Improving Fault Tolerance to Radiation Effects in Integrated Systems

Gustavo Neuberger[1], Fernanda Kastensmidt[2], Ricardo Reis[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

[2]Universidade Estadual do Rio Grande do Sul (UERGS)
Estrada Santa Maria 2100 –Guaiba – RS – Brazil

{neuberg,reis}@inf.ufrgs.br, fernanda-lima@uergs.edu.br

Abstract. *This paper describes the radiation effects in integrated systems and discusses some techniques to mitigate these effects. The main circuits analyzed are SRAM memories and SRAM-based FPGAs. New techniques used to protect these circuits against these effects are also proposed in this work. One of the presented techniques to protect FPGAs is based on a combination of Double Modular Redundancy (DMR) with Concurrent Error Detection (CED) that can reduce overheads comparing to Triple Modular Redundancy (TMR). In the case of SRAM memories, a technique based on the Reed-Solomon Code and Hamming Code was developed, as well as a tool to generate a core for fault-tolerance that minimizes the area cost of the code. Some results are presented.*

1. Introduction

Fault-tolerance on semiconductor devices has been a meaningful matter since upsets were first experienced in space applications several years ago. Since then, the interest in studying fault-tolerant techniques to keep integrated circuits (ICs) operational in such hostile environment has increased, driven by all possible applications of radiation tolerant circuits, such as space missions, satellites, high-energy physics experiments and others [NASA 2002]. Spacecraft systems include a large variety of analog and digital components that are potentially sensitive to radiation and must be protected or at least qualified for space operation.

This paper reviews some techniques currently used to protect integrated circuits against radiation effects, and introduces new techniques developed to increase fault tolerance and/or reduce the cost involved to protection of the circuits. These techniques target fault tolerance in SRAM-based FPGAs and SRAM memories. The technique for FPGA combines time and hardware redundancy to reduce overhead comparing to the traditional Triple Modular Redundancy (TMR). It is based on duplication with comparison and concurrent error detection. The new technique proposed was specifically developed to cope with soft errors in the user combinational and sequential logic, while also reducing pin count, area and power dissipation. This technique is implemented in the high description level without modification in the FPGA architecture.

Concerning about memories, the techniques used are based on Error Detection and Correction Codes (EDAC). A popular one is Hamming code [Azumi 1975] that is a

preferred solution for implementation at circuit level due to area and performance reasons. However, it has limited error correction capability. The use of a different code, the well know Reed-Solomon code [Houghton 1997], is proposed in a hardware core. There are many different configurations of this code, such as the primitive polynomial, the number of bits to be protected, the size of the internal blocks and others. Because of this complexity, a tool able to automatically search for the best solution for a given set of parameters was developed. This tool aims to minimize the cost of the hardware implementation of this code. It generated a high level description of the code. Results show that it can be a very good alternative to Hamming code, with an efficient hardware implementation. The final implementation of Reed-Solomon code and Hamming code in a case study memory is presented and fault injection experiments have confirmed the reliability of the methods.

This paper is organized as follows. Section 2 describes the radiation effects on integrated circuits manufactured using CMOS process and specifically the effects in SRAM memories and in SRAM-based FPGA architectures. Section 3 discusses some SEU mitigation techniques that can be applied to protect circuits in general and that has been used in the past. Section 4 introduces a new high-level technique for designing fault-tolerant systems for SRAM-based FPGAs, without modifications in the FPGA architecture, able to cope with transient faults in the user combinational and sequential logic, while also reducing pin count, area and power dissipation compared to the traditional TMR. The section 5 shows a tool developed to create optimized hardware implementations of the Reed-Solomon code, an attractive alternative to the well-known Hamming code, but with more error correction capability. Section 6 presents a fault tolerant memory designed to be able to cope with all double bit upsets and a large amount of multiple bit upsets, combining Reed-Solomon and Hamming codes. The main conclusions are discussed in section 7, followed by the references.

