Towards Virtual Fault-based Attacks for Security Validation

R. Leveugle, M. Ben Jrad, P. Maistri
TIMA Laboratory
Grenoble University (Grenoble INP, UJF, CNRS)
46 Avenue Félix Viallet - 38031 Grenoble Cedex - FRANCE
{Regis.Leveugle, Mohamed.Ben-Jrad, Paolo.Maistri}@imag.fr

***Abstract*— Applications increasingly rely on secure embedded systems or "trusted hardware". ASICs (or smart cards) are typically used for high security but SRAM-based FPGAs are also appealing to implement lower-cost and flexible systems. In both cases, designers need a validation of the achieved level of security before they undergo long and costly official security qualification. This paper presents a methodology to accurately evaluate at design time the robustness level with respect to fault-based attacks, without resorting to costly equipments. Practical results are shown. The same methodology can be used in other contexts, for example to evaluate the robustness with respect to particle hits and radiations in spatial or aeronautics electronics although in this case, error models are in general easier to handle.**

***Keywords*— security, dependability, design time robustness evaluation, SRAM-based FPGAs, ASICs**

I. INTRODUCTION

Embedded hardware-software integrated systems are today the heart of many products. Most application domains are concerned since embedded systems bring new features, added value and competitiveness due to innovation. In many cases, such systems are critical either from a safety or from a security point of view. In both cases, the systems must undergo accurate validations before they are subject to official qualification procedures. In this paper, we will focus on the validations required to guarantee the expected level of robustness with respect to natural or intentional (malicious) perturbations, with a special focus on the later i.e., so-called fault-based attacks.

Due to the overwhelming costs induced by recent technologies, many applications cannot afford developing a specific integrated circuit. They therefore make use of programmable devices (often called FPGAs). For several reasons that will be detailed in the next section, FPGAs configured by uploading data in a volatile memory (SRAM) are very appealing platforms. However, their configuration can easily be perturbed due to the sensitivity of the SRAM to many perturbation sources. When critical functions such as for example crypto-processors are implemented in such a FPGA, it is therefore necessary to analyze and mitigate the effect of errors not only in the user logic and flip-flops but also in the configuration memory.

We will demonstrate in the next sections that under practical attack conditions (using a laser or power glitches) a large number of bits can be simultaneously modified. The most usual error models employed to analyze the effects of natural perturbations (e.g., single bit-flip, or SEU) are therefore not adequate in such situations. Although fault injection techniques can be used to evaluate robustness level and the efficiency of some protection mechanisms [1], the main problem remains the injection of realistic error patterns.

Of course, the best solution to demonstrate the robustness of a given design is to implement it and to put it under real perturbation conditions, for example in a particle accelerator for accelerated testing or under a laser for malicious attacks. However, using such equipments is not always possible due to their availability and to the cost such experiments induce. A new methodology is therefore required in order to reduce the global validation and qualification costs. Our proposal is to use fault injection campaigns as a first validation step, but with accurate error patterns in order to achieve a sufficient precision. These error patterns are first gathered during platform characterizations and can be re-used for several versions of the design, or several designs. The implemented analysis environment is flexible and can mimic different types of error sources. It only requires qualifying a single time the implementation target (e.g., the selected FPGA family) in the considered environments. Our evaluation environment has been implemented for Virtex II/Virtex II Pro devices but can be extended to other FPGAs.

We detail in section II the global context and motivations for this study. Section III summarizes practical results obtained under various attack conditions on several platforms. Section IV presents how such results can be re-used to perform accurate and low-cost security evaluations. Some results obtained with the new methodology are then shown in section V.

