Microelectronics Reliability 51 (2011) 1459–1463

Contents lists available at ScienceDirect

Microelectronics Reliability

journal homepage: www.elsevier.com/locate/microrel

FIFA: A fault-injection–fault-analysis-based tool for reliability assessment at RTL level

L.A.B. Naviner [a,*], J.-F. Naviner [a], G.G. dos Santos Jr. [a,b,*], E.C. Marques [c], N.M. Paiva Jr. [c]

[a] Institut Télécom/Télécom ParisTech, CNRS-LTCI UMR 514, COMELEC, Paris, France
[b] Électricité de France, EDF R&D, Paris, France
[c] Military Institute of Engineering, Rio de Janeiro, Brazil

ARTICLE INFO

Article history:
Received 30 May 2011
Accepted 17 June 2011
Available online 19 July 2011

ABSTRACT

This paper presents an efficient platform for fault robustness estimation of digital circuits. The proposed platform, named FIFA, was designed as a hardware IP to accelerate the Fault Injection and Fault masking Analysis approach. It supports several fault models as well as single and multiple faults. Synthesis results have shown that the proposed platform can exceed those existent in the literature in terms of area efficiency and performance. In addition, the FIFA platform allows the designer to control complexity and completeness of the analysis process.

1. Introduction

Fault injection has been considered very useful to evaluate the behavior of computing systems in the presence of faults [1]. The basic idea of such approach is to produce or simulate faults during system operation. The outputs of the circuit, both with and without internal faults, are compared in order to evaluate its robustness. Several works related to fault injection methods and tools can be found in the literature [1–4]. The main drawbacks of the reported tools are: the number of simultaneous faults that can be generated, the number of fault models that can be produced or simulated, and the time required to generate such faults. In fact, most of the tools deal with single faults, but not simultaneous multiple faults. Also, they consider transient or permanent faults, but not both. If multiple simultaneous faults are considered, that is, if several gates may fail at a same time t (fault multiplicity $k > 1$), the number of tests for exhaustive analysis in large circuits may become prohibitive.

This work presents a new platform to evaluate the robustness of digital circuits against faults. It was designed as a hardware IP to accelerate the Fault Injection and Fault masking Analysis (FIFA) approach. The proposed platform was implemented on FPGA, and the analysis is performed at register transfer level (RTL). The FIFA platform is fully parameterizable, allowing the designer to adapt it for analysis of practically any digital circuit. In addition, this IP can help the designer to establish efficient trade-offs between cost (time, amount of FPGA resources) and completeness of the analysis. Unlike previous works, the FIFA platform deals with several fault models and no FPGA reconfiguration is necessary to simulate different fault patterns for the same circuit [4].

This paper is organized as follows. Section 2 presents the proposed platform. Section 3 explains how it can be used for reliability assessment, and the obtained results are presented in Section 4. Concluding remarks and future work are discussed in Section 5.

2. FIFA platform

Let be a digital circuit OP for which we want to analyze the robustness against faults. The basic idea behind the proposed platform is to inject faults in OP and observe whether this internal fault injection produces errors in the circuit's output. The fault injection mechanism presented in FIFA platform is based on saboteurs. A saboteur comprises a small set of components which provides the capability to alter the values contained in a circuit node. Therefore, appending such components to the nodes of OP allows the injection of single or multiple faults.

The FIFA platform contains, among other items, two versions of OP: one fault-free (OPREF) and one fault-prone (OPFAULTY). The analysis of the robustness against a specific fault f_1 in OP takes two steps: first, we inject the internal fault f_1 in OPFAULTY by enabling the corresponding saboteur (s); next, we compare the outputs of both circuits in order to detect any mismatch. Notice that this procedure is done considering a given input i for both circuits, OPREF and OPFAULTY. If the injection of f_1 does not modify the circuit's output, we can say that the circuit is robust to such fault, that is, f_1 was masked.