2. Radiation Effects in Digital Circuits

The digital circuits located in the space environment are affected by the charged particles generated by the solar flares. When a charged particle hits the silicon, it can provoke a transient pulse, which can be interpreted as an internal signal [Bessot 1993]. This transient pulse can change the state of a memory cell. This phenomenon is called soft error or Single Event Upset (SEU). The consequences of SEUs are entirely device specific, and depend on the impact of the corrupted information in the system. A charged particle can also hit the combinational logic. The generated current pulse may be propagated by the logic and if latched by memory elements, it provokes a soft error (SEU) as well.

New generation of integrated circuits are becoming more sensitive to radiation effects. High-density devices require smaller feature size, this means less capacitance and hence information is stored with less charge. Lower voltage or lower power devices means that less charge or current is required to store information. Each of these effects makes the device more vulnerable to radiation and means that particles with little charge, which were once negligible, are now much more likely to produce upset or damage. As higher is the performance of a circuit more sensitive to radiation environment it is.

2.1. Radiation Effects in SRAM Memories

The most common circuit sensitive to SEU is the memory element. The memory cell is designed so that it has two stable states, one that represents a stored '0' and one that represents a stored '1'. In each state, two transistors are turned on and two are turned off (SEU target drains). A bit-flip in the memory element occurs when an energetic particle causes the state of the transistors in the circuit to reverse. This phenomenon occurs in many microelectronic circuits including memory chips and microprocessors. In a computer operating in space, for example, a bit-flip could randomly change critical data, randomly change the program, or confuse the processor to the point that it crashes.

Although SEU is the major concern in space applications, multiple bit upsets (MBU) start to be also a matter to be addressed in very deep submicron (VDSM) technologies. When a single high-energy ion passes through the silicon it can energize two or more adjacent memory cells [Reed 1997]. MBUs can be induced by direct ionization or nuclear recoil. The energy of the particle is more likely to provoke double bit upsets while multiple bit upsets are caused by an increase of the particle incident angle. Experiments in memories under proton and heavy ions fluxes have shown the probability of multiple upsets provoked by a single ion [Wrobel 2001], [Johansson 1999], [Buchner 2000].

2.2. Radiation Effects in FPGAs

In FPGAs, the upset has a peculiar effect when hit the combinational and sequential logic mapped into the programmable architecture. Let’s take for example the SRAM-based FPGAs such as the Virtex® family from Xilinx that is one of the most popular programmable devices used in the market nowadays.

Virtex devices consist of a flexible and regular architecture composed of an array of configurable logic blocks (CLBs) surrounded by programmable input/output blocks (IOBs), all interconnected by a hierarchy of fast and versatile routing resources. The CLBs provide the functional elements for constructing logic while the IOBs 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 channel. The Virtex matrix also has dedicated memory blocks called Block Select RAMs 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 programmed using a bitstream, which contains all the information to configure the programmable storage elements in the matrix located in the Look-up Tables (LUT) and flip-flops, CLBs configuration cells and interconnections (figure 1). All these configuration bits are potentially sensitive to SEU and consequently they were our investigation targets.

In SRAM-based FPGA, both the user’s combinational and sequential logic are implemented by customizable logic memory cells, in other words, SRAM cells, as represented in figure 1. When an upset occurs in the combinational logic synthesized in the FPGA, it corresponds to a bit flip in one of the LUTs cells or in the cells that control the routing. An upset in the LUT memory cell modifies the implemented combinational logic. It has a permanent effect and it can only be corrected at the next load of the

configuration bitstream. The effect of this upset is related to a stuck-at fault (one or zero) in the combinational logic defined by that LUT. An upset in the routing can connect or disconnect a wire in the matrix. It has also a permanent effect and its effect can be mapped to an open or a short circuit in the combinational logic implemented by the FPGA.