II. CONTEXT AND MOTIVATIONS

As previously mentioned, the behavior of an integrated system can be perturbed in several ways. Natural perturbations can occur for example due to ionizing radiations, particle hits or electromagnetic interferences. Malicious perturbations using for example a laser or voltage glitches can be used to discover secret data stored in a circuit. Such fault-based attacks have become one of the main threats for systems with high security requirements. An early example of fault-based attack is Lenstra's attack on RSA [2], taking advantage of the computations based on the Chinese Remainder Theorem. This attack combines fault injection and crypto-analysis to discover the secret key

 ISBN: 978-1-61208-149-6

stored in a circuit. Similar attacks have been published for other types of cryptographic primitives, in particular DES or AES [3, 4]. In the case of malicious perturbations, the errors generated in the circuit are often much more complex than those generated by natural perturbations. We will therefore focus on malicious attacks in this paper, although the presented methodology can also be used in other contexts. We will also focus in the sequel on transient effects i.e., spurious modifications of some data stored in the circuit. Several bits can be simultaneously modified but without damage to the chip; restoring the erroneous data can therefore restart a correct computation.

Secure systems are often implemented in ASICs i.e., circuits specifically manufactured for a given application. In the case of large production volumes, for example for Pay-per-view television access, specific circuits allow developers to achieve both low cost and maximum protection. However, the development costs induced by recent technologies are quickly increasing and ASICs are no longer affordable for many applications. In most cases, they are replaced by programmable devices (often generically called FPGAs). Among those devices, three main categories exist. Some devices can be programmed only once (e.g., by fuses or antifuses). Once programmed, they are quite close to an ASIC in terms of characteristics. Their main drawbacks are a limited offer and quite high prices. Another device category makes use of non-volatile memories to store the device configuration. The best devices in this category use today Flash memory that is quite robust with respect to perturbations although some critical parts (e.g., control logic) are more sensitive. Such devices are more flexible but slightly less robust than devices based on fuses. Finally, the third category of devices makes use of volatile memory (SRAM) to store the configuration. These devices are less costly than the others and they generally exploit the most advanced technologies. They offer the largest device complexity, so the best integration level. They can easily be reconfigured to add new features or correct some bugs in the system and they often have partial reconfigurability capabilities that can be used to optimize the system implementation and behavior. Specific protections are available to protect the configuration against cloning (e.g., by encrypting it). SRAM-based FPGAs are therefore very appealing for many applications. However, their main drawback is the sensitivity of the configuration SRAM to natural or malicious perturbations. Errors occurring in a SRAM-based FPGA can therefore affect not only the user logic and flip-flops but also the configuration memory. In the first case, the problem is similar to perturbations occurring in an ASIC and modifies either manipulated data or the application control flow. Errors in the configuration memory are in general much more difficult to cope with since several effects can be induced. Manipulated data can be modified, but the function of the circuit can also be changed and will remain erroneous (although the effect is not destructive) until at least part of the circuit is reconfigured. In some cases, there may also be no effect at all because this bit has no active role in the application definition. Identifying the actual effect of errors in the configuration memory is therefore quite difficult, especially when multiple bits are simultaneously modified. Several types of design techniques can be used to make a given application more robust, but they are more limited than in the case of an ASIC design, since they have to be compatible with the existing features in the FPGA; it is for example not possible to add some sensors to detect a given kind of attacks. Achieving and validating a given security level is therefore difficult. Before using costly experimental equipments such as lasers, the designer must justify that the implemented functions are well protected against realistic errors. In most cases, the efficiency of the protection mechanisms is evaluated at design time by fault injections (either based on simulation or emulation) [1]. One of the limitations is the accuracy of the error model typically used during the injections, that may adequately represent some types of perturbations but not necessarily all types. The approach proposed in Section IV aims at overcoming this limitation. The need for specific error pattern characterization is first illustrated in Section III.

III. ERROR PATTERNS WITH RESPECT TO ATTACK TYPES

A. *Results of Laser-based Attacks*

One of the most efficient means for fault-based attacks is today a laser. Such equipment allows the attacker to have a very good control on the error location, both in space and time. However, several types of lasers exist and they can be used in several manners; actual attack effects depend on these attack conditions.

