Restoring Reliability in Fault Tolerant Reconfigurable Systems

Manuel G. Gericota
DEE/ISEP
Rua Dr. Antº Bernardino de Almeida
4200-072 Porto
PORTUGAL
mgg@dee.isep.ipp.pt

José M. Ferreira
DEEC/FEUP
Rua Dr. Roberto Frias,
4200-465 Porto
PORTUGAL
jmf@fe.up.pt

Abstract

The new generations of SRAM-based FPGA devices, built on nanometer technology, are the preferred choice for the implementation of reconfigurable computing platforms. However, smaller technological scales increase their vulnerability to manufacturing imperfections and hence to the occurrence of electromigration. Moreover, the large internal RAM (for configuration purposes or as embedded memory blocks) makes them more prone to soft errors.

The incorporation of self-reconfiguration capabilities in recent FPGAs, allied to the use of soft and hard microprocessor cores, facilitates the offset of these vulnerabilities by enabling the development of self-restoring fault tolerant reconfigurable systems. In the methodology presented in this paper, the embedded microprocessor is also responsible for the implementation of online self-test-and-repair strategies, based on modular redundancy and on self-reconfiguration. The detection of faults, caused by soft or hard errors, may be followed by repairing actions, depending on the fault type. This approach leads to smoother system degradation, extending its lifetime and improving its reliability.

1. Introduction

The reduction in size of the transistors in each new generation of semiconductor technology led to a greater integration and to a per unit power reduction, enabling chips to grow in size and complexity. But new nanometer scales also brought some negative aspects, such as a greater vulnerability to manufacturing imperfections and to soft errors, due to Single Event Upsets (SEU) originating in background radiation. Soft errors do not physically damage the chip, but the values stored in memory cells may be affected. In contrast, manufacturing imperfections may lead to the occurrence of electromigration, which may result in permanent physical damages to the chip after large periods of operation.

These problems have a particular impact on the reliability of SRAM-based Field Programmable Gate Arrays (FPGAs). The exponential growing on the amount of memory cells needed for configuration purposes makes these components especially vulnerable to soft errors. Moreover, the amount of embedded memory blocks available for user's applications has also been growing.

The recent addition of new features, such as dynamic reconfiguration and self-reconfiguration, may help to cope with the previous problems, in particular when dealing with critical applications that require a high reliability level. Dynamic reconfiguration extends FPGA's flexibility by enabling multiple independent functions, from different applications, to share the same logic resources in the spatial and temporal domain [1]. More recently, and via self-reconfiguration [2], it became possible for an implemented function to control the dynamic reconfiguration of its own FPGA.

The advantages of using dynamic reconfiguration in the implementation of online structural test and fault tolerance strategies were largely explored in previous works [3, 4]. However, those approaches relied on a rotate and test methodology, which created a test latency that degraded the performance of the test strategy. If a defect affects the functionality of a given function, the resulting fault will be propagated to the rest of the circuit until the test function has reached the defective resource. By then, the fault could already have caused the irreversible malfunctioning of the whole system, eventually interrupting its operation.

Traditionally, highly critical applications relied on hardware redundancy, like Triple Modular

Redundancy (TMR), illustrated in figure 1, to increase their reliability. In this method, extra components are used to instantaneously mask the effect of a faulty component, meaning that no propagation of the fault will occur. Each module may be a complete system, such as a computer, or a less complex unit, like a microprocessor or even an adder or a gate. The voting element accepts the outputs from the three sources and delivers the majority vote at its output.

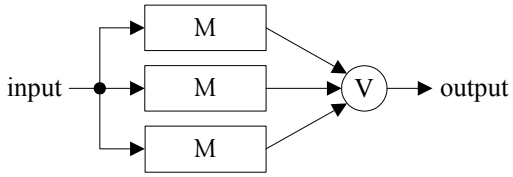

Fig. 1. Triple Modular Redundancy

In this case, it is assumed that the majority voter does not fail, which is an unrealistic principle. However, its reliability can be improved by replicating it as well.