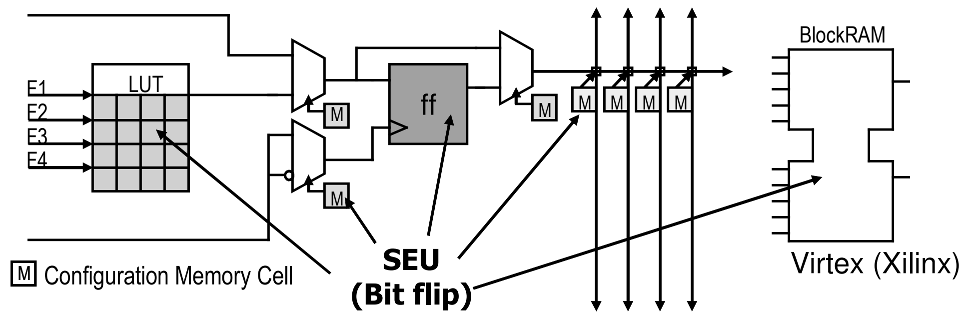

Figure 1. SEU Sensitive Bits in the CLB Tile Schematic

Radiation tests performed in Xilinx FPGAs [Fuller 2002][Carmichael 2001] show the effects of SEU in the design application and prove the necessity of using fault-tolerant techniques for space applications. A fault-tolerant system designed into SRAM-based FPGAs must be able to cope with the peculiarities mentioned in this section such as transient and permanent effects of a SEU in the combinational logic, short and open circuit in the design connections and bit flips in the flip-flops and memory cells.

3. SEU Mitigation Techniques

A SEU immune circuit may be fulfilled through a variety of mitigation techniques, including hardware, software, and device tolerance solutions. The most cost efficient approach may be an appropriate combination of SEU-hard devices and other mitigation solutions. However, the availability, power, volume, and performance of radiation-hardened devices may difficult their use. Hardware or software design also serves as effective mitigation, but design complexity may be a problem. A combination of the two may be a good select option.

Solutions to turn a logic device SEU tolerant can be implemented at different steps of the device development process. Possible solutions to design fault tolerant circuits are:

- Change of the technology process;
- Substitution of memory cells by modified ones;
- Use of Error Detection and Correction Codes (EDAC)
- Use of Triple Modular Redundancy (TMR)

3.1. Hardening by Technology

The SEU mitigation technique by technology consists in the use of a specific technology process to turn the entire device immune to radiation particles such as Silicon on Insulator (SOI) CMOS process [IBM 2000]. In this case the charged particle has much less chance to affect the device.

3.2. Hardened Memory Cells

The main idea is to provide CMOS memory cells with an appropriate feedback devoted to restore the data when it is corrupted by an ion hit. The principle is to store the data in two different locations within the cell in such way that the corrupted part can be restored. The main problem is how to organize the extra transistors used to realize the feedback that will result in new sensitive nodes, without affecting the SEU sensitivity.

The main advantages of this method are high performance (read/write time), low sensibility to temperature, technology process independence and voltage supply good SEU immunity. The main drawback is silicon area overhead. Many memory cells based on this approach have been developed in the last years, like IBM Memory Cell [Rockett 1992], NASA Memory Cell [Whitaker 1991] and HIT Memory Cell [Bessot 1993].

3.3. Hardening by EDAC

The Error Detection and Correction (EDAC) solutions [Label 1999] are examples of solutions that can be used to detect or/and correct SEUs when they occur. Some of them can achieve an acceptable level of reliability. An example is the Hamming Code technique. This approach can be used either in the circuit design or in the system level. Using Hamming code as a SEU mitigation solution in the design of a circuit, extra logic blocks are needed to code and decode the stored values such as registers and internal memory.

The use of EDAC is especially appropriated to protect high amounts of data, like in registers and memories. Some extra bits are necessary to store the redundancy bits, but the number of them is smaller than the number of data bits. But an extra circuit to encode and decode the data is needed, and it can also affect the performance of the system. Each code that can be used has a different level of protection, according to the model of faults used (single, multiple).

