# On-Line Testing of Permanent Radiation Effects in Reconfigurable Systems

Luca Cassano[†], Dario Cozzi[‡], Sebastian Korf[‡], Jens Hagemeyer[‡], Mario Porrmann[‡], Luca Sterpone[*]
[†]Department of Information Engineering, University of Pisa, Italy
[‡]Center of Excellence Cognitive Interaction Technology, Bielefeld University, Germany
[*]Dipartimento di Automatica ed Informatica, Politecnico di Torino, Italy
emails: [†]luca.cassano@ing.unipi.it, [*]luca.sterpone@polito.it
[‡]{dcozzi, skorf, jhagemey, mporrmann}@cit-ec.uni-bielefeld.de

***Abstract*—Partially reconfigurable systems are more and more employed in many application fields, including aerospace. SRAM-based FPGAs represent an extremely interesting hardware platform for this kind of systems, because they offer flexibility as well as processing power. In this paper we report about the ongoing development of a software flow for the generation of hard macros for on-line testing and diagnosing of permanent faults due to radiation in SRAM-FPGAs used in space missions. Once faults have been detected and diagnosed the flow allows to generate fine-grained patch hard macros that can be used to mask out the discovered faulty resources, allowing partially faulty regions of the FPGA to be available for further use.**

***Keywords*-Automatic Test Pattern Generation, Fault Diagnosis; On-Line Testing; Permanent Radiation Effects; SRAM-FPGA**

## I. Introduction and Related Work

Reconfigurable systems are gaining more and more interest in the field of space applications [1]. SRAM-based FPGAs represent an extremely interesting hardware platform for this kind of systems thanks to high performance, relatively low power consumption and low cost [2]. Moreover partial dynamic reconfiguration of modern FPGAs enables maximum flexibility and can be used for performance increase, energy efficiency improvement, and fault tolerance enhancement [2].

Radiations are responsible for causing instantaneous and long-term damages in digital devices. Instantaneous damages are *Single Event Upsets (SEUs)*, i.e., changes of the content of memory elements, and *Single Event Transients (SETs)*, i.e., transient undesired electrical impulses [3]. Long-term damages are caused by the *Total Ionizing Dose (TID)*, i.e., the accumulation of charge trapped in the oxide layer of transistors in CMOS circuits [4]. TID first causes degradation of the performance of the system and ultimately it may also cause the complete destruction of parts of the system.

Among the published work on FPGA testing, two families of methods may be distinguished: *application-independent* and *application-dependent* methods. Application-independent methods, such as [5], [6], [7], [8] generally aim at detecting structural defects due to the manufacturing process of the chip. These techniques are performed off-line by the FPGA manufacturer and they target every possible fault in the device without any consideration of which parts of the FPGA are actually used by a given design. Application-dependent methods [9], [10], [11] address only those resources of the FPGA actually used by the implemented system. These techniques can be applied either off-line or on-line by the user after the system design has been defined. Application-dependent methods have been proposed for in-service testing of both structural defects [9], [10] and SEUs [11].

978-3-9815370-0-0/DATE13/©2013 EDAA

The problem of detecting faults caused by the long term exposure to radiation of an FPGA has not yet exhaustively been explored. To the best of our knowledge, the only work addressing the problem of on-line testing of faults due to long term exposure to radiations is the one reported in [12]. The work is focused on the description of the architecture used for testing reconfigurable areas without interfering with the normal functioning of the system, while few information about how the test hard macros (HMs) and the corresponding test patterns are generated is given. Moreover the presented test architecture addresses any possible faults in the logic components of the FPGA, while faults in the routing structure are not considered.

In this work we report about the ongoing development of a software flow for the generation of HMs (and the associated test patterns) for testing fault induced in SRAM-based FPGAs by the long term exposure to radiations. The proposed flow addresses any faults in both the logic and the routing components of SRAM-based FPGAs. Moreover test HMs can be generated for both area-oriented testing, i.e., testing every resource available in a particular area of the FPGA, and for module-oriented testing, i.e., testing only those resources used by a given module. For this reason we believe that such a flow could represent the basis for an on-line on-demand accurate testing service for partially reconfigurable computing systems used in space missions.