Let us now define a **test configuration** of OP as a couple comprising a given input and a given fault. For a specific set of test configurations, the proposed platform analyses the error masking capabilities of OP, and determines its corresponding **masking coefficient**. The masking coefficient of a circuit represents the number

* Corresponding authors. Address: Institut Télécom/Télécom ParisTech, CNRS-LTCI UMR 514, COMELEC, Paris, France. Tel.: +33 145817661; fax: +33 145804036 (L.A.B. Naviner), tel.: +33 145817333; fax: +33 145804036 (G.G. dos Santos Jr.).

E-mail addresses: lirida.naviner@telecom-paristech.fr (L.A.B. Naviner), junior@telecom-paristech.fr (G.G. dos Santos Jr.).

0026-2714/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.microrel.2011.06.017

of test configurations for which it generates correct outputs. Thus, this coefficient is directly related to the **robustness** of a circuit. In our case, we classify the error masking coefficient according to the number k of simultaneous faults injected. Then, we define c_k as the masking coefficient representing the robustness of OP regarding the occurrence of k simultaneous faults.

2.1. Defining the components of the FIFA platform

Fig. 1 shows the proposed platform which comprises the following modules:

- OPREF – A fault-free version of OP.
- OPFAULTY – A faulty version of the operator under analysis. Programmable saboteurs are appended to the nodes of OP where we would like to inject faults. These programmable saboteurs allow the use of different fault models. The fault models available on this platform are: bit flipping (Single Event Upset or Multiple Bits Upset), stuck-at-0, and stuck-at-1. In order to emulate a fault in a node j, the corresponding saboteur must be activated (see Fig. 2). This is done by a control signal (bit e_j in bus **e**). If $e_j = 0$, the node j is supposed to be fault-free. If $e_j = 1$, the node j is supposed to be faulty. The fault model to be used is selected by the signal $\mathbf{m}[m_1{:}m_0]$.
- STIMULI GENERATION – Generates the data inputs for OPREF and OPFAULTY (bus **x**).
- FAULT INJECTION – Generates the control signals to activate/deactivate the saboteurs in OPFAULTY (bits e_j of bus **e**). This module was implemented according to the work presented in [5]. We took into account the second algorithm presented in this work, which can generate all the possible C_k^N vectors **e** for a given number of simultaneous faults $k_{min} \leqslant k \leqslant k_{max}$. This algorithm is explained in Section 2.3
- MASKING ANALYSIS – Compares the outputs provided by OPREF (bus $\mathbf{y}_{ref}$) and OPFAULTY (bus **y**) in order to evaluate the masking coefficients ($\mathbf{c}_k$ values).
- SUPERVISOR – Manages the communication signals among modules (**m**, *req*, **k**, $\mathbf{k}_{min}$ and $\mathbf{k}_{max}$).

2.2. Explaining the communication signals

The FIFA platform is a synchronous circuit operating on the rising edges of a clock signal (*clk*). In order to better understand the functioning of the proposed platform, it is a prime concern to comprehend the purpose of the communications signals used during operation (see Fig. 1).

Signal $\mathbf{k}_a$ represents the number of simultaneous faults to be used in the current test configuration. Considering an OPFAULTY that contains N saboteurs, the FAULT INJECTION module generates the set of $C_{k_a}^N = \binom{N}{k_a}$ different **e** vectors corresponding to all possible

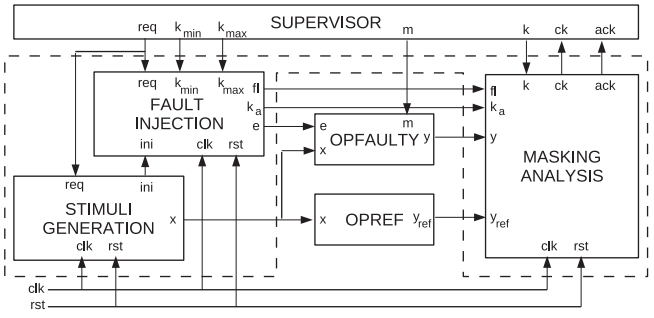

Fig. 1. General scheme of the proposed platform.