3.4. Hardening by TMR

Redundancy between modules, circuits or systems provides a potential mean of recovery from a SEU on a system. Autonomous or ground controlled switching from a prime system to a redundant spare may be an option, depending on spacecraft power and weight restrictions. With three identical circuits, the voter can choose the output that at least two agree upon. This approach is called Triple Modular Redundancy (TMR). The main drawbacks of this technique are the high area needed to triplicate the circuit, and that the voter must be designed in such way that no error can occur in the voter. In this case, the voter is designed using some previous techniques in the circuit or design level.

4. Protecting SRAM-based FPGAs against Soft Errors

The Triple Modular Redundancy (TMR) technique 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. Many applications can accept the limitations (penalties) of the TMR approach but some cannot. 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, a new technique based on time and hardware redundancy to protect the user's combinational logic is presented, 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 reliability and the safety of TMR scheme compared to self-checking-based fault-tolerant schemes were discussed in [Lubaszewski 1998]. The experimental results presented that the higher the complexity of the module, the greater the difference in reliability between self-checking and TMR. In summary, the self-checking fault-tolerant scheme can achieve a higher reliability in comparison to the TMR if the self-checking overhead bound of 73% is not exceeded. The idea of using self-checking fault-tolerant scheme can be extended for FPGAs by using the duplication with comparison (DWC) method combined with concurrent error detection (CED) technique that it works as a self-checking. Figure 2 presents the scheme, called hot backup DWC-CED. The CED is able to detect which module is faulty in the presence of an upset, and consequently, there is always a correct value in the output of the scheme, because the mechanism is able to select the correct output out of two.

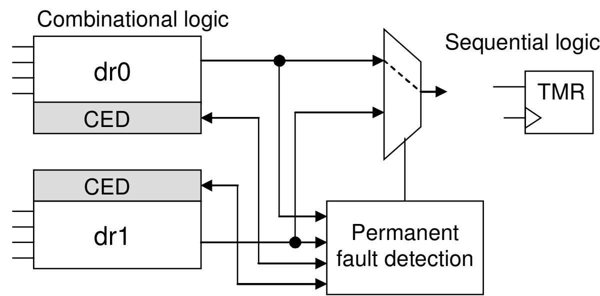

Figure 2. DWC combined with CED scheme

In the case of SEU detection in SRAM-based FPGAs, the CED must be able to identify permanent faults in the redundant modules. The CED works by finding the property of the analyzed logic block that can help to identify an error in the output in the presence of a permanent fault. There are many methods to implement logic to detect permanent faults, most solutions are based on time or hardware redundancy and they manifest a property of the logic block that is being analyzed. The CED scheme based on time redundancy recomputes the input operands in two different ways to detect permanent faults. 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 and they depend on the characteristics of the logic block.

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 a high 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 encoders, decoders and output comparator.

This method has been named duplication with comparison combined to concurrent error detection block (DWC-CED) [Lima 2003]. The scheme is shown in figure 3. Module dr0 connects to register tr0 and module dr1 connects to register tr1. 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). By default, the circuit starts passing the module dr0. A comparator at the output of 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 by the encoder and decoder method to detect which module has the permanent fault. The detection takes one clock cycle. The encode and decode blocks implement the technique used to identify permanent faults in logic: re-computing with shifted operands (RESO) [Patel 1982].

