Fast SRAM-FPGA Fault Injection Platform based On Dynamic Partial Reconfiguration

Fakhreddine Ghaffari* ghaffari@ensea.fr
Fouad Sahraoui* fouad.sahraoui@ensea.fr
Mohamed El Amine Benkhelifa* benkhelifa@ensea.fr
Bertrand Granado‡ bertrand.granado@upmc.fr
Marc Alexandre Kacou* assi-marc.kacou@ensea.fr
Olivier Romain* olivier.romain@ensea.fr

*ETIS, CNRS UMR 8051, ENSEA, UCP
6 avenue du Ponceau, 95000 Cergy-Pontoise, FRANCE

‡ LIP6, CNRS UMR 7606, UPMC
4 Place Jussieu, 75252 PARIS Cedex 05, FRANCE

Abstract—**SRAM-based FPGAs are very sensitive to harsh conditions, like radiations or ionizations, and need to be hardened to insure correct running. To validate any fault tolerant solution for these SRAM-FPGA, fault injection campaigns must be conducted carefully. In this work, we present a new design flow to perform localized internal fault injection on specific parts of a Design Under Test (DUT). To achieve this, we combine between 1) Partial Dynamic Reconfiguration (PDR) via Internal Configuration Access Port (ICAP) for rapid fault insertion on SRAM; 2) Isolation Design Flow (IDF) to isolate both of placement and routing of Design Under Test into a specific partial region.**

Moreover, we applied realistic fault distribution laws deduced from ground-based radiation experiments to reflect realistic behavior of FPGA toward radiations. The implemented injection platform using this flow shows the importance of using distribution laws driven approach. Results show that our fault injection experiments are done more than 15 times faster than one of the traditional FPGA based fault injection methods with a speed-up on simulation time up to 8.

Keywords: Fault Injection, Fault Emulation, Fault Tolerance, Module Isolation, Fault Distribution Law.

I. Introduction

In the last years, SRAM-based Field Programmable Gate Arrays (FPGAs) have made a significant progress by advancing from prototyping platform to execution ones, which comes mainly from their attractive features, like high available resources, high speed of execution and complete/partial reconfiguration capability. The use of SRAM technology to store the configuration data, called bitstream, was also a key feature for this advance. Despite this progress, SRAM-based FPGAs are very sensitive to fault occurrences and halt their wide use in aerospace domain. This weakness is especially true at configuration memory which represents more than 80% of total memory (20% is BRAM) and can lead to erroneous and critical execution and incorrect results.

To cope with this weakness, many Fault Tolerant (FT) methods emerged to enhance the reliability of FPGA-based systems [1]. A great number of those methods are based on redundancy, like Triple Modular Redundancy (TMR) [2] or Duplication with Comparison (DWC) [3] or their derivations. In these systems more resources are needed, either they are tripled for TMR approach or duplicated for DWC and voters are added to validate intermediate/final results. Other methods aim to avoid using redundancy, which impose a huge overhead and may be unsuitable to apply on large design. This other methods take advantage of Partial Dynamic Reconfiguration (PDR) feature, like for example Configuration Scrubbing [4], where a golden bitstream is periodically written to SRAM memory to eliminate any eventual bit upset occurrence. Error Detection and Correction Codes (EDAC) can also be a solution to minimize redundancy overheads. Hardware checkpoint and recovery [5] can also be used to protect computations against faults occurrence and errors propagation with an acceptable time overhead.

FT methods is incorporated into the considered system and fault injection is carried to evaluate the robustness of the used FT. To achieve a high level of fault injection accuracy, one can put the system in real conditions of execution [6]. However, approaches like the one described in [6] are often hard to control, always time consuming and very expensive. New approaches need to be elaborated to achieve similar objectives with lower complexity and lower costs.

Emulation via Dynamic Partial Reconfiguration [7] has been proven to be an accurate approach to perform fault injection evaluation at early stage of system design and its validation against faults. Fault injection via emulation isn't a new method for reliability assessment [8] but is still limited by closed bitstream format which prevents in deep and accurate identification of specific parts of the design to be injected. In this paper, we propose a relevant way to enhance classical design flow of fault injection by isolating parts of interest in individual partitions at early stage of system design. Isolation does not only allow constraining functional elements as compared to PDR, but also constrains wires and matrices used by these parts of design into specific known regions of SRAM configuration layer.