In new nanometre technology the use of fault tolerance mechanisms is essential, not only due to SEU induced errors, but because it is unrealistic to expect that a manufacturing test will cover all possible faults. In particular, delay faults emerging from defects of resistive type, or due to crosstalk or ground bounce, are almost impossible to foresee [5].

The inclusion of spatial or temporal redundancy leads to an increase in the reliability level of a system. However, its cost rises enormously, not only in terms of the direct cost of components and space required for its implementation, but also in terms of power consumption. Reconfigurability is one way of improving the reliability of a system without implying a proportional rise in costs.

A new methodology to increase the reliability of systems implemented in dynamically reconfigurable FPGAs is presented in the next sections, followed by the discussion of several aspects related to its practical implementation. Future research lines are presented in the concluding section.

2. SEU effects over FPGAs

In non-reconfigurable technologies, such as ASICs, the protection against SEUs is restricted to flip-flops, because logic paths between them are typically hard-wired. Notwithstanding, Single Event Transients (SETs) may be propagated to flip-flop inputs, where they have a high probability to be registered, causing soft-errors in the user data [6]. Further protection would only be achieved through full module redundancy. This is also a preferred choice to improve the reliability of highly critical applications based on FPGAs [7-11]. Due to their inherent configurability, FPGAs are especially suitable for the implementation of modular redundancy, since it does not require any new architectural feature and it is function independent. However, the dependency on memory cells to define logic paths makes them also susceptible to SEUs. Again in this case, the only effective protection is full module redundancy [8].

Possible errors in the on-chip configuration memory cells may be recovered by simply performing a partial readback operation of the configuration of the detected faulty module. The retrieved bitstream is compared with the original configuration, and if a modification is detected, the correct bitstream can be re-established through partial reconfiguration. This technique is known as scrubbing, and defined as the process of re-writing the configuration memory during (and without disturbing) normal FPGA operation [12].

The circuit that controls the FPGA reconfiguration should also have a compatible reliability index. This circuit, typically a microprocessor, does not need to be a dedicated configuration module. Due to the usual long time interval between module failures [13], a generic soft microprocessor core that carries on other tasks related to the operation of the whole system may be used.

The readback and partial reconfiguration do not affect the data stored in flip-flop registers, and consequently soft-errors in data registers cannot be recovered using this method. However, due to the transient nature of upsets, the error will be recovered by the circuit in the subsequent update of the affected flip-flop. The propagation of soft-errors, either affecting data registers or the functionality of the circuits, is avoided by redundancy.

3. Reliability of circuits in FPGAs

In a discrete implementation of a TMR system, if a defect affects the functionality of one module, reliability decreases, but the system still works correctly. A second failure in one of the remaining modules may lead to a malfunctioning of the system. Ideally, when a module fails, it should be replaced to restore the initial redundancy. However, this action may not be possible immediately. In certain cases, like in space applications, it may even be impossible.

The great advantage of using FPGAs is that, in the event of a module failure, a diagnose-and-repair mechanism may be activated and the initial redundancy re-established. This may be done transparently and without human intervention, since physical component replacement is not needed.

It is not easy to detect a fault in a TMR implementation using traditional online test strategies, due to the masking properties of

redundancy. In our approach, the detection of the faulty modules is done through a scan chain that regularly captures the values at the outputs of all the modules and voters, including those of the microprocessor, as shown in figure 2. The resulting bitstream is shifted to the microprocessor where it is analyzed. If an incoherency is detected, the module or voter where it was produced is probably not working as expected.

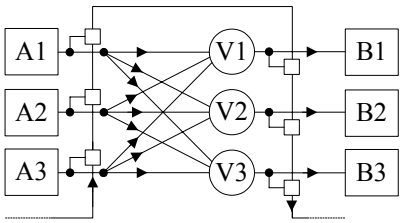

Fig. 2. Example of a T-TMR implementation with an embedded scan chain for faults detection

4. Detection-Diagnose-and-Repair

If an error is detected in the configuration bitstream, it is possible to confine its origin to the space between the scan chain cells, corresponding to the module or voter where the value was captured, and to the interconnections in-between [14]. Three possible causes for the fault may be considered:

1. the faulty value is due to a transient failure caused by a SEU or SET affecting one of the circuit registers;
2. the faulty value is due to a transient failure caused by a SEU affecting a configuration memory cell, which leads to a change in the functionality of the module or voter, or the interconnections.
3. the faulty value is due to a permanent physical defect affecting the structure of the FPGA.