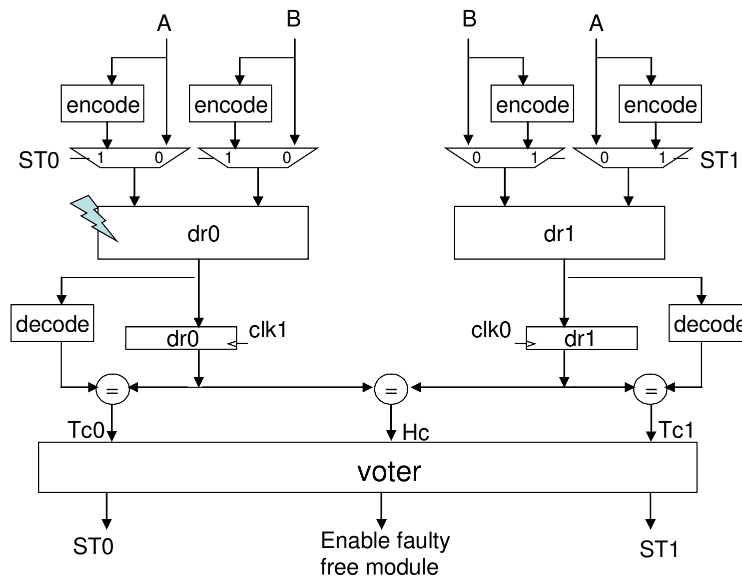

Figure 3. Fault tolerant technique based on DWC-CED for SRAM-based FPGAs

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 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.

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). Results show that according to the size of the combinational logic block, it is possible to not only 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 (166 available for the user); while the same multiplier, implemented by the proposed technique could fit in the chip, and also occupy less area. In terms of performance, the TMR approach has presented an estimated frequency of 33.8 MHz, while the DMR-CED approach has presented a frequency of 26.7 MHz.

Table 1. Comparison of multiplier implementations (XCV300-PQ240)

Multipliers	Standard		TMR		DWC-CED	
Non-registered output	8x8	16x16	8x8	16x16	8x8	16x16
Total of I/O pads	32	64	96	192	66 (-31%)	130 (-32%)
Number of 4-LUTs	156	711	551	2159	425 (-23%)	1442 (-33%)
Number of ffs	0	0	0	0	34	66

5. Reed-Solomon Core Automatic Generator Tool

Error detection and correction code (EDAC) is a well-known technique to protect storage devices against transient faults because it can be implemented in a high-level design step, without changes in the mask process (low NRE cost). There are examples of SEU mitigation techniques using EDAC performed by software [Shirvani 2000], and by hardware [Redinbo 1993]. An example of EDAC is the Hamming code that is largely used to protect memories against SEU because of its efficient ability to correct single upsets with reduced area and performance overhead [Hentschke 2002]. However, Hamming code is not able to ensure reliability in Very Deep Submicron (VDSM) technology in presence of multiple bit upsets caused by the environment.

In the other hand, Reed-Solomon [Houghton 1997] is a block-based error correcting code, able to cope with multiple upsets. It has a wide range of applications in digital communications and storage. Reed-Solomon (RS) codes are used to correct errors in many systems including: storage devices, wireless or mobile communications, high-speed modems and others. RS encoding and decoding is commonly carried out in software. The RS code is based in the Finite Field Arithmetic. The main concepts of the arithmetic and the basic algorithm of the RS code can be found in [Neuberger 2004]. The code needs a polynomial to be created.

The first inconvenience of RS code in hardware implementation is the necessity of large tables of constants that usually are implemented in memory arrays. This structure presents a large area overhead for the code implementation and it cannot be easily integrated in a single chip because of its customization technology. A description of the RS code was developed based on multipliers that have completely replaced the table of constants [Neuberger 2003]. This solution has shown a good compromise in area, performance and reliability. However, for some code parameters, the high number of multipliers and the constants chosen can still present a non-optimal cost.

The main area cost of the RS code implemented in hardware lies on the size of the data block to be encoded, the generator polynomial used to create the multipliers and the size of the symbol, which implicates the set of constants to be used.

The main part of the Reed-Solomon encoding and decoding circuits are multiplications by several constants using a multiplier. To decrease the overhead in the overall circuit, the bigger effort can be spent to optimize the area used by this multiplier. The possible optimizations that can be done in the multiplier are:

- choice of the generator polynomial that produces the best multipliers,
- choice of the most appropriated constants for the multipliers.

