# Using Partial Dynamic FPGA Reconfiguration to Support Real-Time Dependability

José Luís Nunes, João Carlos Cunha
Polytechnic Institute of Coimbra / CISUC – ISEC/DEIS
3030-199 Coimbra, Portugal
{jnunes, jcunha}@isec.pt

Raul Barbosa, Mário Zenha-Rela
University of Coimbra / CISUC – DEI
3030-290 Coimbra, Portugal
{rbarbosa, mzrela}@dei.uc.pt

## ABSTRACT

*Field Programmable Gate Arrays (FPGAs), are being increasingly used in custom systems requiring fast time-to-market delivery due to their flexibility; being reprogrammable in the field is a real value for long unattended operation and whenever on-site maintenance is costly as is the case in many remote data acquisition stations. The most powerful FPGAs are based in SRAM technology which is particularly prone to transient faults. Fault-tolerance is therefore mandatory and this can be done by simply reprogramming the FPGA, thus repairing the corrupted configuration. Recent advances in FPGA technology allow the configuration of just a portion of the FPGA, which lowers significantly the time overhead, while the remaining parts are running. This truly dynamic partial FPGA reconfiguration can be used to provide fault-tolerance for hard real-time applications, guaranteeing high reliability in long missions. This short paper addresses the technological aspects of partial dynamic reconfigurable FPGAs, presents some of most important threats to dependability of these devices, and identifies some research areas to investigate in order to increase dependability.*

## Categories and Subject Descriptors

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

## General Terms

Reliability.

## Keywords

FPGA reconfiguration, dependability.

## 1. INTRODUCTION

The ubiquitous usage of embedded devices in today's technological society is an unquestionable fact. Actually, 98% of computing devices are embedded in all sort of electronic devices and machines. Although the large majority of these devices are low-cost high-volume products, a significant number of applications require reconfigurability after deployment, in order to accommodate new functionalities, upgrades or even corrections.

In this scenario, the number of embedded systems using Field Programmable Gate Arrays (FPGA) is increasing, as they allow fast time-to-market delivery by offering the possibility of being upgraded after deployment. Pushed by demand and technological advances, these devices are becoming increasingly flexible and powerful [1].


*EWDC '11*, May 11-12, 2011, Pisa, Italy


Some of the most recent FPGAs are based in SRAM technology, which has the dual benefit of fast reprogramability and high density. However these advantages come at the cost of being particularly prone to transient faults, especially radiation induced [5], which may affect the configurable logic elements, thus the correct functioning of the device. Some sort of fault-tolerance is therefore mandatory on most applications to achieve a desired level of dependability [8].

Recent advances in FPGA technology allow the configuration of just a portion of the FPGA lowering significantly the time overhead for this operation which, in some situations, can be performed between different stages of system computation. It is thus fast enough to stop the system, reconfigure the FPGA, and resume operation without interfering with its functional behavior, as perceived by the outside world [2]. However, this operation is only feasible whenever the application dynamics allows it, which is not the case in many applications requiring high reliability, and truly uninterrupted service, such as in many hard real-time systems.

In some scenarios it is possible to partially reconfigure the FPGA, while the remaining parts are running [6]. This truly dynamic partial FPGA reconfiguration can be used to provide fault-tolerance and hard real-time behavior under both transient and permanent faults, thus ensuring high reliability even in long missions where the use of TMR is not feasible.

## 2. FPGA RECONFIGURATION

State-of-the-art SRAM-based FPGAs are composed by a matrix of Configurable Logic Blocks (CLB) interconnected by routing resources, with I/O capabilities and special blocks of RAM. The device comprises a SRAM, named configuration memory, for implementing the logic and necessary routing. Every time the device starts, the configuration memory is programmed with a *bitstream*, usually stored in a flash device, generated by a design tool using a graphical or a programming language approach.

To reduce programming times, manufacturers first introduced the capability of *partial reconfiguration* in SRAM-based FPGAs, allowing the reconfiguration of the device by programming just a fraction of the FPGA configuration memory. Then, *dynamic partial reconfiguration* was added to support the partial reconfiguration while a static module is running inside de device. In this case, an external port is used to reprogram the device. Finally, *dynamic partial self-reconfiguration* led to the introduction of ICAP, an internal port that allows a module running inside the FPGA to reprogram the configuration memory, and thus the device functionality.

*Dynamic partial reconfiguration*, using either an internal or external port opens the possibility of 1) changing the device

functionality during execution time, thus reducing FPGA footprint, as usually not all modules need to be running at the same time and can be "loaded" as needed [7]; and 2) recover faulty modules affected by transient faults by "reloading" the module's *bitstream* from permanent storage [2].

# 3. FPGA FAULT MODEL

Given that our goal is to ensure the dependability of systems that use FPGAs as the final solution, we must consider the threats to which such circuits are exposed after deployment. The main concerns are faults affecting the configuration memory. Thus, we focus on faults affecting SRAM-based memory.