We will summarize here some results obtained during practical attack experiments, in order to illustrate the variability of the possible effects.

We reported in [5] an attack campaign done on an ASIC manufactured in the ST HCMOS 130 nm process, with 6 metal layers. The ASIC implemented several versions of a Montgomery multiplier for RSA acceleration, two of them with parity-based protections (Protected1 and Protected2). The laser was a pulsed Yag laser with a green output at 532 nm, 6 ns impulsions and an energy tunable from 0 to 100%, with the possibility to control the spot size. During the experiments, we used a large spot size and we only used the energy "zero" that means the lowest possible energy level, corresponding to some "leakage beam". As shown in Figure 1, this very low energy was sufficient to create many errors in the circuit as demonstrated by the number of alarm signals that were activated, each of them corresponding to a different subset of the logic. This is especially interesting because the result was obtained without any special preparation of the circuit. We just opened the package and attacked the circuit front side through the metal layers.

Similar attacks on this circuit were attempted using a much more sophisticated laser [6] part of the ATLAS platform, the pulsed laser facility of the University of Bordeaux, dedicated to laser testing and analysis of integrated circuits. We used an ultra-short pulsed laser source (1 ps) with a wavelength tunable from 780 nm to 1000 nm and microscope objectives giving adjustable spot sizes ranging from 1 µm to 20 µm. The available laser pulse energy on the attacked circuit is adjustable up to typically

 ISBN: 978-1-61208-149-6

1 nJ. In that case, it was not possible to generate errors when attacking the circuit front-side.

This case study demonstrates that (1) complex error patterns can be generated, far from the usual single bit-flip or even multiple bit-flip (MBU) models and (2) the attack effects strongly depend on the perturbation source. As a matter of fact, the most sophisticated equipment is not always the most efficient to be used in attack contexts.

Experiments have then been carried out on SRAM-based FPGAs, especially Xilinx Virtex II devices. The reason was the availability of a tool called SEFEA-ProD that allowed us to analyze in details the configuration and data errors in these devices [7]. The v1000 FPGA used in the experiments was manufactured in a 0.15 µm CMOS process, with 8 metal layers. It is encapsulated Flip-Chip and was attacked backside through the substrate. Experiments were done on the ATLAS platform [8] but also with other lasers. One of them had a wavelength near 900 nm, a power of a few Watts and several focus levels corresponding to 8 µm, 20 µm, 40 µm and 100 µm expected spot diameters [9, 10]. For the experiments reported in [9, 10], the die was thinned by a mechanical process until a residual thickness of 30 µm to ensure a good optical transmission of the light in the active layers of the device. The various experiments allowed us to draw some conclusions.

First, in most cases, a medium or large number of configuration bits are modified even after a single shot but single-bit errors can also be obtained [10].

Second, one type of bit-flip (from 1 to 0) has a much higher probability than the other. This was explained by the structure of the memory elements [10].

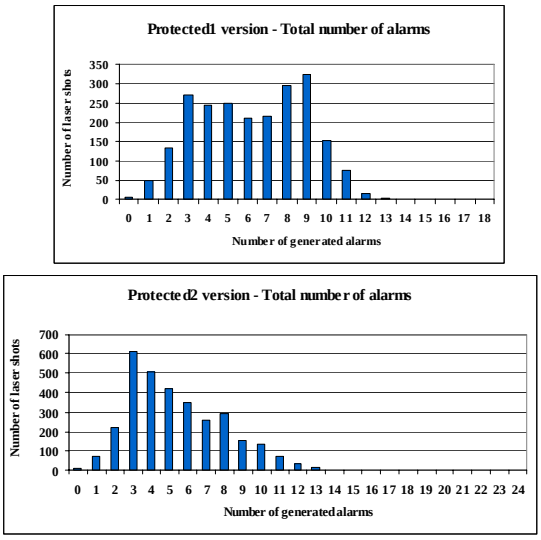

Figure 1. Repartition of shots with respect to the number of asserted detection bits in the protected Montgomery circuits.