The remainder of the paper is organized as follows: Section II gives some background about the existing tools and about how we intend to modify them; Section III presents the structure of the proposed flow; Section IV concludes the paper.

## II. Background

In this section we give background information about the previously developed tools that, once modified to address permanent faults, will be employed in the presented flow. We point out that these tools have been designed to analyse the effects of SEUs in the configuration memory of FPGAs. These faults can simply be detected through the readback of the configuration memory itself. The modified versions of the tools will analyse the effects of permanent faults in FPGAs.

### A. The STAR Tool

STAR [13] is an analytical tool able to determine which are the critical resources for a given reconfigurable module (or fabric area) inside an SRAM-based FPGA. The critical resources for a given reconfigurable module (or fabric area) are those resources whose faulty behaviour directly affects the behaviour of the considered reconfigurable module (or fabric area). In the proposed software flow the STAR tool will be used to identify the critical resources of the reconfigurable module (or fabric area) that has to be placed on the FPGA under test.

### B. The DHHarMa Tool

DHHarMa [14] is a software flow for the automatic generation of homogeneous HMs starting from high level HDL descriptions. Using a data base containing the complete description of the resources of a given FPGA and the connections among them, DHHarMa is able to homogeneously pack, place and route generic HDL HM descriptions. In the presented software flow, DHHarMa will be used to automatically generate a set of HMs that use all the resources that have to be tested. On the one hand the generated HMs should be as large and dense as possible, so to occupy a large number of resources (thus a small number of macros would be necessary to test the whole FPGA). On the other hand very large and dense HMs could be difficult to test and to place on the FPGA. The modified version of DHHarMa should find a good trade-off between these issues in order to generate optimized HMs.

### C. The SEU-X Tool

SEU-X [15] is a tool for analysing the testability of SEUs affecting the configuration memory of SRAM-based FPGA systems. In particular the tool is able to formally demonstrate which of the SEUs cannot be tested. In the presented software flow a version of the tool, modified to keep into account permanent faults due to radiations, will be used. In particular the tool will be used to determine whether the generated test HMs are fully testable or not, and thus if using the set of generated test HMs it is possible to test all the possible faults, or if more test HMs are necessary.

### D. The Genetic Algorithm Based Test Pattern Generation Tool

The tool presented in [11] is a genetic algorithm-based test pattern generator addressing SEUs affecting the configuration memory of the resources actually used by a system implemented on an SRAM-based FPGA. It has been designed in order to maximize the fault coverage obtained by the generated test patterns on the one hand and to minimize the length of the test patterns themselves on the other hand. Also this tool will be modified to take into account permanent faults. In the presented flow it will be used to generate test patterns associated with the generated test HMs.

## III. The Proposed Software Flow

The aim of the proposed software flow is supporting on-line on-demand testing, diagnosis and fault masking architectures for dynamically reconfigurable systems like [1], [12].

The proposed software flow is hierarchical and it is developed in two steps. The first step aims at determining whether a given area of an SRAM-based FPGA is free from permanent faults due to radiation. This first step will be performed very often thus it must be as fast as possible. If a faulty behaviour of the system is detected, the second step performs a fine-grained diagnosis in order to isolate the faulty resources. Since this second step is performed only when a fault occurred in the system, it can take a much longer time then the first step.

After the faulty resources have been detected, a patch HM is created in order to mask them out, thus allowing partially faulty regions of the FPGA to be available for further use. For masking a resource out we mean generating a HM that occupy that resource, thus making the resource unavailable for user defined modules. Obviously the diagnosis process will be as fine-grained as possible in order to mask out the minimum amount of resources of the FPGA, thus maximizing the amount of resources still available for further use. In this way the flow will allow partially damaged SRAM-based FPGAs deployed in space missions still to be used.

Modern SRAM-based FPGAs allow partial dynamic reconfiguration and partial readback of the configuration memory. We use partial dynamic reconfiguration to place a test HM on the FPGA and to apply the corresponding test patterns. We capture the responses of the HM in memory elements and then we use memory readback to access the response of the HM in order to analyse it and to determine whether the tested reconfigurable module (or fabric area) is fault-free or not. Access to the configuration memory is made available by dedicated IP Cores, such as [16].

