Cost effective FPGA probabilistic fault emulation

O. Boncalo, A. Amaricai
Computer Engineering Department
University Politehnica Timisoara
Timisoara, Romania
{oana.boncalo, alexandru.amaricai}@cs.upt.ro

C. Spagnol, E. Popovici
Department of Electrical and Electronic Engineering
University College Cork,
Ireland
{christian.spagnol@ue.ucc.ie,e.popovici@ucc.ie}

***Abstract*—This paper presents a cost effective FPGA fault emulation technique for probabilistic errors. The problem it addresses is how to efficiently inject faults in many locations within a circuit under test. For this purpose, the emulated fault injection (EFI) components proposed are a trade-off between the desire for speed/performance and the inherent physical device limitations of the FPGA. The proposed method also allows exploring the best option for this trade-off with minimal effort. The proposed solution allows enough flexibility to be able to deal with the different EFI architectures selectable by minor code intervention. An analysis of the overhead introduced by EFI components when varying the number of fault locations has been provided. Furthermore, this paper presents a case study of two ISCAS benchmark circuits in order to test our methodologies and to highlight the differences for combinatorial and a sequential circuits. It is shown that the number of fault locations can be increased more than 20 times with similar overhead than other state of the art methods reported in the literature.**

Keywords—FPGA; fault emulation; probabilistic noise; sub-powered circuits;

I. INTRODUCTION

Power consumption represents one of the major issues in current and future semiconductor devices. One of the most radical approaches in tackling the power issues is represented by the aggressive supply voltage scaling that forces the components to work on near and sub threshold regions of operation [1]. A major drawback of supply voltage reduction is represented by the decreased reliability of the semiconductor devices. This is further aggravated by the transistor scaling to deep nano-meter sizes. The decreased resilience to external factors (e.g. temperature, power supply variations) and process variation determines a probabilistic behavior of CMOS logic gates [2][3] For these circuits, the logic gate has the correct output with a probability smaller than 1. The term probabilistic CMOS has been used for circuits which present this type of behavior. The fault model adopted for such devices is represented by the independent probabilistic fault.

In this context, reliability evaluation becomes critical when designing with these circuits. Methods for carrying out this task can be classified as analytical methods, simulation, and FPGA emulation or prototype evaluation. Simulation and FPGA emulation based methods represent a trade-off between the cost and the fault modeling capability. Simulation based methods, on the one hand provide good observability, but have a major drawback: the high number of required simulations [5]. FPGA emulation addresses this shortcoming, at the price of limited observability. The use of FPGA devices to emulate the circuit-under test (CUT) behavior in the presence of faults has been supported by technology advances that brought forward devices with ever-increasing performance and size. Fault emulation of devices with probabilistic fault behavior relies on using random number generator units (RNG), either true (TRNG) or pseudo (PRNG). In a straight-forward approach, a truly uncorrelated probabilistic emulation requires the usage of a RNG for each fault location [5]. If analysis is carried out at gate level, each gate output corresponds to a fault location. This leads to a dramatic increase in the cost of the design for FPGA emulation for medium size circuits.

In this paper, we try to mitigate the problem of the large cost of FPGA fault emulation for probabilistic CMOS gates by employing two schemes: shift register based serial and hybrid serial-parallel units. The proposed schemes use the concept of shift registers in order to load the corresponding fault bits for each fault location [4]. Error bits are generated using a TRNG each clock cycle. In a pure serial implementation, the number of clock cycles required to load the shift register is equal to the number of fault locations in the CUT. In order to reduce the number of clock cycles required for loading the fault bits, hybrid serial-parallel schemes are proposed. The *k*-layer hybrid implementation is obtained by breaking the shift register chain used in serial scheme into *k* TRNG-shift register instances. The proposed schemes (serial and hybrid) present two advantages: (i) reduce cost, because a limited number of TRNGs are used, (ii) true un-correlation. Thus, we present a cost effective way to emulate on FPGA devices probabilistic independent faults.

The contributions of this work are: (i) a cost efficient implementation of serial chains for independent probabilistic faults based on the TRNG choice for gate-level EFI; (ii) hybrid fault activation chains: a tradeoff between area and circuit emulation speed (iii) accurate probabilistic fault emulation by the usage of TRNG for fault generation

The paper is organized as follows: Section II presents related work; Section III is dedicated to the proposed EFI methodology; cost and performance are discussed in Section IV; concluding remarks are found in the last section.