Third, it was shown that in spite of multiple bit flips a non negligible percentage of errors does not affect the basic functionality. This is partly due to the unused resources in the FPGA for a given design, but also to a redundant encoding of interconnection control bits in the architecture.

Finally, the detailed analysis of the modified configuration bits demonstrated noticeable differences in terms of repartition depending on the laser energy and on the spot focalization. As an example, it was shown in [9] that the slice inputs and the Hex lines were the most sensitive elements in the CLB tiles (Configuration Logic Blocks), with 70% of errors impacting such connections. However, the most sensitive element between these two was dependent on the laser spot size. With a laser spot size of 40 µm, the probability to modify a slice input control bit was larger than the probability to flip a control bit of Hex lines. With the 8 µm spot, the trend was opposite.

B. Attack Parameters and Adversary Model

Results summarized in the previous section clearly show that it is not possible to use a simple error model to represent all possible attacks. An attacker can modify several important parameters and he will use the best values to achieve his goals. The robustness of a given design must therefore be validated taking into account a given adversary model based on some knowledge about the implementation technology, on the attack equipment that is supposed to be available to the attacker and on the configuration parameters for this equipment.

When laser-based attacks are concerned, the main parameters are related to:

- Manufacturing technology, internal architecture and internal topological organization of the circuit,
- Type of laser and configuration (wavelength, energy, pulse duration, spot sizes),
- Design implementation (placement in the device).

This implies first that a real validation of the design robustness can only be done quite late in the design flow; the final placement and routing must be achieved before.

Also, the validation cannot be done without taking into account realistic error patterns for the different attack conditions. Limiting the evaluation to a generic error model (such as single bit-flips or multiple bit-flips with a given maximum multiplicity) will not be in general realistic. On the other hand, single bit-flips can occur and therefore cannot be completely neglected. Also, when multiple errors occur in the configuration, all the potential combinations of erroneous bits are not realistic since they depend on the laser parameters but also on the physical organization of the configuration bits in the FPGA array. Making multiple injections in the configuration on the basis of a random sampling is therefore not accurate even if very large multiplicity values are taken into account. A more accurate representation of possible error patterns is therefore necessary to achieve significant validation campaigns based on fault injections.

C. Glitch-based Attacks

In the previous sections, we only considered laser-based attacks. Attacks can also be based on other types of perturbations, for example glitches on the power or clock signals. In general, the clock of a secured circuit is protected (e.g., by using an internal oscillator, or a digital locked loop in a FPGA) so we will focus here on glitches induced on the

 ISBN: 978-1-61208-149-6

power lines that are more difficult to mitigate when FPGAs are used. Depending on the glitch intensity and on the occurrence time, the effects can be very different. As an example, Figure 2 [11] shows results with several intensity and polarity of glitches. The attack was made on a v1000 device running an AES encryption. The glitches were triggered either on the rising clock edge or on the falling clock edge, with intensity between 5V and 80V, during the encryption cycles 49 to 60. As illustrated in the figure, the average number of bits modified in the FPGA configuration is very large (several hundreds in most cases) and clearly depends on the injection time within the clock period [11]. In that case, injections on the falling clock edge led to about twice erroneous bits.

The need for a good adversary model, with an accurate representation of possible error patterns, is therefore also confirmed in that case.

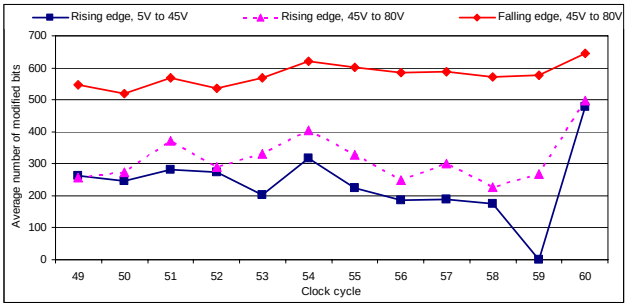

Figure 2. Average number of modified configuration bits in a v1000 device running an AES encryption for several types of power glitch attacks.