To automatically develop an optimized RS code for the given characteristics, using the described techniques of optimization, a tool was developed. Based on the user's input, which represents a set of parameters, it creates the multipliers needed, calculates the cost associated with each one, and creates a VHDL description of the best possible hardware implementation of the code.

After the implementation of the automatic generator, some comparisons were made with previously manually implemented cores, to evaluate efficiency. The results shown in this section are based in the synthesis of the generated VHDL code in a Xilinx FPGA VirtexE V600EHQ240. The table 2 shows a comparison between codes with 7-bit symbols and 112-bit words. The first version was presented in [Neuberger 2003], and was designed manually, without the optimization techniques showed. The second version was manually generated too, but the constants to be used were carefully chosen, trying to reduce area overhead choosing constants with more zeros. These two versions were designed using the same polynomial (137). The third version was created using the automatic approach with the same polynomial. The constants chosen automatically were not the same as the ones used in the second version. The fourth version was automatically generated, but without the restriction of the polynomial to be used. The polynomial chosen this time was 131.

Table 2. Comparison between codes generated manually and automatically

	LATW 2003		Manually Optimized		Automatic 137		Automatic Free Poly	
	Enc	Dec	Enc	Dec	Enc	Dec	Enc	Dec
# 4-LUTs	215	538	140	402	134	392	129	370
Delay (ns)	14.5	47.6	13.9	30.5	13.9	30.5	13.8	30.4

The results show that a high reduction in area has occurred from the first to the second version, but the generator can achieve a more significant area reduction, and the fourth version, using the polynomial 131 had the best result from all of them. This means that the techniques presented can produce a better area result than manually implemented versions could produce, besides reducing the design time for future implementation versions.

6. Protecting Memories against Multiple Bit Upset (MBU)

An n-bit data memory can be protected against faults by using correcting code techniques based on encoder and decoder blocks and extra bits to store the data parities,

as represented in figure 4 [Neuberger 2003]. The encoder and the decoder can use any error detection and correction code. For example, in the case of using Hamming code, the memory can only support single bit upsets. The problem is that multiple upsets are likely to occur in current and in future technologies. In addition, upsets can accumulate in the memory if the correction process is not fast enough comparing to the upset flux.

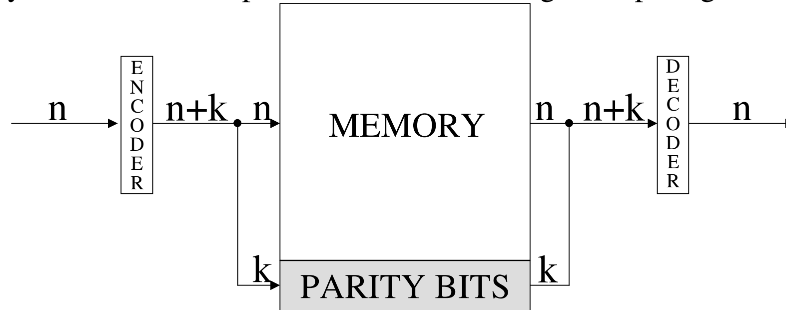

Figure 4. General schematic of a fault tolerant memory

To be able to correct multiple bit upsets, RS code must be used. The data word is divided in symbols, and each data word is a different RS coded word. For example, in a 256-rows memory, the data word uses the entire row, and each data word is divided in m symbols according to the symbol size and to the memory data size. Multiple upsets may occur in any portion of the matrix, but they are more likely to occur as double bit flips that are in the same symbol, in vertical adjacent symbols, or in horizontal adjacent symbols. The RS code can easily correct the first two types of errors, but the third one will not be corrected, because it is equivalent to errors in two different RS symbols in the same word, and the implemented RS is not capable to correct single errors in different symbols (blocks), only multiple errors in the same symbol.