II. RELATED WORK

The topic of EFI has received substantial attention in recent years due to the new FPGA devices which offer better performance and considerable size. Many research studies ([4][6][7][8][9]) have focused on Single Error Upset (SEU) fault model. A key point is represented by the problem of

efficient observation and classification of SEU effects [1][7][8]. These approaches require the modification of the circuit under test (CUT). Three types of components are added to the CUT: the fault injectors (which inject faults into specific locations according to their model), result analyzer, and observation circuitry. In some approaches the result analyzer and observation logic communicate with software running on a host PC [7][9], while for others the analysis is ran autonomously on the FPGA [4][8]. The amount of communication varies amongst solutions from only some commands and configuration values from the host PC [1][10] to the entire Fault Injection(FI) campaign being prepared on the PC side and data downloaded to the FPGA [7][9]. Regarding result processing, it can be either realized on the FPGA board (see [4]), or at the cost of more communication overhead it can be computed on the PC (see [7]).

Work addressing probabilistic faults has been reported in [10][5]. First the authors try a straightforward implementation in [5] with many concurrent EFI units (i.e. one for each fault location). The overhead, even for a simple LFSR based PRNG is very large (more than 8000%). However, although it achieves the best performance, this solution is un-tractable for large circuits with high number of fault locations. The drawbacks of using PRNGs are: (i) successive values are correlated due to pseudo- randomness (ii) requires additional overhead for seed initialization. In [10], in order to serialize the generation of faults, the authors add a serial binomial generator to the PRNG. It outputs a random number every 32 cycles. The fault model studied is SEU. The time analysis presented in [10] indicates that one probabilistic fault is generated for every simulation experiment. Two clock domains are used, which would typically require additional synchronization logic.

III. PROBABILISTIC EMULATED FAULT INJECTION

A. EFI Framework

The proposed EFI approach addresses the injection of probabilistic faults in many locations (several thousands) by the usage of a low cost infrastructure. Furthermore, we target truly un-correlated fault generation and insertion. Figure 1 depicts the developed EFI infrastructure. It consists of: *(i) EFI Fault Generator and Control* – the role of this module is to generate fault bits and to insert them in the corresponding fault locations; *(ii) Autonomous testbench* – the role of this module is to provide the test vectors, the error-free outputs and the result processing *(iii) Observation logic* – this module allows reliability metrics monitoring, as well as parameter changes for several EFI campaigns.

The observation logic is based on the Xilinx Chipscope Pro logic analyzer [11]. The dedicated cores for this module are: ILA – Integrated Logic Analyzer – which allows signal observation and triggers; ICON – Integrated CONtroller – which provides the communication between JTAG interface and the ILA core; and VIO – Virtual Input/Output – which provides the interface to monitor and to drive signal from the testbench.

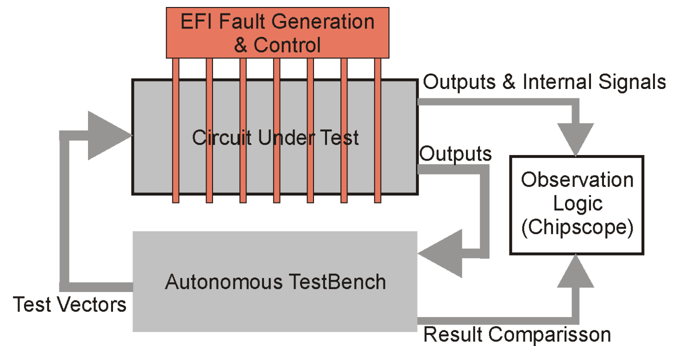

Figure 1 – Proposed EFI Architecture

The main reason behind the Chipscope usage is that it provides cores for monitoring, triggering and driving signals that are optimal in terms of cost and performance for the Xilinx FPGA devices.

The autonomous testbench provides the input vectors for the CUT, the correct outputs (which will be compared with the outputs of the fault injected CUT) and the EFI campaign control. Regarding the result comparison and analysis, three approaches may be used: *(i) implementing the error-free version of CUT and running in parallel with the injected version* [5] (*ii) duplicating the sequential elements in the design* [6] (iii) *storing correct outputs in either FPGA block RAM or external memory (the board memory)*. In our FPGA EFI framework, we have used the memory based solution.