IV. METHODOLOGY OVERVIEW

A. *Fault injection techniques*

Several approaches have been used to evaluate at design time the dependability level of a circuit. We will focus here on approaches able to inject errors in both user flip-flops and configuration memories for SRAM-based FPGAs. In that case, simulation techniques are in general not adequate because there are almost no simulation models available including the configuration memory of commercial devices.

One approach proposed to perform fault injection campaigns is based on partial reconfiguration of a circuit prototype implemented on a SRAM-based FPGA [12-13]. The approach was primarily developed to inject transient errors in the user logic but had the potential to study the effect of errors in the configuration memory as well. The control of the reconfiguration was made by a host PC connected to the FPGA board. In [14], the authors proposed to control the partial reconfiguration by a program executed on an embedded processor. Such an "endo-reconfiguration" has the advantage to considerably reduce the number of data to be exchanged between the FPGA component and the host PC, thus accelerating the fault injection process. The approach is made possible by using the ICAP interface (Internal Configuration Access Port) provided by Xilinx for the Virtex FPGAs. Only SEUs (Single Event Upsets i.e., single bit-flips) were considered. Run-time reconfigurability is also used in [15] to inject faults in Look-Up Tables (LUTs) and in user flip-flops. An approach similar to [14] is presented in [16] but takes into account some multiple and cumulative bit errors. In addition, the environment can perform error propagation analyses, but restricted to specific fault detection or tolerance features. In [17], the FLIPPER platform is introduced to emulate SEU-like faults. Partial reconfiguration is again used but on a more complex platform composed of a main board and a daughter board with the FPGA under test. Complex dedicated hardware is therefore necessary. Similarly, dedicated hardware is required by FT-UNSHADES-C [18]. In addition, although this environment uses partial reconfiguration, it also requires implementing two copies of the system under test, one being used for a golden reference. The first consequence is of course to severely limit the complexity of the system that can be evaluated on a given device. The second consequence is that it is in general not possible to have the same placement and routing of the system during the evaluation and in the final product. Since the consequence of a given bit-flip strongly depends on the placement and routing of the design, such an approach is often too intrusive to give significant results. By comparison, using the ICAP interface only requires a small part of the device logic. With a guided placement and routing, it is often possible to avoid any change in the design implementation.

A few other approaches have been reported. On-the-fly modification of the configuration bitstream of XC4000 FPGAs, during reconfiguration, by some logic on the board was proposed in [19] but requires additional board-level modifications. The approach used in [20] focused on the logic used to reconfigure the FPGA component.

In the sequel, we will give results on a platform developed using the ICAP interface, but the methodology may be adapted on other types of fault injection platforms.

B. *Virtual Fault-based Attacks on SRAM-based FPGAs*

In general, fault injection campaigns are performed on the basis of a given error model, most often the SEU model implying one bit flip at a time in a circuit. In some cases, multiple bit errors are considered, assuming a given multiplicity value and a random distribution. However, as previously explained, such models do not represent well some perturbation conditions and they are unable to accurately take into account the actual layout and sensitivity of the elements in a chip or the physical characteristics of the perturbation source. Such models may therefore lead to large errors on the robustness quantitative evaluations.

Our goal is therefore to inject more realistic patterns, based on previous pre-characterization of the technological target. This pre-characterization can be done once for a given device and a given perturbation source (e.g., some particle flux, some electromagnetic fields or some type of laser with a given focus and a given energy). It can be done statically on the idle circuit, if possible using a device configuration that covers most of the possible configuration patterns for the CLBs and embedded memory blocks. It can also be done dynamically on the circuit running a given application. In the later case, results can be more precise with respect to this

 ISBN: 978-1-61208-149-6

application but can induce some bias if the analysis has finally to be done for a different application.