A new code is needed to correct all possible double errors. The first option is the use of a Reed-Solomon code with capability to correct two different symbols. But this RS code has more than twice the area and delay overhead of the single symbol correction RS [Houghton 1997], which makes this solution inappropriate for memory architectures. The second alternative is the use of one bit protected by Hamming code between the RS symbols. The number of bits protected by Hamming will be the same of the number of symbols protected by Reed-Solomon, so this option does not significantly increases the area overhead. Figure 5 presents the insertion of Hamming code in row already coded by RS code.

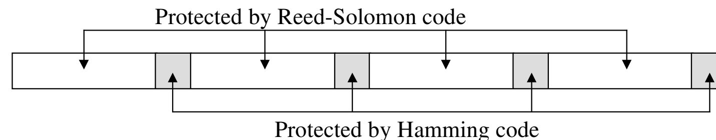

Figure 5. Schematic of a memory row protected by Reed-Solomon and Hamming

All single upsets are corrected by the code. Double bit upsets occurring horizontally or vertically are corrected by the code because each row is a different coded word and there is Hamming code protection in the RS symbols interface. Some multiple bit upsets can also be corrected. The only type of multiple upsets that can not be corrected by the method is when the hamming bit and two adjacent RS symbols are inverted. However, it is unlikely to occur in current technologies.

Once defined the codes to be used, the next step is to determine the specifications of the memory and the Reed-Solomon and Hamming codes sizes. The first memory study case is a 128-bit data memory, where a 128-bit code is needed. This number has been chosen based on the previously compared tradeoffs of RS code [Neuberger 2002]. A 7-bit symbol RS code was used to protect 112 bits of the data memory and a 16-bit Hamming code was used to protect the bits between each RS symbol. Although most memories do not have a word as large as 128 bits, it still has special interest in cache memories that are becoming larger in actual microprocessors.

The memory study case was described in VHDL and synthesized in a Virtex-E FPGA using embedded memories (BlockRAMs) and programmable complex logic blocks (CLBs). Results are presented in table 3. In the results, it is noticed that the fault tolerant memory has an area overhead that is basically the area used by the encoder and decoder blocks. Only two more BlockRAMs are needed, one to store the RS redundancy symbols and other to store the Hamming extra bits. The performance penalty in the fault tolerant memory synthesized in the FPGA is around 50%.

Table 3. Area and performance comparison between a no-protected 128-bit memory and a fault tolerant 128-bit memory based on RS and Hamming code.

	No protected Memory	Fault-tolerant Memory
# 4-LUTs	95	770
# BlockRAMs	16	18
Speed (MHz)	71	30

7. Conclusions

This work presented new techniques to increase fault tolerance to radiation in integrated circuits. The first technique present was 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. Also, a tool that generates efficient hardware implementations of the Reed-Solomon code was presented, and results show that it is a good alternative to the Hamming code. Finally, a SRAM memory that combines Reed-Solomon and Hamming code was presented, making the memory capable to tolerate all double bit upsets, that are likely to occur nowadays in latest technologies.

References

Azumi, S. and Kasami, T. (1975) "Of Optimal Modified Hamming Codes", In: Trans. Inst. Electr. Commun. Eng. Jap., vol. A58, no. 6, pp. 325-330.

Bessot, D. (1993) "Conception de Deux Points Memoire Statiques CMOS Durcis Contre L'effet des Aleas Logiques Provoques par L'environment Radiatif Spatial", These. INPG. November, 1993.

Buchner, S., Campbell, A., Meehan, T., Clark, K., McMorrow, D., Dyer, C., Sanderson, C., Comber, C. and Kuboyama, S. (2000) "Investigation of Single-Ion Multiple-Bit Upsets in Memories on Board a Space Experiment", In: Proceedings of IEEE Transactions on Nuclear Science, June 2000.

Carmichael, C., Fuller, E., Fabula, J. and Lima, F. (2001) "Proton Testing of SEU Mitigation Methods for the Virtex® FPGA", In: International Conference on Military and Aerospace Applications of Programmable Logic Devices, MAPLD, 2001.

