
Number of ffs	0	0	0	0	34	66

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

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 time redundancy is combined with hardware redundancy to reduce pins and power dissipation in the interface. It is possible to use DMR and time redundancy only in the interface of the FPGA, in this way reducing pins. DMR and time redundancy can also be used in the design to reduce not only number of I/O pads, but also area for large combinational circuits as presented in table 1.

5. CONCLUSIONS

This work presented a new technique for upset detection and voting that combines duplication with comparison (DWC) with concurrent error detection (CED) based on 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. In addition, it can also reduce area when large combinational blocks are used. The proposed approach was validated by fault injection experiment in a Virtex prototype board using emulation. A large number of 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 8x8 bits multiplier has shown that 100% of the faults can be detected and all of them can be voted on the fly before being captured by a CLB flip-flop. Ongoing work includes performing additional fault injection evaluation in more complex circuits such as larger multipliers, filters and to study the applicability of this method for other cores such as microprocessors.

6. REFERENCES

[1] Anghel, A., Alexandrescu, D., Nicolaidis, M., "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.

[2] Anghel, L., Nicolaidis, M., "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.

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

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

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

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

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

[8] Johnson, B.W., Aylor, J. H., Hana, H., "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.

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

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

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

[12] Patel, J., Fung, L., "Multiplier and Divider Arrays with Concurrent Error Detection", Proceedings of FTCS-25, Vol. 3, 1996.

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

[14] Rocket, L. R., "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.

[15] Shu-Yi, Y., McCluskey, E., "Permanent Fault Repair for FPGAs with Limited Redundant Area", IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2001.

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

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

[18] XILINX, Inc. Chipscope Software and ILA Cores User Manual, Xilinx User Manual, 0401884 (v2.0) Dec., 2000