Towards Low-cost Soft Error Mitigation in SRAM-based FPGAs: a Case Study on AT40K

J. B. Ferron, L. Anghel, R. Leveugle
TIMA Laboratory (Grenoble INP, UJF, CNRS)
46 Avenue Félix Viallet - 38031 Grenoble Cedex - FRANCE
Lorena.Anghel@imag.fr, Regis.Leveugle@imag.fr

***Abstract*—Soft errors in the configuration memory of SRAM-based FPGAs cause significant and remanent application disturbances. Typical mitigation techniques induce large overheads in terms of resource usage and power consumption. We propose a new approach achieving efficient trade-offs between robustness and overheads, applied to the internal architecture of commercial AT40K devices.**

I. INTRODUCTION

Soft errors are spurious modifications of bits in memories or flip-flops, without any damage in the circuit [1]. They can be caused by several phenomena, including radiation effects. Initially well known in space applications or aeronautics, they are today an identified threat for most applications even at sea-level due to the increasing amount of memory and logic in integrated systems, and to technology evolutions leading to higher frequency and smaller voltage.

Many applications rely today on programmable devices such as FPGAs. This trend is increasing due to the huge investments necessary to develop a specific circuit (ASIC) in recent technologies. Reconfigurable devices are also very appealing because of their flexibility. Among those, SRAM-based FPGAs propose a larger logic density than Flash-based devices with in general lower cost. However, the design configuration is stored in a SRAM memory and is therefore quite sensitive to soft errors. In many applications, soft errors in the configuration memory must therefore be mitigated to achieve the required reliability or availability level.

Many works have been devoted in the last decade to this problem. However, the focus is in general on how protecting the system against all possible soft errors, accepting very large overheads to achieve this goal. In this paper, we propose a very different approach based on the maximum exploitation of unused resources to increase the design robustness at no (or low) cost. The approach is therefore not intended for high-reliability or high-availability applications, but for cost-sensitive applications with some level of reliability constraints. We also focus on soft errors in the configuration memory; errors in the user flip-flops may be mitigated with existing approaches such as error detecting codes, but have a much smaller probability due to the huge number of bits in the configuration.

Section II summarizes the main categories of approaches previously presented in the literature. Section III explains the basics of the proposed approach. A case study based on an AT40K device is reported in Section IV.

II. PREVIOUS WORKS

We will focus in this section on the techniques aiming at improving the dependability of designs implemented in reconfigurable FPGAs, and especially SRAM-based FPGAs, with respect to soft errors in the configuration memory. Several approaches have been reported; most of them fit into at least one of the following categories:

- replacing the typical SRAM cells by either hardened cells less sensitive to soft errors [2] or by new memory technologies [3], and/or proposing specific families based on hardened process [4, 5],
- proposing new FPGA architectures with error detection or error correction mechanisms directly implemented in the internal structure [6],
- modifying the architecture of the design itself so that error detection and/or correction can be achieved [4, 7-10],
- improving the placement and routing of the design so that less configuration bits are critical [11-14],
- monitoring the FPGA configuration at the system level [4].

The two first categories of approach require manufacturing of a specific FPGA. This is in general unaffordable for the final user and can only be considered by semiconductor manufacturers or for very specific applications as reported in [2]. A final user may of course take advantage of mechanisms included in some commercial FPGAs, such as periodic checksum verification available in some Altera devices [15].

When commercial FPGAs have to be used, most approaches are based on the use of massively redundant

This work is supported by French research funds in the frame of the CATRENE OPTIMISE project..

architectures for the design. The most common approach is to use triplication and voting (TMR); this has been proposed and automated for example for Xilinx devices with the XTMR implementation [4]. Many works are based on this level of redundancy, with several granularity levels [9] and sometimes limited to duplication on some I/Os [16]. Some works also tried to achieve detection by duplication, associated with some correction techniques [7]. The main drawbacks for this category of approaches are the huge overheads in terms of resource usage and in terms of power consumption. Since the main advantages of SRAM-based devices over for example Flash-based devices are related to density and cost, their advantages are considerably reduced when massive redundancy is applied. Also, using these approaches leads to the objective of maximum protection against soft errors; a trade-off between overheads and robustness is usually not considered.

Improving the placement and routing of the design has been proposed in several works. However, this improvement is in general targeted to designs already implemented with massive redundancy, and aims at avoiding single point failures due to routing paths that may induce some resource sharing between several copies of the design [11]. In a few cases, the routing modifications may include redundancy insertion [14]. Placement has also been optimized so that partial reconfiguration can be used to correct the circuit after error detection, but as a complement to TMR [10].