B. EFI Architectures

In this paper we have identified two types of fault locations: combinational logic gates and register/memory elements. The behavior of a faulty gate is modeled with probabilistic fault activation logic at the output. A faulty memory element is modeled by a probabilistic fault manifesting at the data input line of a correct component. This way, the erroneous state is latched. The solution for fault activation on a line is based on a XOR gate with two inputs. Two distinct actions have been identified: (1) fault activation bit generation and propagation to the desired location, and (2) actual fault manifestation by means of the XOR gate. For a CUT, a dedicated EFI control signal is used to freeze normal execution during one emulation cycle, until all fault activation bits are in place.

We have implemented 3 architectures: parallel, serial and hybrid serial-parallel. The parallel approach is straightforward and has also been implemented in [5]. The main drawbacks of this are: (i) high overhead when dealing with a large number of fault locations (ii) the use of LFSR as RNG leads to correlations between successive errors (iii) seed number generation is required in order to avoid correlation between fault locations. However, TRNGs are more expensive than LFSR based solutions. Furthermore, even if LFSRs are employed, having one such element for each location, when discussing thousands of locations ads consistent averhead as presented in the analysis from the results section.

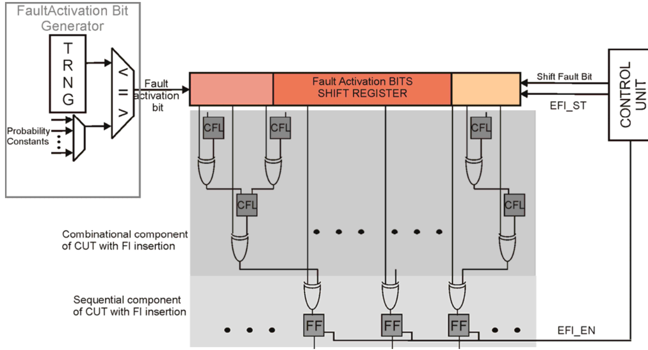

Figure 2 – Serial Implementation of EFI
(CFL – combinational fault location; FF – flip-flop)

For better cost, we have used serial generation of fault activation bits that are propagated through a shift register to their designated locations (Fig. 2). Because we target true un-correlations between errors, we have used TRNG. Due to the area usage constraints and good randomness we have selected for our implementation the solution proposed in [12] by Xilinx, which has passed most of the DieHard randomness tests used for cryptographic applications. An EFI layer component is built of 1 TRNG, one comparator, multiplexing logic and shift register. The built EFI layer architectures layers for generating fault activation bits come in two flavors based on their configurability. The simplest has the same fault occurrence probability for all locations. The number of fault locations is a configuration parameter, while the threshold value is a 32 bit input port that is driven by the testbench. The second version, allows different fault occurrence probabilities for different locations. It has several configuration parameters that states the number of different fault activation probabilities and the number of locations with a given fault probability. The component interface has an input signal driven from the testbench that specifies the threshold value corresponding to each fault probability. The main disadvantage of the serial EFI is represented by the large number of clock cycles required to load the shift register, equal to the number of fault locations.

In order to improve emulation performance, we have implemented a hybrid version for EFI (Fig. 3). It consists of *k* versions of serial TRNG – shift register modules. The length of the shift register is reduced by a factor *k*. Therefore, the reduction of the number of clock cycles required for loading the shift registers is proportional with the number of layers. This increase in performance is obtained at the cost of increase of the total area consumption since a larger number of TRNGs is required, while the total number of the shift registers flip-flops remains equal to the one used in serial version.