The first case may be immediately excluded if the error is captured at the output of a voter, since voters are typically implemented using combinational logic only. If it has its origin in a module, one can expect that the fault will be automatically corrected at the next register update. A new scan chain capture operation may show that the error has already been fixed and no further action is needed. If not, the second situation may have occurred. A background task is launched to readback part of the configuration bitstream of the area where the affected module is implemented. Comparison with the original bitstream may be done by bit comparison or Cyclic Redundancy Check (CRC). If an incoherency is found, the microprocessor performs a partial reconfiguration, restoring the original configuration and eliminating the cause of the failure. The output of the module should now be captured again and its correctness verified. If no error on the configuration bitstream is detected after the readback-and-compare operation, but the fault persists, the most probable reason is the existence of a physical defect in the array. In that case, it is necessary to reconfigure the module, avoiding the faulty area, in order to restore the reliability index. The module should be dynamically replicated and its operation transferred to the new module. This transference should neither affect the operation of the function nor introduce disturbances in the output signals [1]. After that, the resources occupied by the faulty module are released and subsequently tested to detect and diagnosis the origin of the fault. This procedure is controlled by the microprocessor [3, 4]. Any resource found defective is flagged to avoid its reuse in later reconfigurations. The remaining resources that are tested OK can be reused in later replacements of any other faulty module. In this way, the available spare resources are almost entirely restored for future replacements. Figure 3 shows the diagram flow of the proposed methodology.

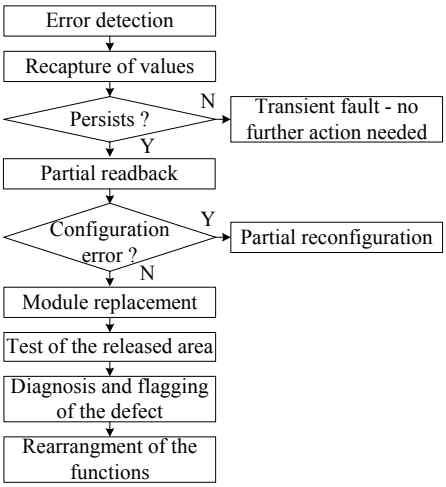

Fig. 3. Flowchart of the detection-diagnose-and-repair methodology proposed

The use of design diversity, where each redundant module is synthesized using a different synthesis technique (which leads to different implementations of the same logic circuit), further enhances reliability. The emergence of possible systematic defects in specific parts of the configurable logic space may eventually lead to the simultaneous failure of more than one module of the same function (and consequently of the function itself), if all its modules are identical and implemented using equal resources [11].

This methodology extends the reliability of each function and enables a smoother degradation of the global reliability index. Despite being a static TMR implementation, a faulty module is dynamically

repairable *n* times, where *n* depends on the cause of the failure. If the origin is not a permanent physical defect, then *n* is infinite. Otherwise, *n* depends on the initial amount of spare resources and on the location of the defects that affect the structure of the FPGA.

The microprocessor is also implemented using TMR to keep its reliability index compatible with the remaining blocks. The microprocessor is divided in small functional modules, facilitating replacement in case of fault detection, and reducing the spare space needed for replication. If the defective module is part of one of the three implemented processors, the remaining two will be responsible for the replication of the malfunctioning module. Subsequent test procedures will already be assumed by the whole three.

Self-reconfiguration is necessary to embed the whole system in a single FPGA, including the self-tolerance features. The Virtex-II and Virtex-II Pro families have an Internal Configuration Access Port (ICAP). ICAP enables a hard- or soft- implemented microprocessor to control its own dynamic reconfiguration or the reconfiguration of any external modules, without stopping or disturbing the operation of the whole system. In this way, a self--healing system may be included in a single FPGA.