The last category of approach is based on the configuration readback capability available in some FPGAs. In that case, the configuration can be periodically checked to identify any spurious modification. This is called scrubbing and is often used in association with massive redundancy, to avoid error accumulations that may lead to failures in spite of the redundancy. Configuration corrections may take advantage of partial reconfiguration capabilities. Since readback mechanisms are not available in all FPGAs for security reasons, the configuration can also be periodically refreshed without reading; this may however lead to unacceptable performance loss since the application has to be periodically interrupted and the whole configuration has to be rewritten. In both cases, some controller must be implemented in the system to trigger the FPGA scrubbing or reconfiguration.

III. PROPOSED APPROACH

In the following, we will focus on the use of commercial FPGAs so no modification is possible in the FPGA structure or technology. Scrubbing may be used at the system level, but we will focus on the intrinsic robustness of the design implemented onto the FPGA. The goal is to insert a limited amount of redundancy so that effective errors occurring in the configuration can be detected and a signal can be sent to a system-level controller that can in that case trigger a reconfiguration. Modifications are made by inserting detection logic during the placement and routing phase, either avoiding overheads or limiting them to a predefined level. The goal is therefore not to achieve full protection as with TMR-based approaches, but to increase the robustness at no (or low) cost by taking advantage of unused resources for the implemented design.

A. *Target Device Architecture*

The reported study targets Atmel AT40K devices [17]. In order to achieve low implementation cost, the redundant elements must be carefully selected with respect to the implemented design and to the FPGA architecture. In consequence, the approach has to be defined for a given FPGA family.

Each AT40K device is a regular 2D array of basic cells called PLBs (Programmable Logic Blocks). Repeaters are inserted each four cells in the two directions to ensure good signal propagation on the interconnections. RAM blocks are inserted at each crossing between repeater lines and columns. These repeaters partition the global array into sectors of 16 PLBs. The PLB structure is illustrated in Figure 1. Each PLB contains two 3-input Look Up Tables (LUTs), one D flip-flop, a few logic gates and a set of elements configuring the interconnections with the other PLBs. The two LUTs can be used together in order to implement a 4-input Boolean function.

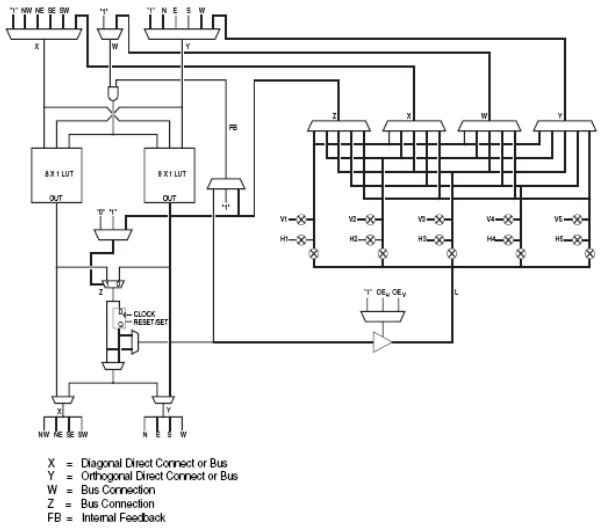

Figure 1. PLB structure in AT40K devices [17]

B. *Assumptions*

The proposed approach is focused on the protection of the LUT configurations; the bits configuring the interconnections will not be explicitly taken into account in this case study.

We focus on single-bit errors (SEUs) in the configuration memory since previous experiments had shown that multiple-bit errors have a much smaller probability [18]. However, multiple-bit errors can also be mitigated when the erroneous bits have uncorrelated roles in the device configuration, or when they are in the same LUT.

Preliminary identification of critical bits that may induce application interrupts or service deviations (SEFIs) can be done using static analysis, for example with the SEFEA-ProD tool [18]. LUTs without critical bits do not need to be protected. In case trade-offs are required between overheads

and dependability increase, the LUTs with the largest numbers of critical bits can be selected in priority for hardening.

C. *Hardening Approach*

As previously mentioned, the proposed mitigation approach is based on error detection, followed by a system-level reconfiguration. We will only discuss here the implementation of the detection capabilities.

The error detection is based on local duplication of the selected sub-functions configured by critical bits. The duplication and the comparison are both implemented within the logic sector in which the original sub-function has been placed, taking advantage of the resources unused during the placement and routing of the original function. Local error signals generated in the FPGA sectors are then combined to generate a global error detection signal. Partial reconfiguration limited to the erroneous sector may be triggered by the local error signals.

In order to achieve the best protection with the available resources, three cases have to be distinguished:

- Logic functions with 3 inputs or less, implemented by one LUT, the other LUT in the PLB being unused,
- Logic functions with 3 inputs or less, implemented by one LUT, the other LUT in the PLB being used to implement a second function,
- Logic functions with 4 inputs, implemented by using both LUTs in the PLB.