Fuller, E. (2002) "Radiation test results of the Virtex FPGA and ZBT SRAM for Space Based Reconfigurable Computing", In: International Conference on Military and Aerospace Applications of Programmable Logic Devices, MAPLD, 2002.

Hentschke, R., Marques, F., Lima, F., Carro, L., Susin, A. and Reis, R. (2002) "Analysing Area and Performance Penalty of Protecting Different Digital Modules with Hamming Code and Triple Modular Redundancy", In: Symposium on Integrated Circuits and Systems Design (SBCCI), September.

Houghton, A. D. (1997) "The Engineer's Error Coding Handbook", London, Chapman & Hall.

IBM.(2000) "SOI Technology: IBM's Next Advance in Chip Design", In: http://www.ibm.com, January.

Johansson, K., Ohlsson, M., Olsson, N., Blomgren, J., and Renberg, P. (1999) "Neutron Induced Single-Word Multiple-bit Upset in SRAM", In: IEEE Transactions on Nuclear Science, December 1999.

Label, K. (1999) "Commercial Microelectronics Technologies for Applications in the Satellite Radiation Environment", In: http://flick.gsfc.nasa.gov/radhome.htm.

Lima, F., Carro, L. and Reis, R. (2003) "Techniques for reconfigurable logic applications: Designing fault tolerant systems into SRAM-based FPGAs". In: Proceedings of the 40th conference on Design automation, June.

Lubaszewski, M. and Courtois, B. (1998) "A reliable fail-safe system", In: IEEE Transactions on Computers, New York, v.47, n.2, p. 236-241, February.

NASA. (2002) "Radiation Effects on Digital Systems", In: http://radhome.gsfc.nasa.gov/top.htm, January.

Neuberger, G., Lima, F. and Reis, R. (2002) "Designing a Reed-Solomon Core Optimized for Area and Performance", In: Proceedings of XVII South Symposium on Microelectronics, June.

Neuberger, G., Lima, F., Carro, L. and Reis, R. (2003) "A Multiple Bit Upset Tolerant SRAM Memory", In: Latin-American Test Workshop, February.

Neuberger, G., Kastensmidt, F. and Reis, R. (2004) "Improving the Use of Reed-Solomon Codes to Increase Fault-tolerance in Very Deep Sub-Micron Integrated Circuits", In: Latin-American Test Workshop, March.

Patel, J. H., Fung, L. Y. (1982) "Concurrent Error Detection in ALUs by Recomputing with Shifted Operands", IEEE Transactions on Computers, Vol. C-31, July.

Redinbo, G., Napolitano, L. and Andaleon, D. (1993) "Multibit Correcting Data Interface for Fault-Tolerant Systems", In: IEEE Transactions on Computers, April.

Reed, R. (1997) "Heavy Ion and Proton Induced Single Event Multiple Upsets", In: Proceedings of IEEE Nuclear and Space Radiation Effects Conference (NSREC), July 1997.

Rockett, L. (1992) "SEU Hardened Scaled CMOS SRAM Cell Design Using Gate Resistors", In: IEEE Transactions on Nuclear Science, October.

Shirvani, P., Saxena, N. and McCluskey, E. (2000) "Software Implemented EDAC Protection Against SEUs", In: IEEE Transactions on Reliability, September.

Whitaker, S., Canaris, J. and Liu, K. (1991) "SEU Hardened Memory Cells for CCSDS REED-Solomon Encoder", In: IEEE Transactions on Nuclear Science, December.

Wrobel, F., Palau, J., Calvet, M., Bersillon, O., Duarte, H. (2001) "Simulation of Nucleon-Induced Nuclear Reactions in a Simplified SRAM Structure: Scaling Effects on SEU and MBU Cross Sections", In: Proceedings of IEEE Transactions on Nuclear Science, December 2001.