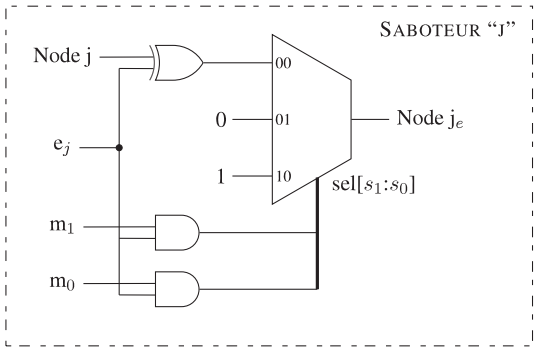

Fig. 2. General scheme of a saboteur (notice that $\mathbf{m} = [1{:}1]$ is not defined).

occurrences of k_a errors. In other words, each vector $\mathbf{e} = [e_{N-1}{:}e_0]$ contains exactly k_a bits at logic value 1 as a means to activate the desired k_a saboteurs. Signals $\mathbf{k}_{min}$ and $\mathbf{k}_{max}$ indicate the minimum and maximum number of simultaneous errors to be considered, respectively. The platform considers fault multiplicity in ascending order. It means that the test configuration starts with $\mathbf{k}_a = \mathbf{k}_{min}$ and concludes with $\mathbf{k}_a = \mathbf{k}_{max}$.

Initialization is done with the asynchronous signal *rst*, which is active at logic level 0. Indeed, when $rst = 0$, signals *fl*,*ack*, and all bits of buses **e**, **x** and $\mathbf{c}_k$ are set to zero.

Signal *req* is used to indicate a reliability analysis request. If $req = 1$, both STIMULI GENERATION and FAULT INJECTION modules are enabled. The first one generates all the possible values of **x** in ascending order: $0 \leqslant \mathbf{x} \leqslant (2^Z - 1)$, where Z represents the width of the bit-vector **x**. The second one generates a bit-vector **e** which is responsible to activate/deactivate the desired saboteurs. When the last **x** value is generated, the STIMULI GENERATION module sends a signal $ini = 1$ to the FAULT INJECTION module. This signal enables the corresponding module to generate the next bit-vector **e** and to reinitialize the bit-vector **x** (i.e. $\mathbf{x} = 0$).

Signal *fl* enables the MASKING ANALYSIS to compare $\mathbf{y}_{ref}$ and **y** in order to evaluate the masking coefficients $\mathbf{c}_k$.

Signal *ack* indicates that the analysis is finished ($ack = 1$), that is, all masking coefficients $\mathbf{c}_k$ are now available. Then, if we want to retrieve a specific $\mathbf{c}_k$ value, we can use the input **k** and the output $\mathbf{c}_k$ from the MASKING ANALYSIS module.

Fig. 3 shows a timing diagram with the platform signals. It considers a circuit in which N nodes may fail. Notice that the MASKING ANALYSIS module requires the activation of both signals, *rst* and *fl*, in order to start its functioning.

2.3. Fault Injection Module

As stated above, the FAULT INJECTION MODULE is responsible to control the activation/deactivation of the saboteurs presented in OPFAULTY. Considering the presence of N saboteurs, each fault pattern is defined by a vector $\mathbf{e} = [e_{N-1}e_{N-2}\cdots e_1e_0]$, where $e_i = 1$ activates

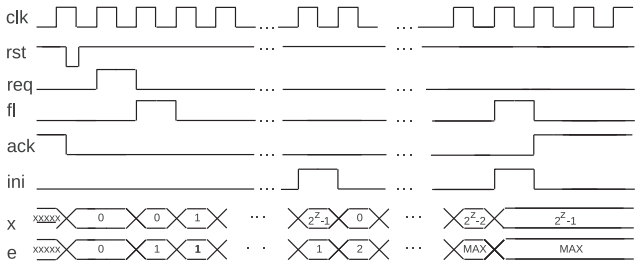

Fig. 3. Timing diagram of communication signals in the platform ($MAX = 2^N - 2^{N-k_{max}}$).

the saboteur i. Therefore, given a number of simultaneous errors k, the FAULT INJECTION MODULE generates the set of C_k^N vectors **e** containing exactly k bits equal 1.