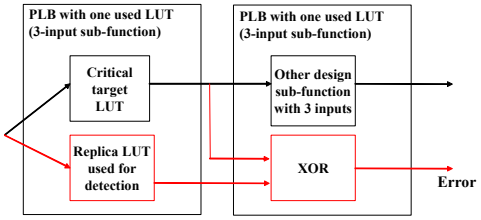

Figure 2. Basic scheme for one 3-input sub-function

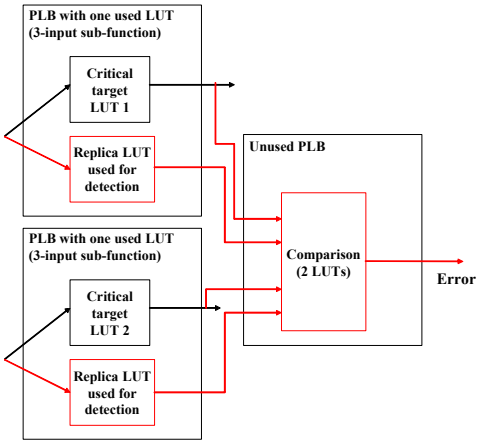

Figure 3. Basic scheme for two 3-input sub-functions

In the first case, the unused LUT allows us to implement a copy of the sub-function to protect. An additional LUT is used in another PLB of the same sector to implement a XOR function between the two LUT outputs in order to generate the local error detection as illustrated in Figure 2. In fact, this XOR function only requires 4 bits of a LUT. Depending on the initial design density, this scheme can be improved by protecting two different 3-input sub-functions in two PLBs, and using a PLB to globally generate the error signal for these two LUTs. This principle is illustrated in Figure 3 and leads to one LUT gained for the generation of the global error signal.

When the two LUTs of a PLB are used to implement a 4-input function, the duplicated sub-function must be implemented in another PLB of the same sector. A LUT in a third PLB has to be used for the XOR function.

When the two LUTs of a PLB are used to implement two 3-input functions, the implementation may be similar to the scheme in Figure 3, except the two critical LUTs are in the same PLB and an unused PLB has to be available for the two copies. If only one sub-function is critical, the implementation is similar to the scheme in Figure 2 but the replica must be implemented in another PLB than the critical sub-function.

The local error signals in each sector are locally combined and the sectors are chained so that each sector receives one error signal that is combined with the local ones, and outputs the global result to the next sector. The propagation time of the error signal is therefore larger than using a tree structure, but the routing modifications are noticeably simplified. In case the propagation time is too large, the FPGA can be partitioned in several areas, each of them generating an independent error signal.

The best case in terms of protection at the sector level is illustrated in Figure 4 for 3-input sub-functions and in Figure 5 for 4-input sub-functions. The maximum density leading to a complete protection without actual resource overheads is respectively 9 sub-functions and 6 sub-functions per sector. A larger resource usage in the initial design either requires trading-off the protection coverage with respect to the available unused PLBs, or accepting actual overheads and using a larger device.

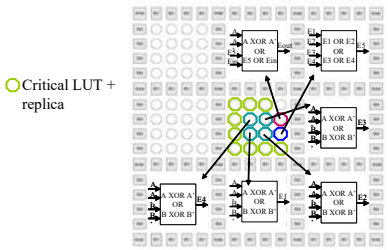

Figure 4. Best implementation case for 3-input sub-functions

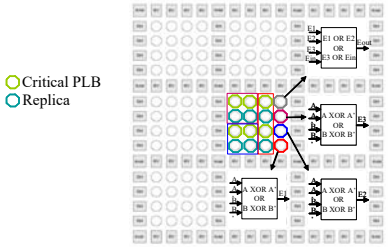

Figure 5. Best implementation case for 4-input sub-functions

Considering a device with M PLBs, and a design requiring to use N PLBs including X PLBs fully used and (N-X) PLBs implementing only one 3-input function, a complete protection of the LUTs can be achieved without actual overheads if: M>= 24/27*X +16/9* N, assuming no blocking routing congestion.

IV. Case Study

A. Implemented Design

The proposed approach has been applied to the implementation of a FIR digital filter. The circuit behavior is based on a 3-phase computation loop: sample acquisition (e.g., from an ADC converter), computation of the response and result emission on one output. The computation successively adds partial products of N samples stored in a FIFO memory with N pre-defined coefficients.

The implementation has been done on a device with 48*48 PLBs. The filter requires 814 PLBs after placement and routing by the Atmel tool IDS. 284 of these PLBs implement a single 3-input function.