IV. Results

A. Performance analysis

In this sub-section we will present the performance analysis for the proposed methodology. Emulation time is given by :

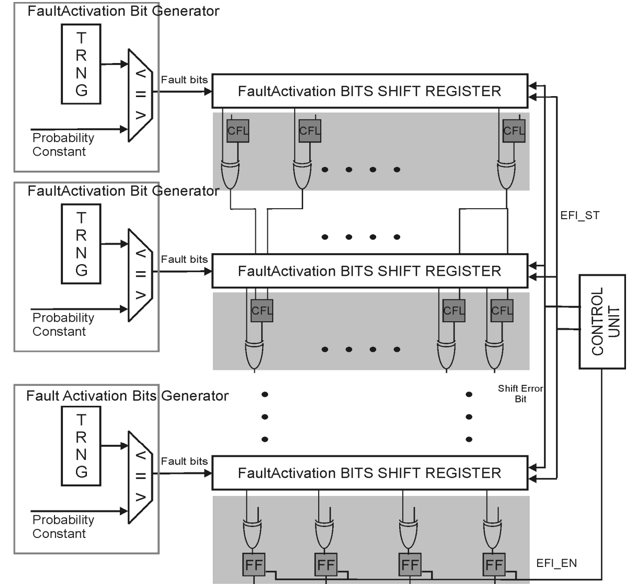

Figure 3 – Multi-layer Hybrid Implementation of EFI

$$T_{EFI} = T_{init} + N_{EFI-Phases} * \left(T_{control} + T_{fault-insert} + T_{run} \right) \quad (1)$$

T_{init} represents the time required for initializations before an EFI campaign; $N_{EFI-Phases}$ represent the number of emulation cycles; $T_{control}$ and $T_{fault-insertion}$ are the times when the CUT operation is frozen for error generation and insertion; T_{run} represents the time for one error emulation run. The duration of one EFI phase is presented in Fig 4. The number of clock cycles required to load the error bits represents the most important component in the time required for one EFI phase. In a full serial implementation, the number of clock cycles required to load the shift register equals the number of fault locations. In hybrid implementations consisting of *k* parallel TRNG – shift registers configurations, significant reduction of these clock cycles is obtained.

B. Case studies

The proposed EFI architectures have been used for analyzing probabilistic behavior of two ISCAS 85 and 89 benchmarks circuits: *c499* (combinational) [13] and *s1196* (sequential) [14]. Table I presents the post place-and-route area estimates for the two circuits. The EFI modules for these two circuits have been implemented on a Digilent Genesys board with Xilinx Virtex-5 FPGA. The synthesis and implementation process has been performed using Xilinx ISE 14.4 software.

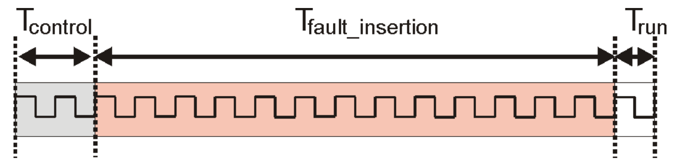

Figure 5 – Emulation Phase

TABLE I – IMPLEMENTATION RESULTS FOR *C499* AND *S1196*

	c499 (LUT-FF pairs)	*s1196* (LUT-FF pairs)	Overhead *(c499)*	Overhead *(s1196)*
Serial	1070	1588	1750%	1185%
Hybrid	1145	2361	1877%	1761%
Parallel	9281	14057	15200%	10400%
No EFI	61	134	0	0

Regarding the *c499* benchmark circuit, a number of 5 layers for the hybrid implementation have been considered. The maximum number of fault locations for a single layer is 40. The number of fault locations for *c499* is 188. For the *s1196* benchmark circuit, the hybrid implementation consists of 5 layers, with the maximum number of fault locations for a single layer of 119. The number of fault locations for *s1196* is 406. The results show that using the full serial implementation, the overall cost overhead (which includes the EFI Error Generator and Control, the autonomous testbench and the Chipscope based observation logic) is 1700% for the *c499* and 1200% for *s1196* benchmark circuits with respect to the two circuits with no EFI. The 5-layer hybrid implementation has an increased area consumption cost of 6.5 % for the *c499* and 48 % for *s1196* with respect to the full serial. The cost of the fully parallel implemented with 24-bits LFSR is around 9 times higher with respect to the fully serial implementation.

Other probabilistic EFI approaches have been presented in [5][10]. Their approach considers the fault locations only the memory elements (the flip-flops); the fault locations considered in their experiments is 18 for *s1196* both in [5][10].The overhead obtained in their approaches is more than 2000% with respect to the basic circuit. In our cases, both the serial and the hybrid present better cost, especially if we consider that in our EFI implementations for *s1196* the number of fault location is more than 22 times higher with respect to [10]. This difference of the area can be explained by the proposed EFI architectures, as well as our autonomous testbench that drives the simulation, and by the usage of ChipscopePro cores that are optimized for Xilinx technology. Furthermore, our experiments also indicate (as in other FPGA EFI implementations) that the cost of the autonomous benchmark and the observation logic represent an important component in the overall cost of the EFI circuitry.