The rest of this paper is structured as follows. In Section II, a discussion of fault injection methods and its related works are presented. In section III, Isolation Design Flow benefits and its integration in fault injection via emulation are presented. Section IV depicts rapidly implementation details and shows the importance of self-injection problem, especially in routing resources, which are easily overcome by our Isolation based Fault Emulation (IFE). Section V concludes this work and enumerates some perspectives.

II. Related Works

Depending on the desired behavior to observe and to validate, different methods are available to perform fault injection into SRAM-based FPGAs. They can be grouped into: HDL simulation based methods, FPGA emulation based methods, accelerated particles or radiation based methods. Each one provides a certain number of advantages and drawbacks which need to be considered before using it [9].

With High Description Language (HDL) simulation, it is possible to evaluate hardened design without the need to consider the internal behavior of the target FPGA. Such methods are very effective at early stage of prototyping and conception; they can pinpoint rapidly weakness in applied FT. Despite the high level of abstraction and the precision of fault location offered by HDL simulation, some drawbacks have to be considered when using such approach. A huge temporal cost for the simulation step, which can grow rapidly with the complexity of the DUT or the set of fault combinations to be injected. Moreover, HDL simulation is non-adapted to perform black box simulation where generation of its stimulus can be a serious problem. In [10], FuSE tries to overcome the temporal cost by accelerating simulation step with SEmulator software and a dedicated FPGA platform, which can be an additional cost to consider. HDL simulation methods generate a huge quantity of data to analyze.

As opposite to HDL simulation methods, ground-based radiation methods are used to reproduce pseudo-realistic harsh conditions of execution. They are very suitable to enumerate possible fault models and to evaluate space readiness of new SRAM-based FGPAs. However, ground-based radiation method [11] is a very expensive approach and requires a specialized infrastructure. This fault injection method also suffers from a large interval of time to perform irradiation and it is totally uncontrollable regarding fault location and fault model.

Emulated based fault injection methods can be more suitable junction between the advantages of previous described methods. By applying a bitstream alteration, it is possible to recreate effects of radiation or ionization, and by executing design directly on FPGA target, it is possible to validate the behavior of FT in conjunction with FPGA behavior at runtime execution.

Approaches using Partial Dynamic Reconfiguration (PDR) to perform bit upset at runtime offers more complex scenarios. PDR can be achieved either from a PC or another FPGA via external reconfiguration port [8], like Select Map or JTAG, and it allows the validation of the DUT with its real routing and placement on the chip. PDR can be performed via Internal Configuration Access port (ICAP) to reach high rate of fault insertion and removal, which can be a key feature to emulate advanced scenarios and realistic fault rate.

The additional step of isolation proposed in this work gives a new way to localize the resources to be injected, especially routing wires/switches. Another benefits of using our approach is to separate between the logics/routes of the DUT from those belonging to the injector or between different instances in the case of TMR for example.

III. Isolation Based Fault Emulation - IFE

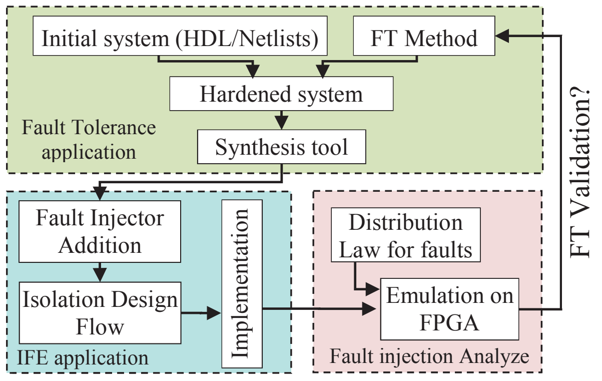

Fig. 1: FT Validation using Isolation based Fault Emulation.