The proposed software flow for the generation of the test HMs and the corresponding test patterns, is represented in Figure 1. Figure 2 represents the software flow for the fine grained diagnosing of faults and for the generation of the patch HM used to mask out the faulty resource. In the following these two flows are presented.

### A. Fault Testing

The input of the flow for the generation of the test HMs and the associated test patterns is a file (called *primary input*, see Figure 1) containing either the definition of the module that has to be placed on the FPGA or the coordinates of an area of the FPGA that is intended to be tested.

The first step of the flow is the execution of STAR on the primary input file. The output of the analysis carried out

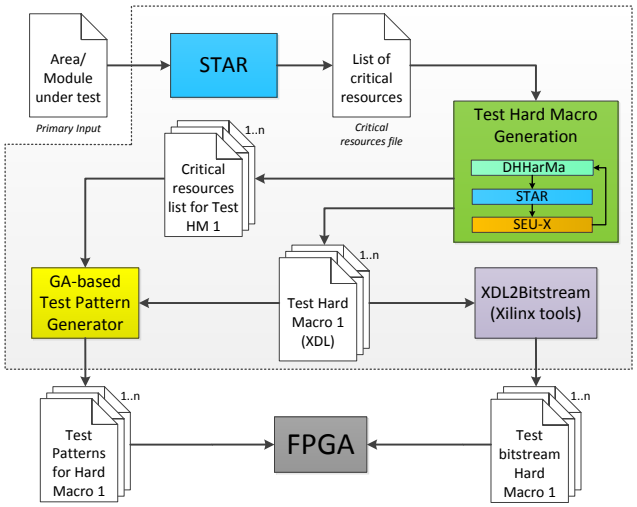



Fig. 1. The software flow for fault testing.

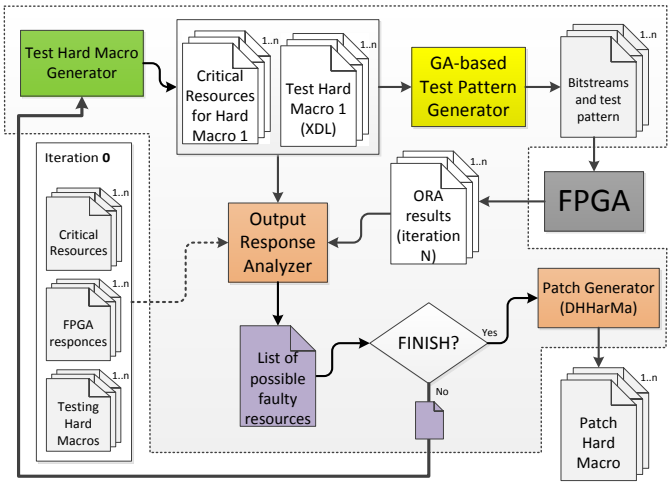



Fig. 2. The software flow for fault diagnosing.

by STAR (called *critical resource file*, see Figure 1) is a file containing all the critical resources for the reconfigurable module (or fabric area) specified in the primary input file. These resources will be the ones addressed by the test flow. It is worth noting that the critical resources for a given reconfigurable module (or fabric area) are not only the ones occupied by the reconfigurable module or included in the fabric area, but also all those adjacent resources whose behaviour directly affects the behaviour of the considered reconfigurable module (or fabric area) [17].

The output file of the STAR tool represents the input file of the *Test Hard Macro Generator* block (see Figure 1). This block performs an iterative execution of the following steps:

1) DHHarMa reads the critical resource file and it generates a list of HMs that use all the critical resources for the considered reconfigurable module (or fabric area).
2) The list of HMs generated by DHHarMa is analysed by STAR , that produces the lists of the critical resources for these HMs. This step checks whether the HMs generated by DHHarMa use all the critical resources specified in the critical resource file. If the generated HMs do not occupy all the critical resources, a *not used critical resource file* (file *F1* in Figure 1), containing the list of the critical resources still not used by the generated HMs is produced.
3) SEU-X is executed on the generated HMs in order to determine whether they are fully testable or not. In particular SEU-X determines which of the critical resources for the considered reconfigurable module (or fabric area) can be tested using the generated HMs. The list of the not testable critical resources is specified in the *not testable critical resource file* (file *F2* in Figure 1).
4) If both the not used and the not testable critical resource files are empty, the generated test HMs can fully test all the critical resources for the considered reconfigurable module (or fabric area), thus the execution of the test HM generator can stop. Otherwise the new critical resource file, containing the union of the not used and the not testable critical resource files, is created and the execution restarts from step 1, to generate test HMs for the missing critical resources.