5. Conclusions

Despite the generalized idea that TMR makes FPGAs virtually immune to defects or SEUs, further research is necessary and several issues related to their use in reconfigurable systems have yet to be considered, namely the following:

- the probability of total failure due to a single-event-functional-interrupt (SEFI), caused by an upset in the device Power-On Reset (POR), which leads to the total clearing of the configuration memory and causes the loss of state data;
- the probability of a fault in a function output due to bridging faults between modules;
- the possibility of an incorrect module or voter failure diagnosis caused by defects or upsets affecting the scan chain that captures their outputs;
- the vulnerability of the ICAP to defects or upsets;
- the influence, over voter output, of the position of replicated modules concerning their original location (due to a variation on the path delay between modules and voters);
- the vulnerability of the memory holding the original or current configuration file, which must also be protected against upsets using error checking and correction techniques;
- the possibility of an upset to change the content of the memory block holding the microprocessor program, since TMR does not offer any protection in case of software errors.

Work is being done in these issues to prepare the experimental stage that will validate the proposed methodology.

References

[1] M. G. Gericota, G. Alves, M. L. Silva, J. M. Ferreira, "Run-time Defragmentation for Dynamically Reconfigurable Hardware", *in: "New Algorithms, Architectures, and Applications for Reconfigurable Computing*". Springer, December 2004.

[2] B. J. Blodget, S. P. McMillan, P. Lysaght, "A lightweight approach for embedded reconfiguration of FPGAs", *Proc. DATE Designers' Forum*, pp. 399-400, 2003.

[3] M. Abramovici, C. Stroud, C. Hamilton, S. Wijesuriya, , Verma, V., "Using Roving STARs for On-Line Testing and Diagnosis of FPGAs in Fault-Tolerant Applications", *Proc. ITC*, pp. 973-982, 1999.

[4] M. G. Gericota, G. Alves, M. L. Silva, J. M. Ferreira, "Active Replication: Towards a Truly SRAM-based FPGA On-Line Concurrent Testing", *Proc. IOLTW*, pp. 165-169, 2002.

[5] M. Nicolaidis, L. Anghel, "Concurrent checking for VLSI", *Microelectronics Journal*, Vol. 49, Nos. 1-2, pp. 139-156, November 1999.

[6] D. Alexandrescu, L. Anghel, M. Nicolaidis, "Simulating single event transients in VDSM ICs for ground level radiation", *JETTA: Journal of Electronic Testing-Theory and Applications*, Vol. 20, No. 4, pp. 413-421, 2004.

[7] J. Moore, "Design Security in SRAM-based FPGAs", *MAAPLD Conf.*, 2003.

[8] C. Carmichael, "Triple Module Redundancy Design Techniques for Virtex FPGAs", *XAPP 197 Application Note*, Xilinx, Inc., 37 p., 2001.

[9] TMRTool. Information available at: http://www.xilinx.com/products/milaero/tmr/

[10] C. Carmichael, E. Fuller, P. Blain, M. Caffrey, "SEU Mitigation Techniques for Virtex FPGAs in Space Applications", *MAAPLD Conf.*, 1999.

[11] N. R. Saxena, S. Fernandez-Gomez, Wei-Je Huang, S. Mitra, Shu-Yi Yu, E. J. McCluskey, "Dependable Computing and Online Testing in Adaptive and Configurable Systems", *IEEE Design and Test of Computers*, Vol. 17, No. 1, pp. 29-41, January-March 2000.

[12] C. Carmichael, M. Caffrey, A. Salazar, "Correcting single-event upsets through Virtex Partial Configuration", *XAPP 216 Application Note*, Xilinx, Inc., 12 p., 2000.

[13] P. L. Murray, "Re-Programmable FPGAs in Space Environments". Available at: http://www.seakr.com/data/Unsorted/reprogrammable_fpga_in_space1.doc

[14] J. H. Lala, R. E. Harper, "Architectural principles for safety-critical real-time applications", *Proceedings of the IEEE*, Vol. 82, No. 1, pp. 25-40, January 1994.