Figure 1 presents an illustration of FT validation using our Isolation based Fault Emulation flow. After an application of the FT method upon an initial system, synthesis is performed and later a fault Injector design is added. Isolation Design Flow is applied to separate between the DUT and the rest of the design. Finally emulation is performed using a fault distribution law to validate the FT robustness by checking DUT computation results.

A. *Resources and Configuration layer structure*

Following explanations use Virtex-5 FPGA as an illustration device but doesn't restrict the proposed flow to it.

Resource layer in SRAM-based FPGAs is generally structured into three main blocks, which are: Configurable logic blocks (CLB) used to implement Boolean functions, routing matrices used to interconnect logic blocks and input/output blocks allow communication with external world. In recent devices, there are also embedded resources on the die, like Block RAM or DSPs, which aim to accelerate some specific part of implemented modules. Each column of resource is configured with a specific number of frames stored on the configuration layer (SRAM). A frame is a vector of (1, n)-bits (n is specific to the device). For example, in Virtex-5 a frame has n=1312 bits height and 36 frames are needed to totally configure a column of configurable logic blocks (CLB).

To perform fault injection by emulation, frames are read from the configuration layer, altered by inverting one or more bits and written back to the configuration layer [8].

B. Isolation Design Flow in IFE

Xilinx Isolation Design Flow (IDF) [12], is a specialized flow used to build high dependable design for critical system like cryptography, where attributes like confidentiality, availability, safety, reliability, maintainability and integrity are very required. Especially, this flow is used on system with redundancy, where it is helpful to conduct fault confinement, analyze of redundant instance, fault removal and correction. For each isolated partition, an ”SCC_ISOLATION” attribute is added to notice the Placement and Routing tool (PAR) that these partition need special care when performing their placement and routing. An unused column of CLB resource, called Fence, must be inserted vertically or horizontally between each isolated partitions and the Top design partition, like illustrated in Figure 2.

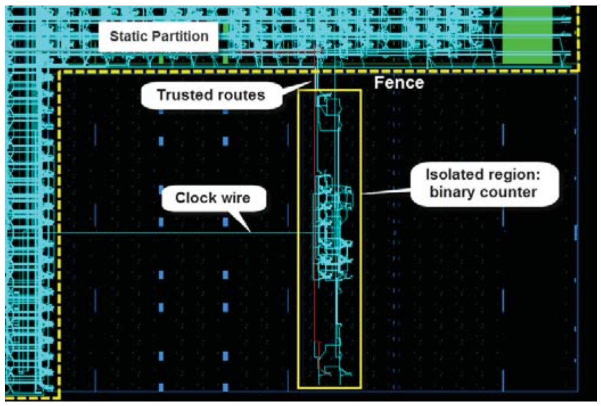

Fig. 2: Binary counter isolated with Xilinx Isolated Design flow.

Figure 2 shows also that all routing are localized on their respective partition, except for connections that link the static partition to the isolated partition, which are performed via trusted routes [13]. This flow insures a correct execution of the embedded fault injector when using ICAP.

C. Fault Distribution Law

Fault injection by emulation can be conducted either using a random law or following a realistic distribution law. The random approach is more generic and can be used to rapidly validate preliminary designs, while the second approach provides more realistic scenarios and emulates more complicated faults that are uncovered in the first one, like for example multiple bit upset and their shape.
Combining our IFE flow with a fault distribution law, we are able to emulate ground-based fault injection experiments upon isolated partition, with a very low cost and at high speed. Table I illustrates an example of a fault distribution law of a Virtex 5 configuration layer deduced from [11]. This example outlines that actual FPGAs are subject to complex scenarios, where using random law to choose fault location and size, may fail to evaluate correctly a given FT method.

TABLE I: Example of fault distribution law deduced from [11].

LETs & Angles		68.3 $(MeV - cm^2/mg)$ $(0°, 0°)$		72.8 $(MeV - cm^2/mg)$ $(90°, 60°)$	
SBU	1-bit	41%		28%	
MBU	2-bits	59%	34%	72%	23%
	3-bits		13%		15%
	4-bits		12%		18%
	5-bits		-%		7%
	6-bits		-%		9%

D. Fault Generator

To be able to reproduce fairly a distribution law and its involved scenarios, it is important to have a fault generator algorithm which is capable of determining for each generated fault a number of properties, like size, location and type of resource involved, shape in case of multiple bits upset (MBU), and time of its insertion. We constructed fault generator to perform such choices where it takes those fault distribution law tables as inputs to drive its operations.