V. CONCLUSIONS

This paper presents a novel approach for FPGA based EFI, targeting probabilistic circuits. The proposed implementations rely on a TRNG for fault bits generation and a shift register for fault bits insertion to their according fault locations. The main contributions of our EFI framework are: (i) truly un-correlated fault generation, both spatially and timely (ii) low cost infrastructure for fault generation and insertion (iii) cost efficient EFI framework for observation and control by the usage of the Xilinx Chipscope. The most cost-efficient approach is represented by the serial implementation. The drawback of it lies in the high number of clock cycles for fault insertion. In order to tackle the number of large clock cycles required to load the shift register in the serial implementation, we have developed the hybrid serial-parallel approach. This way, we target better performance a *k*-layer hybrid implementation reduces the number of clock cycles for loading the shift registers up to *k* times. Regarding the cost of the proposed approach, we have "inserted" a number of 406 fault locations with respect to the 18 fault locations in [10] for the same area overhead. This represents an increase of more than 22 times in FPGA resource efficiency.

ACKNOWLEDGMENT

This work has been supported by the Seventh Framework Program of European Union under Grant Agreement 309129, project *i-Risc*. We would like to thank Xilinx for the software tools provided through the Xilinx University Program.

REFERENCES

[1] H. Khaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, S. Bokhar "Near Threshold Voltage Design: Opportunities and Challenges" Proc. Design Automation Conference (DAC), 2012, pp. 1153-1158

[2] K. Palem, "Energy aware computing through probabilistic switching: A study of limits," Computers, IEEE Transactions on, vol. 54, no. 9, pp. 1123–1137, 2005

[3] A. Bhanu, M.S.K. Lau, K.V. Ling, V.J. Mooney, A. Singh " A More Precise Model of Noise Based PCMOS Errors" Proceedings 5th Int. Symp. On Electronic Design, Test and Application (DELTA), pp 99-102, 2010

[4] C. López-Ongil, M. García-Valderas, M. Portela-García, L. Entrena, "Autonomous Fault Emulation: A New FPGA-Based Acceleration System for Hardness Evaluation," IEEE Trans. On Nuclear Science, Vol. 54, No. 1, pp. 252-261, Feb. 2007

[5] D. May, W. Stechele, "An FPGA-based Probability-aware Fault Simulator", International Conference on Embedded Computer Systems (SAMOS), 2012.

[6] A. Ejlali, S. G. Miremadi, "Eror propagation analysis using FPGA-based SEU-fault injection," Microelectronics Reliability, vol. 48 pp. 319–328, June 2008.

[7] P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda, and M. Violante, "An FPGA-Based Approach for Speeding-Up Fault Injection Campaigns on Safety-Critical Circuits," Journal of Electronic Testing: Theory and Applications 18, Kluwer Academic Publishers, pp. 261–271, 2002.

[8] M. S. Shirazi, B. Morris, H. Selvaraj, "Fast FPGA-Based Fault Injection Tool for Embedded Processors," 14th Int'l Symposium on Quality Electronic Design, 2013.

[9] M. Sauer, V. Tomashevich, J. M¨uller, M. Lewis, A. Spilla, I. Polian, B. Becker, and W. Burgard, "An FPGA-Based Framework for Run-time Injection and Analysis of Soft Errors in Microprocessors," IEEE 17th International On-Line Testing Symposium, July, 2011.

[10] D. May, W. Stechele, "A resource-efficient probabilistic fault simulator, " 23rd International Conference on Field Programmable Logic and Applications (FPL), September, 2013.

[11] http://www.xilinx.com/tools/cspro.htm. Xilinx ChipScope.

[12] C. Baetoniu "Method and Apparatus for True Random Number Generation" US Patent 7389316, 2008

[13] M. Hansen, H. Yalcin, J. P. Hayes, "Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering," IEEE Design and Test, vol. 16, no. 3, pp. 72-80, 1999

[14] F. Brglez, D. Bryan, K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," Proc. 1989 Int. Symp. on Circuits and Systems (ISCAS), 1989