The actual error patterns obtained during this device pre-characterization are recorded and analyzed, e.g., with a tool like SEFEA-ProD [7], indicating all erroneous bits obtained in a bitstream after readback. Then the absolute values of the erroneous bits for each recorded error pattern are abstracted in order to obtain error coordinates that are relative to a CLB.

The abstracted relative coordinates are stored in a database that is then used during the fault injection campaigns performed at design time using partial reconfiguration, without resorting again to the physical perturbation source. Each relative error pattern in the database can be relocated i.e., injected into any CLB (or memory block) used in the device for a given design and at any clock cycle during the application execution for dynamic robustness evaluations. Of course, this relocation is only possible thanks to the regularity and repeatability of a FPGA structure. Another advantage of this regularity, in the case of laser-based pre-characterizations, is that only a small representative part of the device has to be scanned; the effects induced in the whole CLB matrix can then be inferred. This can noticeably reduce the laser availability requirements for the device pre-characterization. The relocation process is illustrated in Figure 3.

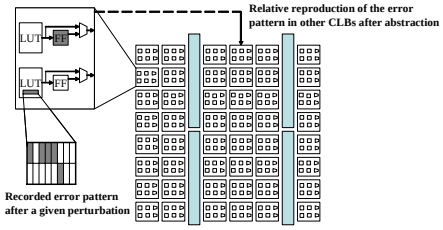

Figure 3. Illustration of error pattern relocation.

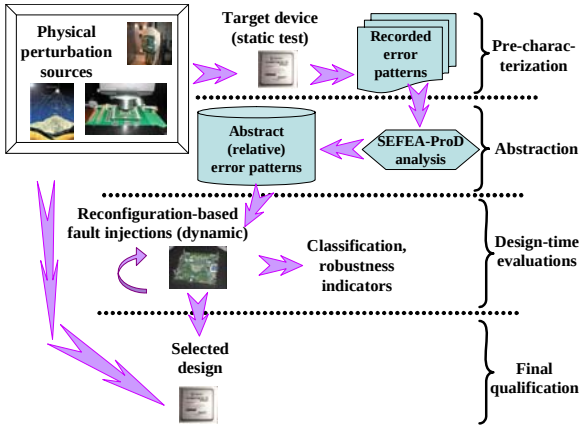

Figure 4. Global robustness evaluation flow.

Using the generated database, the same campaign can easily be run on several versions of a design in order to compare several implementations or several protection schemes. Furthermore, the consequence of any pattern obtained once in the device during the pre-characterization can be assessed at any position in the whole device and at any time. Let us notice that this would not be possible for example under a particle beam (for natural perturbations) due to the random distribution and the usually small number of events obtained during a given beam slot. A more comprehensive and realistic robustness analysis can thus be achieved at low cost with our methodology. The global flow is summarized in Figure 4.

C. *Virtual Fault-based Attacks on ASICs or some FPGAs*

For FPGAs with permanent or error-immune configurations, the same methodology can be applied but limited to errors in the user flip-flops.

In the case of ASICs (or any masked circuits) the relocation principle presented in the previous section is in general not applicable. However, the same concept can be used for regular circuits (parallel architectures, multi-cores …), limited to the repetitive structures.

Another possible use concerns the evaluation of software-related robustness. In the case of a microprocessor (or DSP), for example, the pre-characterization allows recording possible error patterns within the processor core. It is then possible to use the proposed approach to evaluate off-beam the effect on robustness of software modifications, provided a synthesizable model (or a very accurate simulation model) exists for the microprocessor. A processor prototype can be implemented on the FPGA and only error patterns in user flip-flops are injected.