In our experiment, both generation and insertion of faults were performed with an embedded injector. This helped the validation the absence of fault injection into injector logics (self-injection problem) when using IFE approach. An important step in this process is MBU generation. It is crucial to fairly reproduce the behavior of the relationship between bits involved into an MBU, and the way to choose frames that are hit by MBU. A solution to achieve this objective is to keep track of different shapes of collected MBUs when characterizing an FPGA and their corresponding probability of occurrence. Table II presents an example of such information. A shape is described by the smallest matrix (NxM) which bounds the MBU bits.

TABLE II: Fault distribution law of shape for MBU on Virtex 5 FPGA.

MBU	2-bits			3-bits	4-bits	
Shape	(1x2)	(2x1)	(2x2)	(3x1)	(2x2)	(3x2)
Heavy ion	21%	56%	7%	5%	3%	3%
Proton	50%	32%	-	8%	10%	

IV. Implementation and Results

To outline benefits of isolation design flow for fault injection with emulation, we first implemented a MicroBlaze-based system that runs the previous presented fault generator. We placed some functional logics inside partial region as DUT. The system perform fault injection via Xilinx SEU controller [14], and reports results for later analyze through serial port to a host PC.

Table III shows a timing performance comparison of two fault injection methods for configuration memory of SRAM-based Xilinx FPGAs. We note that the injection with SEU controller is much faster. It allows an injection of a fault every 14 microseconds. Thus, with this method we can emulate all radioactive environments that have rate of fault arrival less or equal to $71.43kHz$. Moreover, preliminary results from experiment using traditional fault emulation approach demonstrate that the injector is susceptible to self-injection problem, even when targeting the DUT configuration memory.

TABLE III:
Fault Injection Time (= Read + Write of one frame).

Injection Method	READ Frame	WRITE frame	Injection time	**Freq Max**
HWICAP Miroblaze (100 MHz)	0.82 (ms)	1.01 (ms)	1.83 (ms)	546,44 Hz
SEU CTRL Picoblaze (100 MHz)	7 (μs)	7 (μs)	14 (μs)	**71,43 KHz**

While using our Isolated Fault Emulation, it is possible to performed fault insertion into any frame inside the DUT routing and CLB without harming the functionality of our internal injector design. Furthermore, the application of Xilinx Isolation design flow doesn't influence the maximum frequency of neither the design under test nor the fault injector. We validate our Fault injection tool on three different workloads: one simple (64 bits counter), a more complex (Bubble sort) and finally a PicoBlaze Processor executing an AES algorithm. As it is shown in table IV, we reached to speed-up more than 15 times in our experiments comparing with existing approaches.

TABLE IV:
Fault Injection Time (= Read + Write of one frame).

DUT	# frames	Traditional Approach [15] external 1 injection = 62.5	Fast Approach [16] internal 1 injection = 218.75	Our Method 1 injection = 14μs
Counter	36	2250 ms	7875 μs	504 μs
Bubble Sort	144	9000 ms	31500 μs	2016 μs
Picoblaze AES	324	20250 ms	70875 μs	4536 μs

Finally, we compared in Table V the simulation time required to validate an approach of reliability by injecting faults with an exhaustive approach or following a realistic fault model. The speed-up of simulation time is very significant and may in some cases exceed 8 times.

TABLE V: Simulation Time Speed-Up.

Injected Errors	Exhaustive Model	Our Model (for Specific Environment) Heavy ion 68.3 MeV-cm^2/mg 0.78 upset/ms (Freq=777.8Hz)	Speed-up
1 bit	36.73 ms	41% = 15.05 ms	2.43
2 bits	12.04 s	34% = 4.09 s	2.94
3 bits	1.46 Hours	13% = 683.28 s	7.7
4 bits	19.91	12% = 2.38 days	8.33

V. Conclusion and Future work

We proposed in this work for fault injection emulation by taking advantage of new capabilities of placement and routing tools. Isolation of the design under test is applied to separate the fault injector logics/routing from the DUT's one. Our injection method eliminates any risk of self-injection of faults inside the injector design (especially routing) which can be an issue for correctness of fault validation results. We show the importance of keeping track of multiple bit upset shapes, especially for newer FPGAs. This information is useful for the reproduction of complicated fault scenario, which can be very difficult to predict fairly and is important to consider even with smaller probability.