This fault pattern generation is done based on an algorithm proposed in [5] comprising 4 tasks:

- TASK 1 – Set $e_i = 1$ for all $0 \leqslant i \leqslant k-1$, and $e_i = 0$ for all $k \leqslant i \leqslant N-1$. This task is performed only once for each value of k.
- TASK 2 – Perform a search from LSB towards MSB in order to find the position m of the first bit 0 after a bit 1 in the previous vector.
- TASK 3 – Create a temporary vector $\mathbf{t} = [t_{N-1} \cdots t_0]$ by flipping the bits e_m and e_{m-1} of vector **e**.
- TASK 4 – Permutate bits t_i and t_{m-2-i} of vector **t** for all $0 \leqslant i \leqslant m-2$.

In order to generate all the possible vectors **e** considering a range of simultaneous errors $k_{min} \leqslant k \leqslant k_{max}$, Algorithm 1 is performed.

Algorithm 1. Fault Pattern Generation

```
1: k ← k_min
2: for k ⩽ k_max do
3:    e_1^k ← Task1(N, k)
4:    j ← 2
5:    for j ⩽ C_N^k do
6:       m ← Task2(e_{j-1}^k)
7:       t ← Task3(e_{j-1}^k, m)
8:       e_j^k ← Task4(t, m)
9:       j ← j + 1
10:   end for
11: end for
```

In an effort to better understand Algorithm 1 realization, let us consider the generation of the whole set of vectors $\mathbf{e} = [e_2 e_1 e_0]$ for $k_{max} = 2$. In this case, the step by step execution is shown in Fig. 4.

3. Reliability assessment

The robustness analysis, such as the one performed by the FIFA platform, is well suitable for the reliability assessment of digital circuits. This can be done by using the Probabilistic Binomial Reliability Model (PBR) [6].

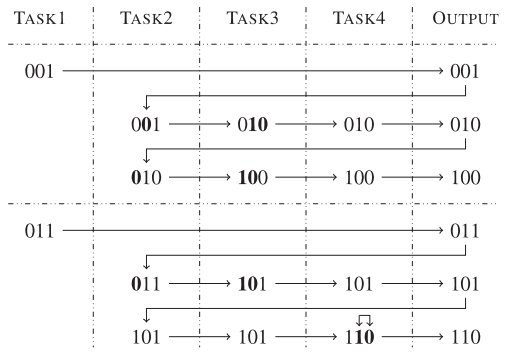

Fig. 4. Example of a step by step execution of Algorithm 1 considering $N = 3$ and $k_{max} = 2$.

According to PBR model, the reliability of a digital circuit is given by (1), where:

- N is the number of gates that may fail.
- q represents the reliability of a gate, that is, the probability that it does not fail. We consider all gates in a circuit as having the same reliability value.
- $f(k) = (1-q)^k q^{N-k}$ denotes the probability that k gates fail simultaneously. Notice that more complex models can also be used to evaluate this term as shown in [6].
- c_k denotes a coefficient related to the masking of k simultaneous errors in a circuit. Considering that the target circuit has Z input bits, it can be calculated using (2).

$$R = \sum_{k=0}^{N} f(k) c k \tag{1}$$

$$c_k = \sum_{j=0}^{2^Z-1} p(x_j) \left(\sum_{l=1}^{C_k^N} \overline{y(x_j, e_{N:0}) \oplus y(x_j, e_{N:k}(l))} \right) \tag{2}$$

4. Synthesis results

We have implemented a full parameterizable HDL description of the proposed platform. Indeed, parameters such as the number of fault-prone gates (N), the number of stimuli data bits (Z), and the number of output bits (P) can be selected in order to match a target design OP.

In order to evaluate the implementation cost of the proposed platform, we have synthesized several versions of the IP using a `STRATIX II EP2S180F1508C3` FPGA from Altera ®. Each implemented version has considered a different number of fault-prone gates (N). Even though the implementations have considered the same number of input/output bits ($Z = 5$, $P = 3$), the proposed IP can deal with any value of Z and P.

The synthesis results are shown in Fig. 5. Only the main components of the FIFA platform are taken into consideration. They are the STIMULI GENERATION, the FAULT INJECTION, and the MASKING ANALYSIS modules, which are responsible for fault injection and fault masking analysis.