B. Mitigation

Using the proposed approach, the protection of the LUTs in the 284 partially used PLBs requires using 221 additional PLBs, including the generation of the global error signals. The total number of PLBs required to protect all the LUTs in the design is 1105. This corresponds to an increase of 129% in resource usage and allows us in that case to implement the filter and its protections in the target device so without actual cost. Using massive redundancy such as TMR would have led to a better robustness since more configuration errors could be mitigated, but would have required to use a larger device.

V. Conclusion

The reported case study demonstrates the feasibility of a pragmatic approach improving the dependability of a design on SRAM-based FPGAs without incurring the large overheads of typical approaches. Most or all of the mitigation functions are implemented taking advantage of unused resources. Overheads can be trade-off against error coverage and can be reduced by preliminary identifying critical bits in the configuration. This case study was focused on bits configuring the Boolean functions (LUTs); further work includes the mitigation of some errors in the interconnections.

References

[1] R. Baumann, "Soft errors in advanced computer systems", IEEE Design & Test of Computers, vol. 22, no. 3, May-June 2005, pp. 258-266

[2] S. Bonacini, F. Faccio, K. Kloukinas, A. Marchioro, "An SEU-robust Configurable Logic Block for the implementation of a radiation-tolerant FPGA", IEEE transactions on Nuclear Science, vol. 53, no. 6, Part I, December 2006, pp. 3408-3416

[3] L. Torres, Y. Guillemenet, S. Z. Ahmed, "A dynamic reconfigurable MRAM based FPGA", ERSA'2010, pp.31-40

[4] http://www.xilinx.com/esp/mil_aero/index.htm

[5] http://www.atmel.com/dyn/products/devices.asp?category_id=172&family_id=641&subfamily_id=1477

[6] Q. Zhao, Y. Ichinomiya, M. Amagasaki, M. Iida, T. Sueyoshi, "A novel soft error detection and correction circuit for embedded reconfigurable systems", IEEE Embedded Systems Letters, vol. 3, no. 3, September 2011, pp. 89-92

[7] F. Lima, L. Carro, R. Reis, "Designing fault tolerant systems into SRAM-based FPGAs", 40th Design Automation Conference (DAC), 2003, pp. 650-655

[8] F. Gusmao de Lima Kastensmidt, G. Neuberger, R. F. Hentschke, L. Carro, R. Reis, "Designing fault tolerant techniques for SRAM-based FPGAs", IEEE Design & Test of Computers, vol. 21, no. 6, November-Décember 2004, pp. 552-562

[9] F. Gusmao de Lima Kastensmidt, L. Sterpone, M. Sonza Reorda, L. Carro, On the optimal design of triple modular redundancy logic for SRAM-based FPGAs, Design, Automation and Test in Europe Conference (DATE), March 7-11, 2005, pp. 1290-1295

[10] C. Bolchini, A. Miele, M. D. Santambrogio, "TMR and partial dynamic reconfiguration to mitigate SEU faults in FPGAs", IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, 2007, pp. 87-95

[11] L. Sterpone, M. Violante, "A new reliability-oriented place and route algorithm for SRAM-based FPGAs", IEEE transactions on Computers, vol. 55, no. 6, June 2006, pp. 732-744

[12] C. Bolchini, A. Miele, C. Sandionigi, N. Battezzati, L. Sterpone, M. Violante, "An integrated flow for the design of hardened circuits on SRAM-based FPGAs", 15th IEEE European Test Symposium, Prague, Czech Republic, May 24-28, 2010, pp. 214-219

[13] M. A. Abdul-Aziz, M. B. Tahoori, "Soft error reliability aware placement and routing for FPGAs", International Test Conference (ITC), 2010, pp. 1-6

[14] C. Kinzel Filho, F. Lima Kastensmidt, L. Carro, "Improving reliability of SRAM-based FPGAs by inserting redundant routing", 8th European Conference on Radiation and its Effects on Components and Systems (Radecs 2005), Cap d'Agde, France, September 19-23, 2005

[15] http://www.altera.com/literature/an/an357.pdf

[16] M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, S. Pastore, G. R. Sechi, R. Weigand, "Evaluation of single upset mitigation schemes for SRAM bassed FPGAs using the FLIPPER fault injection platform", IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, 2007, pp. 105-113

[17] http://www.atmel.com/products/fpga/?category_id=172&family_id=623

[18] J.B. Ferron, L. Anghel, R. Leveugle, A. Bocquillon, F. Miller, G. Mantelet, "A methodology and tool for predictive analysis of configuration bit criticality in SRAM-based FPGAs: experimental results", 3rd International Conference on Signals, Circuits & Systems (SCS), Djerba, Tunisia, November 6-8, 2009