At the end of the execution of the Test Hard Macro Generator block, possible unnecessary (redundant) test HMs will be dropped from the test HM list.

After the complete list of test HMs has been generated, the HMs themselves, together with the associated used critical resources, are given in input to the GA-based test pattern generator. This tool is intended to generate an efficient set of test patterns for every given test HM.

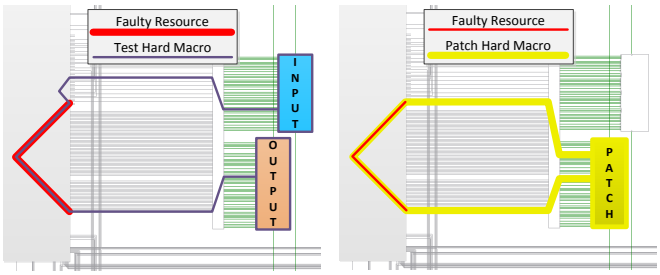
The list of the test HMs, and the associated test patterns represent the output of the fault testing flow. The test HMs are specified in XDL, so they can be translated into an NCD description and then into a bitstream.

We point out that after a test HM has been placed, the associated test patterns will be applied through partial dynamic reconfiguration of one or more LUTs included in the HM itself. The output of the test HM will then be captured into memory elements that will then be readback and analysed in order to discover whether the behaviour of the HM was correct or not. Figure 3(a) shows an example of test HM (marked in blue) for a faulty programmable interconnect point (marked in red).

### B. Fine-Grained Fault Diagnosing

After a faulty response of the system during the test phase, the fine-grained fault diagnosing and patching flow can be executed (see Figure 2). This flow is intended to identify in a fine-grained way the faulty resources and then to generate a set of patch HMs used to mask out the faulty resources. In this way partially faulty regions of the FPGA could still be available for further use.

The fine-grained fault diagnosing and patching flow can be divided into an iterative flow (the *diagnosis flow*) and the *patching flow*. The diagnosis flow starts from the output of the testing phase (the test HMs and the associated output responses). Having these files (and knowing the critical resources covered by each test HM) the *Output Response*




(a) A test hard macro (b) A patching hard macro

Fig. 3. Examples of test/patching hard macros.

*Analyzer* block builds a list of possible faulty resources. Starting from this point the following steps are executed:

1) If the list of possible faulty resources has reached the desired level of granularity, the iterative execution ends. Otherwise step 2 is executed.
2) The test HM generator is executed to generate a set of test HMs for the remaining possible faulty resources (called *diagnostic HMs*). It is worth noting that it is impossible to generate a HM using just one resource, because it would represent an inconsistent circuit, and thus it would be impossible to place it on the FPGA. Because of this, multiple diagnostic HMs will be generated, each using the possible faulty resource, but also some unfaulty resources. It is vital for the diagnosis process that all the generated diagnostic HMs use the considered possible faulty resource, and that each uses different unfaulty resources.
3) The GA-based test pattern generator is executed in order to generate test patterns for the diagnostic HMs.
4) The diagnostic HMs are placed on the faulty FPGA and the corresponding test patterns are applied using partial dynamic reconfiguration
5) By analysing the responses of the diagnostic HMs, the Output Response Analyzer block builds the new list of possible faulty resources and the execution of the flow restarts from step 1.

After the desired diagnostic granularity has been reached, the remaining possible faulty resources should be made unavailable for further uses. This allows partially damaged FPGAs still to be used. It is desirable that just the faulty resources of the FPGA are made unavailable.

A way to exclude resources from the place-and-route is generating patching HMs intended to mask out the undesired resources. Unfortunately in order to integrate these patching HMs in the user design, the HMs will also involve a given amount of unfaulty resources, that will be wasted. Therefore it is important that patching HMs will use the minimum amount of unfaulty resources. In the proposed software flow the Patch Generator block (shown in Figure 2) is in charge of generating patching HMs taking into account the considerations discussed above. Figure 3(b) shows an example of patching HM, where the patching HM is marked in yellow and the faulty resource is marked in red. It can be noticed that more resources than the faulty one are involved in the patch.