Acknowledgements

The authors acknowledge the support of General Council of Val d'Oise (CGVO) and the urban community (CA) of Cergy-Pontoise, under Grant Agreement number 14RETID382 (EXELA-VO project).

References

[1] Jason A. Cheatham, John M. Emmert, and Stan Baumgart. A survey of fault tolerant methodologies for fpgas. *ACM Trans. Des. Autom. Electron. Syst.*, 11(2):501–533, 2006.

[2] C. C. Xapp. Triple modular redundancy design techniques for virtex fpgas, 2006.

[3] J. Johnson, W. Howes, M. Wirthlin, D. L. McMurtrey, M. Caffrey, P. Graham, and K. Morgan. Using duplication with compare for on-line error detection in fpga-based designs. In *Aerospace Conference, 2008 IEEE*, pages 1–11.

[4] M. Lanuzza, P. Zicari, F. Frustaci, S. Perri, and P. Corsonello. Exploiting self-reconfiguration capability to improve sram-based fpga robustness in space and avionics applications. *ACM Trans. Reconfigurable Technol. Syst.*, 4(1):1–22, 2010.

[5] F. Sahraoui, F. Ghaffari, M. El Amine Benkhelifa, and B. Granado. An efficient ber-based reliability method for sram-based fpga. In *Design and Test Symposium (IDT), 2013 8th International*, pages 1–6.

[6] Heather Marie Quinn, Paul S Graham, Keith S Morgan, Zachary K Baker, Michael P Caffrey, David A Smith, Mike Wirthlin, and Randy Bell. Flight experience of the xilinx virtex-4. *Nuclear Science, IEEE Transactions on*, PP(99):1–9, 2013.

[7] R. Velazco, G. Foucard, and P. Peronnard. Combining results of accelerated radiation tests and fault injections to predict the error rate of an application implemented in sram-based fpgas. *Nuclear Science, IEEE Transactions on*, 57(6):3500–3505, 2010.

[8] U. Legat, A. Biasizzo, and F. Novak. Automated seu fault emulation using partial fpga reconfiguration. In *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*, pages 24–27. [same as us] for fault emulation + [cat4] [printed].

[9] H. Quinn, P. Graham, K. Morgan, M. Caffrey, and J. Krone. A test methodology for determining space-readiness of xilinx sram-based fpga designs. In *AUTOTESTCON, 2008 IEEE*, pages 252–258.

[10] M. Jeitler, M. Delvai, and S. Reichor. Fuse - a hardware accelerated hdl fault injection tool. In *Programmable Logic, 2009. SPL. 5th Southern Conference on*, pages 89–94.

[11] H. Quinn, K. Morgan, P. Graham, J. Krone, and M. Caffrey. Static proton and heavy ion testing of the xilinx virtex-5 device. In *Radiation Effects Data Workshop, 2007 IEEE*, volume 0, pages 177–184.

[12] John D. Corbett. The xilinx isolation design flow for fault-tolerant systems, 2012.

[13] L. Gantel, M. E. A. Benkhelifa, F. Lemonnier, and F. Verdier. Module relocation in heterogeneous reconfigurable systems-on-chip using the xilinx isolation design flow, 5-7 Dec. 2012 2012.

[14] Ken Chapman. Seu strategies for virtex-5 devices, 2010.

[15] M. Shokrolah-Shirazi and S. G. Miremadi. Fpga-based fault injection into synthesizable verilog hdl models. In *Secure System Integration and Reliability Improvement, 2008. SSIRI '08. Second International Conference on*, pages 143–149.

[16] M. S. Shirazi, B. Morris, and H. Selvaraj. Fast fpga-based fault injection tool for embedded processors. In *Quality Electronic Design (ISQED), 2013 14th International Symposium on*, pages 476–480.