V. IMPLEMENTED FRAMEWORK AND RESULTS

Our environment is implemented on a Virtex II-Pro component, using one of the embedded PowerPC processors to manage the fault injection campaign and the ICAP IP to perform the partial endo-reconfiguration. This environment is easily portable to other Virtex families. It allows the injection of any single- or multiple-bit error in the configuration and in the user flip-flops of the Configuration Logic Blocks (CLBs). The fault injection can be triggered at any time during the execution of an application onto the FPGA. The originality of the platform lies in the database we added to store the realistic error patterns, as presented in the previous section.

The methodology has been applied in the case of the Leon2 microprocessor, implemented on a Virtex II Pro device, and running several program examples chosen to be representative of several application areas: Fir is a FIR filter, Mtmx is a matrix multiplication, Sieve is a computation of prime numbers with the Sieve of Eratosthenes and AES is a standard encryption/decryption function. The results illustrate the impact of realistic error patterns compared to the most used model in the literature i.e., single bit-flips in the user flip-flops. The relative error patterns were derived from previous experiments [9]. The database included 5435 error patterns. Each pattern involved between 1 and 41 erroneous configuration bits, with an average of 11.7 bits per pattern. A single pattern can involve bits in several LUTs (from 1 to 6 LUTs in the patterns used for the experiments, 1.7 on an average). For each program, we have injected the relative error patterns in the CLBs used to implement the

 ISBN: 978-1-61208-149-6

Leon2 integer pipeline. We used statistical fault injection and the number of randomly selected injection time and location was chosen so that the margin of error on the classification results is 5% with 95% confidence. The number of injections performed for a given pipeline stage and a given program ranges from 8,000 to 160,000. The injected error patterns were classified, based on the system behavior after the injection, as "Silent" (no effect on the application results), "Error" (wrong result but the system is still alive and can perform other computations), "Failure" (wrong program termination, unexpected behavior, uncertain future behavior) or "Crash" (fully unrecoverable error and system behavior, until reset). The results are summarized in Table I for errors globally injected in the five pipeline stages. They clearly show that realistic error patterns lead to much more application failures. So an evaluation based on single bit-flips noticeably under-estimates the failure probability and is not acceptable for security analyses.

TABLE I. EXPERIMENTAL RESULTS OF FAULT INJECTION CAMPAIGNS

		Silent	Error	Failure	Crash
Fir	SEU FF	54.61%	11.63%	10.88%	22.89%
	Error patterns	3.40%	0.98%	60.54%	35.08%
Mtmx	SEU FF	52.59%	12.03%	14.55%	20.83%
	Error patterns	7.61%	0%	31.67%	60.72%
Sieve	SEU FF	47.77%	12.11%	23.46%	16.66%
	Error patterns	4.90%	0%	40.08%	55.02%
AES	SEU FF	57.96%	8.52%	24.67%	8.85%
	Error patterns	18.41%	0.52%	44.21%	36.85%

VI. CONCLUSION

Results shown in this paper demonstrate that classical fault injection campaigns based on single bit-flips noticeably overestimate the robustness of a design, at least for the type of perturbation considered here. The proposed methodology leads to more accurate robustness evaluations, with a minimum use of costly equipments since pre-characterization is done only once for a given device and a given perturbation source.

In this paper, we focused on malicious attacks because error patterns are in general more complex than those obtained under natural conditions. However, the same strategy can be used on the basis of a few events obtained for example under a particle beam, to take advantage of regular structures and increase the confidence in robustness. This approach will be used in further work to evaluate the efficiency of error mitigation techniques.

REFERENCES

[1] R. Leveugle, "Early analysis of fault-based attack effects in secure circuits", IEEE Transactions on Computers, vol. 56, no. 10, October 2007, pp. 1431-1434

[2] A. K. Lenstra, "Memo On RSA Signature Generation In The Presence Of Faults", private communication (available from the author), September 28, 1996.