## IV. Conclusions

We have reported about the development of a flow for the generation of HMs for testing permanent faults caused by long term exposure of FPGAs to radiation, as it is the case in space missions. The flow is hierarchical, and it is composed by two steps. The first step aims at quickly online discovering whether the FPGA is working properly or not. If a faulty behaviour has been detected, the second step is executed in order to locate the faulty resources. Finally a patch HM occupying the faulty resources (and thus masking them out) can be generated, thus making partially faulty regions of the FPGA still available for further use.

## References

[1] J. Hagemeyer, A. Hilgenstein, D. Jungewelter, D. Cozzi, C. Felicetti, U. Rückert, S Korf, M. Köster, F. Margaglia, M. Porrmann, F. Dittmann, M. Ditze, J. Harris, L. Sterpone, J. Ilstad, "A Scalable Platform for Run-time Reconfigurable Satellite Payload Processing," in *Proc. of the NASA/ESA Conference on Adaptive Hardware and Systems*, 2012.

[2] R. Ferguson and R. Tate, "Use of field programmable gate array technology in future space avionics," in *Proc. of the 24th Digital Avionics Systems Conference*, 2005.

[3] R. Baumann, "Radiation-induced Soft Errors in Advanced Semiconductor Technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305 – 316, September 2005.

[4] J. Wang, "Radiation effects in FPGAs," in *Proc. of the 9th Workshop on Electronics for LHC Experiments*, 2003.

[5] W. Huang, F. Meyer, N. Park, and F. Lombardi, "Testing Memory Modules in SRAM-based Configurable FPGAs," in *Proc. of the International Workshop on Memory Technology, Design and Testing*, 1997.

[6] M. Renovell, J. Portal, J. Figuras, and Y. Zorian, "Minimizing the Number of Test Configurations for Different FPGA Families," in *Proc. of the Eighth Asian Test Symposium*, 1999.

[7] J. Smith, T. Xia, and C. Stroud, "An Automated BIST Architecture for Testing and Diagnosing FPGA Interconnect Faults," *Journal of Electronic Testing*, vol. 22, pp. 239–253, 2006.

[8] M. Abramovici and C. Stroud, "BIST-Based Test and Diagnosis of FPGA Logic Blocks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 1, pp. 159 –172, 2001.

[9] M. Rozkovec, J. Jenicek, and O. Novak, "Application Dependent FPGA Testing Method," in *Proc. of the 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, 2010.

[10] M. Tahoori, "Application-Dependent Testing of FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 9, pp. 1024 –1033, 2006.

[11] C. Bernardeschi, L. Cassano, M. Cimino, and A. Domenici, "Application of a genetic algorithm for testing seus in sram-fpga systems," in *Proc. of the 6th HiPEAC Workshop on Reconfigurable Computing*, 2012.

[12] M. Abdelfattah, L. Bauer, C. Braun, M. Imhof, M. Kochte, H. Zhang, J. Henkel, and H.-J. Wunderlich, "Transparent Structural Online Test for Reconfigurable Systems," in *Proc. of the 18th IEEE International On-Line Testing Symposium*, 2012.

[13] L. Sterpone and M. Violante, "A new analytical approach to estimate the effects of seus in tmr architectures implemented through sram-based fpgas," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2217 – 2223, December 2005.

[14] S. Korf, D. Cozzi, M. Köster, J. Hagemeyer, M. Porrmann, U. Rückert, and M. Santambrogio, "Automatic HDL-Based Generation of Homogeneous Hard Macros for FPGAs," in *Proc. of the IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines*, 2011.

[15] C. Bernardeschi, L. Cassano, and A. Domenici, "SEU-X: a SEu UnuXecitbility prover for SRAM-FPGAs," in *Proc. of the 18th IEEE International On-Line Testing Symposium*, 2012.

[16] *LogiCORE IP XPS HWICAP (v5.00a)*, Xilinx, July 2010.

[17] M. Violante, N. Battezzati, and L. Sterpone, *Reconfigurable Field Programmable Gate Arrays for Mission-Critical Applications*. Springer Science & Business Media, 2011.