A transient fault affecting the *configuration bits*, which define the circuit's static logic, causes an error that will remain until the corresponding *bitstream* is reloaded (i.e., it will be permanent unless one recovers from it). An example is a Single Event Upset (SEU), which is typically a transient disturbance, but that may cause a permanent error in the system if the affected bits are not corrected. Since the configuration bits dominate the FPGA usage [3], we regard transient faults affecting configuration bits as the most relevant threat.

In addition to transient faults, we also consider permanent faults affecting the configuration memory, e.g., due to hardware aging. Such faults cause permanent damage to SRAM cells and reloading the *bitstream* is unlikely to correct the resulting errors. There is a need to circumvent the permanently damaged portion of the circuit. Thus, in summary, our focus is on how to recover from transient faults in the configuration bits, as well as permanent faults in the entire configuration memory.

# 4. HIGH RELIABILITY IN REAL-TIME

From the previous sections we observed that transient faults affecting the *configuration bits*, which define the circuit's static logic and routing inside the FPGA, although leading to a permanent error, may be easily tolerated by reloading the *bitstream* into the SRAM, thus dynamically reconfiguring the FPGA. This is a kind of time redundancy which has the advantage of having no spatial overhead and minimum power consumption. However, this time overhead may be unfeasible for more time sensitive applications. Even the use of partial reconfiguration, which reduces significantly this overhead, allowing a short suspension of the computation during reconfiguration, is not acceptable in hard real-time applications. In such cases, hardware redundancy must be used to ensure strict timely behavior.

To guarantee a high level of reliability without delaying or suspending the service, the redundant modules must be kept working, such as in the case of a TMR or a standby-spare in hot standby. This assures an uninterrupted service in case one of the active modules fails, leaving the system in a degraded system dependability level. This can be achieved with a single FPGA, containing the multiple replicas [4].

This is where partial dynamic reconfigurability of FPGAs plays an important role. In order to recover system dependability, thus providing high-reliability, the failed module may be dynamically repaired, either by reloading the *bitstream* in the FPGA frames that contained the failed module or, if the fault was permanent, by moving the module into other free frames of the FPGA.

This dynamic FPGA reconfiguration poses, however, some challenges. First of all, during partial reconfiguration, the active module execution must not be suspended or delayed. This is really a challenge since connecting the recovered module to the voter or switch while this one is executing is not trivial: the modules must be connected to a common bus and some redundant hardware arbitration must be in place. Another important issue regards recovering the state of this module, since it must execute in parallel with the active module(s). This must be done using forward error recovery and, by the end of the recovery process, it must run in synchrony with the active module(s). Finally, power consumption should be also a concern when targeting embedded systems, since using active redundant modules may more than triplicate the increase of power consumption.

# 5. CONCLUSIONS AND CURRENT WORK

In this paper we addressed a novel approach to guarantee hard real-time performance under the occurrence of both permanent and transient faults using hardware reconfiguration. This is only possible due to recent advances in FPGA technology that allows dynamic reconfiguration of parts of the IC while other parts keep working in parallel, unaffected by the reconfiguration. This is a major technological breakthrough in programmable hardware that can be used for dependability purposes. At the very least it becomes possible to recover from transient faults by simply reloading the original *bitstream*, or from permanent faults by relocating it into another area of the FPGA.

We are currently investigating the applicability of this approach in the development of real-time embedded controllers, requiring high reliability in long-term missions.

# 6. REFERENCES


[1] Rana, V., Santambrogio, M., and Sciuto, D. 2007. Dynamic Reconfigurability in Embedded System Design. *IEEE International Symposium on Circuits and Systems* (New Orleans, LA, May 2007).

[2] Bolchini, C., Quarta, D., and Santambrogio, M. D. 2007. SEU Mitigation for SRAM-Based FPGAs through Dynamic Partial Reconfiguration. In *Proceedings of the 17th ACM Great Lakes symposium on VLSI*, ACM, NY, USA, 55-60.

[3] Asadi, H., Tahoori, M. B., Mullins, B., Kaeli, D., and Granlund, K. 2007. Soft Error Susceptibility Analysis of SRAM-Based FPGAs in High-Performance Information Systems. In *IEEE Transactions on Nuclear Science* (Dec. 2007), vol. 54, no. 6, pp. 2714–2726.

[4] Bolchini, C., Miele, A., and Santambrogio, M. 2007. TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs. 2007. *International Symposium on Defect and Fault-Tolerance in VLSI Systems* (Rome, Italy, Sep. 2007).

[5] Baumann, R. 2005. Soft errors in advanced computer systems. *IEEE Design & Test of Computers* (May–June 2005), 258–266.

[6] Blodget, B., James-Roxby, P., Keller, E., McMillan, S., and Sundararajan, P. 2003. A Self-reconfiguring Platform. *Field-Programmable Logic and Applications* (Lisbon, Sep 2003). 565-574

[7] Upegui, A., and Sanchez, E. 2005. Evolving hardware by dynamically reconfiguring Xilinx FPGAs. *Evolvable Systems: From Biology to Hardware*, vol. 3637/2005, 56–65.

[8] Cheatham, J., Emmert, J., and Baumgart, S. 2006. A survey of fault tolerant methodologies for FPGAs. *ACM Trans. Des. Autom. Electron. Syst.* 11, 2 (April 2006), 501-533.