[3] R. Anderson and M. Kuhn., "Low Cost Attacks on Tamper Resistant Devices", 5th International Workshop on Security Protocols (IWSP), 1997, LNCS 1361, Springer-Verlag, pp. 125-136

[4] J. Blömer and J.-P. Seifert, "Fault based cryptanalysis of the AES", e-Print Archive of the IACR, 2002, http://www.iacr.org/.

[5] R. Leveugle et al., "Experimental evaluation of protections against laser-induced faults and consequences on fault modelling", Design, Automation and Test in Europe Conference (DATE), April 16-20, 2007, pp. 1587-1592

[6] V. Pouget, D. Wan, P. Jaulent, A. Douin, D. Lewis, and P. Fouillat, "Recent developments for SEE testing at the ATLAS laser facility", Proc. of 15th Single-Event Effects Symposium, 2006

[7] V. Maingot, J. B. Ferron, R. Leveugle, V. Pouget, and A. Douin, "Configuration errors analysis in SRAM-based FPGAs: software tool and practical results", Microelectronics Reliability, Elsevier, vol. 47, no. 9-11, September-November 2007, pp. 1836-1840

[8] V. Pouget et al., "Tools and methodology development for pulsed laser fault injection in SRAM-based FPGAs", 8th Latin-American Test Workshop (LATW), March 12-14, 2007, pp. 167-172

[9] G. Canivet et al., "Detailed analyses of single laser shot effects in the configuration of a Virtex-II FPGA", 14th IEEE International On-Line Testing symposium, Rhodes, Greece, July 6-9, 2008, pp. 289-294

[10] G. Canivet et al., "Characterization of effective laser spots during attacks in the configuration of a Virtex-II FPGA", 27th IEEE VLSI Test Symposium (VTS'09), May 3-7, 2009, pp. 327-332

[11] G. Canivet, "Analyse et conception sécurisée de plates-formes reconfigurables", PhD thesis, Grenoble INP, Grenoble, France, September 2009 (in French).

[12] L. Antoni, R. Leveugle, and B. Fehér, "Using run-time reconfiguration for fault injection in hardware prototypes", IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, 2000, pp. 405-413

[13] L. Antoni, R. Leveugle, and B. Fehér, "Using run-time reconfiguration for fault injection applications", IEEE Transactions on Instrumentation and Measurement, vol. 52, no. 5, October 2003, pp. 1468-1473

[14] L. Sterpone and M. Violante, "A new partial reconfiguration-based fault-injection system to evaluate SEU effects in SRAM-based FPGAs", IEEE Transactions on Nuclear Science, vol. 54, issue 4, part 2, August 2007, pp. 965-970

[15] P. Kenterlis et al., "A low-cost SEU fault emulation platform for SRAM-based FPGAs", 12th IEEE International On-Line Testing symposium, Como, Italy, July 10-12, 2006, pp. 235-241

[16] C. Bolchini, F. Castro, and A. Miele, "A fault analysis and classifier framework for reliability-aware SRAM-based FPGA systems", 24th IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, Chicago, IL, USA, October 7-9, 2009, pp. 173-181

[17] M. Alderighi et al., "Evaluation of single upset mitigation schemes for SRAM bassed FPGAs using the FLIPPER fault injection platform", 22nd IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, Rome, Italy, September 26-28, 2007, pp. 105-113

[18] L. Sterpone, M. A. Aguirre, J. N. Tombs, and H. Guzmán-Miranda, "On the design of tunable fault tolerant circuits on SRAM-based FPGAs for safety critical applications", Conference on Design, automation and test in Europe (DATE), 2008, pp. 336-341

[19] M. Alderighi et al., "A fault injection tool for SRAM-based FPGAs", 9th IEEE International On-Line Testing symposium, Kos, Greece, July 7-9, 2003, pp. 129-133

[20] M. Alderighi et al., "A tool for injecting SEU-like faults into the configuration control mechanism of Xilinx Virtex FPGAs", 18th IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, November 3-5, 2003, pp. 71-78

 ISBN: 978-1-61208-149-6