When $N = 6$, the platform implementation requires less than 0.1% of the FPGA resources ($N_{LE} = 144$) and $f_{max} = 215.98$ MHz. Even if we consider a large number of fault-prone nodes, the proposed IP remains very compact. For example, with $N = 40$, the platform implementation requires only 2% ($N_{LE} = 3555$) of the available LEs in the target FPGA. In the case of $N = 40$, more than 80 millions test configurations can be generated in every second ($f_{max} = 82.41$ MHz).

In order to compare the FIFA platform with other similar platforms, we have also implemented the Fuse HDL tool proposed in [1]. The comparison in terms of resource requirements and performance is shown in Table 1. Both implementations were done considering $N = 10$ and $P = Z = 32$. Notice that the Fuse platform only deals with single faults, while the FIFA platform deals with single and multiple faults ($k_{max} = N$). Nevertheless, the proposed IP is more efficient in terms of time and resource requirements. If the occurrence of several simultaneous faults (large values of k_a) is not probable, we can still optimize the IP implementation by reducing the width of the buses $\mathbf{c_k}$, $\mathbf{k_{min}}$, $\mathbf{k_{max}}$, $\mathbf{k_a}$ and **k**.

Other tools, such as those presented in [7,8], have a temporal cost that grows with the complexity of the fault model. Instead of this, the FIFA platform presents a fault model which has no significant impact on its performance.

The tool presented in [2] only supports permanent stuck-at faults, and the platforms [1,3] only deal with single faults. Unlike those works, FIFA platform deals with single and multiple faults,

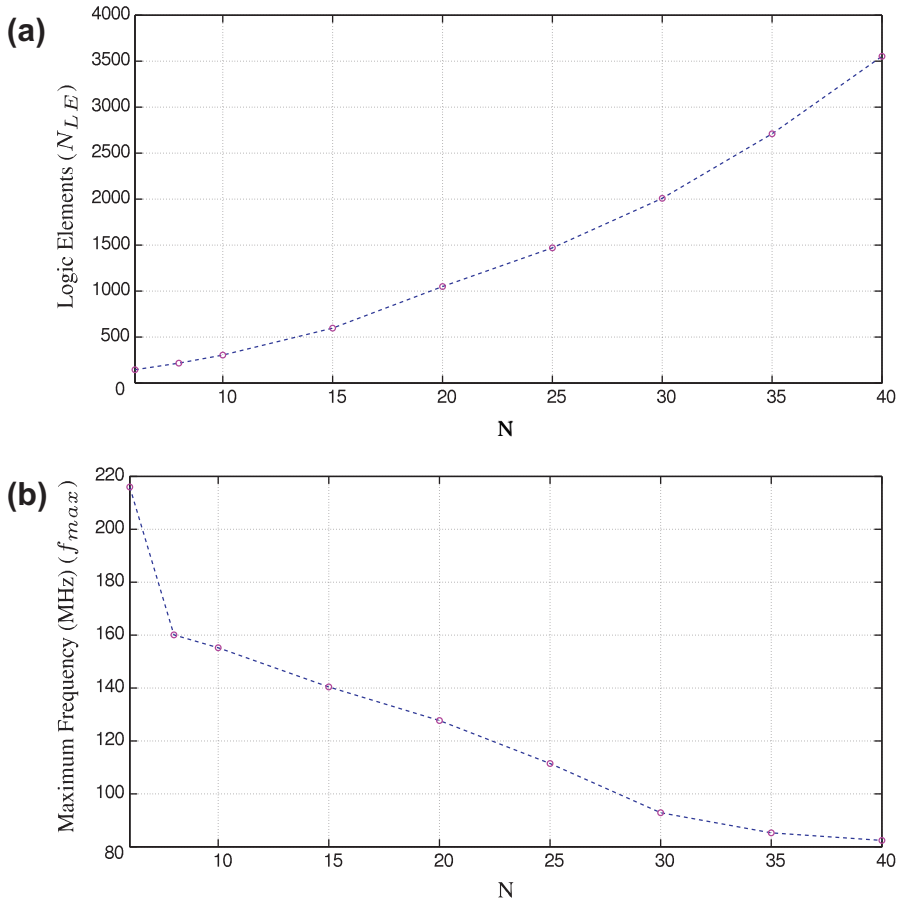

Fig. 5. Synthesis results of the platform (up to *N* simultaneous errors): (a) number of logic elements required in the FPGA. (b) maximum frequency for error generation.

Table 1
Fuse platform vs. FIFA platform.

	Fuse [1]	FIFA
ALUTs	2157	817
Registers	694	467
Maximum frequency	75.1 MHz	109.87 MHz

and it supports permanent and transient faults as described in Section 2.1.

As the fault injection approach can become very time-consuming for large circuits, dedicated hardwares may be used as a means to accelerate calculations [9]. Despite this solution have been used by Antoni et al. in [4], each test configuration requires reprogramming the FPGA. This significantly reduces the efficiency, even if partial reconfiguration is used. The proposed platform is composed in such a way that it avoids reprogramming the FPGA during the analysis of a given circuit. Indeed, using the available control signals we can select not only which nodes we would like to inject faults, but also the fault model (s) to be used.

The FIFA platform was implemented from scratch without benefiting from any proprietary libraries. Thus, all the required functions together with the memory control access (DMA – Direct Memory Access) was performed using standard cells, making the FIFA platform widely adaptable to be used with any FPGAs. The modifications should be restricted to the supervisor module, where we need to specify the communication interface between the platform and the computer according to availability in the target FPGA. Furthermore, the FIFA platform allows the designer to control the time complexity as well as the pertinence of the test configurations.

5. Conclusion

This paper presented a new platform for fault robustness analysis of digital circuits based on fault injection. The proposed platform, named FIFA, allows the designer to establish trade-offs between complexity and completeness of the analysis. The developed IP is fully parameterizable. Synthesis results have shown that it exceeds those reported in the literature in terms of area efficiency and performance. The FIFA platform deals with single/multiple simultaneous faults as well as permanent/transient faults. Moreover, if high fault multiplicity ($\mathbf{k_a} > \mathbf{k_{max}}$) is unlikely, all fault injections and tests related to $bf\,k_{bf\,a} > \mathbf{k_{max}}$ can be avoided without diminishing accuracy in the analysis process.

Acknowledgment

The authors thank Nanoradio STIC AmSud Program for the financial support given to this work.

References

[1] Jeitler M, Delvai M, Reichor S. FuSE – a hardware accelerated HDL fault injection tool. In: 5th southern conf. on programmable logic; 2009. p. 89–94.
[2] Cheng K-T, Huang S-Y, Dai W-J. Fault emulation: a new methodology for fault grading. IEEE Trans Comput Aided Des Int Circ Syst 1999:1487–95.
[3] Kammler D, Guan J, Ascheid G, Leupers R, Meyr H. A fast and flexible platform for fault injection and evaluation in verilog-based simulations. In: 3rd. IEEE int conf on secure software integration and reliability improvement; 2009. p. 309–14.

[4] Antoni L, Leveugle R, Feher M. Using run-time reconfiguration for fault injection in hardware prototypes. In: 17th IEEE int symp on defect and fault tolerance in VLSI Systems; 2002. p. 245–53.
[5] Marques EC, Paiva Jr NM, Naviner LAB, Naviner J-F. A new fault generator suitable for reliability analysis of digital circuits, Escuela Argentina de Microelectronica. Tecnol Appl 2010:41–5.
[6] de Vasconcelos M, Franco D, Naviner LAB, Naviner JF. Reliability analysis of combinational circuits based on a probabilistic binomial model. In: Joint 6th int IEEE northeast workshop on circuits and systems conference; 2008. p. 310–3.
[7] Boue J, Petillon P, Crouzet Y. Mefisto-l: aVHDL-based fault injection tool for the experimental assessment of fault tolerance. In: 28th annual international symposium on fault-tolerant computing; 1998. p. 168–73.
[8] Baraza J, Gracia J, Gil D, Gil P. Improvement of fault injection techniques based on VHDL code modification. In: 10th IEEE int high-level design validation and test workshop; 2005. p. 19–26.
[9] Mavroidis I, Papaefstathiou I. Accelerating hardware simulation: testbench code emulation. In: Int conf on ICECE technology; 2008. p. 129–36.