RTL modeling of laser attacks for early evaluation of secure ICs and countermeasure design

Athanasios Papadimitriou

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Nanoélectronique et Nanotechnologies**

Arrêté ministériel : 7 août 2006

Présentée par

Athanasios PAPADIMITRIOU

Thèse dirigée par **Vincent BEROULLE** et
Co-encadrée par **David HELY** et **Paolo MAISTRI**

préparée au sein du **Laboratoire de Conception et d'Intégration des Systèmes (LCIS)**
dans **l'École Doctorale EEATS**

Modélisation au niveau RTL des attaques laser pour l'évaluation des circuits intégrés sécurisés et la conception de contremesures

Thèse soutenue publiquement le **27 Juin 2016**,
devant le jury composé de :

M. Jean-Luc DANGER
Professeur des Universités, Telecom ParisTech, Rapporteur
M. Lionel TORRES
Professeur des Universités, Université Montpellier II, Rapporteur
M. Bruno ROUZEYRE
Professeur des Universités, Université Montpellier II, Président
M. Vincent BEROULLE
Maître de Conférences, Grenoble INP, Directeur
M. David HELY
Maître de Conférences, Grenoble INP, Co-encadrant
M. Paolo MAISTRI
Chargé de recherche, CNRS, Co-encadrant
M. Régis LEVEUGLE
Professeur des Universités, Grenoble INP, Invité

English Title :

RTL Modeling of Laser attacks for Early Evaluation of Secure ICs and Countermeasure Design

This thesis is dedicated to my parents, Ilias and Roy.

For their endless love and support

Acknowledgments

Firstly, I would like to express my sincere gratitude to my PhD Director Prof. Vincent BEROULLE as well as to my supervisors Prof. David HELY, and Paolo MAISTRI for their continuous support and supervision during this thesis. I strongly believe that their supervision and experience besides leading me to complete this PhD successfully, it has also educated me in thinking as a researcher.

I would also like to thank Professor Regis Leveugle for participating to the supervision of this thesis. His experience has literally amazed me multiple times during my PhD and he is undoubtedly an example of a researcher in the domain.

Furthermore, I would like to thank all the members of the jury, namely the President of the jury Prof. Bruno Rouzeyre and the reviewers, Professors Lionel Torres and Jean-Luc Danger for all their very useful feedback and questions.

Additionally, I owe many thanks to Prof. Jean-Max Dutertre for his time and assistance in obtaining the experimental results by using the laser equipment at CMP in Gardanne. His contribution during the experimental laser campaigns was vital for the validation of this work. Furthermore it was a pleasure visiting CMP to work and exchange with such an experienced researcher.

I would also like to thank all the researchers and PhD students who took part in the LIESSE project. I believe that the harmonic collaboration and interaction during these 3 years has managed to enhance my skills in working as a part of a team.

I cannot thank enough my parents Ilias and Roy, so the best I can do is to dedicate this entire work to them with all my heart.

Endless thanks belong to my girlfriend and partner Igyso for her true support during this PhD. Her support started from day zero of this PhD and continues without stop even as I write these lines. Even though it will be very challenging I only hope that I will be able to provide her the same level of support.

Additionally I want to thank all the staff of the LCIS laboratory especially for creating an ideal collaborative and pleasant environment to work in.

Table of Contents

Table of Figures:

List of Tables

Glossary

AES	Advanced Encryption Standard
API	Application Program Interface
ASIC	Application Specific Integrated Circuit
CMP	Centre Microélectronique de Provence
CU	Control Unit
DU	Data Unit
DUT	Device Under Test
EDA	Electronic Design Automation
EM	Electromagnetic
FF	Flip Flop
FM	Fault Model
FPGA	Field Programmable Gate Array
HDL	Hardware Description Lenguage
IC	Integrated Circuit
IoT	Internet of Things
KU	Key Unit
LCIS	Laboratoire de Conception et d'Intégration des Systèmes
LIESSE	Effets laser et fautes sur les circuits intégrés dédiés à la sécurité
MET	Multiple Event Transient
MEU	Multiple Event Upset
OA	OpenAccess
PCB	Printed Circuit Board
PI	Primary Input
RTL	Register Transfer Level
SET	Single Event Transient
SEU	Single Event Upset
SFI	Statistical Fault Injection
SPICE	Simulation Program with Integrated Circuit Emphasis
TCAD	Technology Computer Aided Design
TIMA	Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés

1 General introduction and state of the art

1.1 General Introduction

1.1.1 Vulnerabilities of secure ICs under fault attacks

Integrated circuits are often considered as the root of trust of embedded systems requiring security. Examples include, but are not limited to: credit cards, set top boxes, access control, as well as recent fields such as the Internet of Things (IoT). The security mechanisms which are integrated in applications such as the above, involve cryptographic implementations. Nonetheless, cryptographic algorithms which are implemented on Integrated Circuits (ICs) should not be considered "a priori" as secure. Multiple techniques have been developed by hackers in order to reveal critical information of the implemented encryption algorithms. These include various types of side-channel attacks as well as fault injection attacks. The outcomes of such attacks can then be utilized by cryptanalysts in order to cancel the security of an application [1]. This could translate for example to deduce the secret key of an encryption algorithm or to enforce a denial of service.

Side-channel attacks are attacks which exploit information provided by the hardware that implements a cryptographic algorithm. For instance sources of information which can be utilized to perform side-channel attacks include the monitoring of the power consumption or the electromagnetic emissions of an IC, among others. Furthermore, side channel attacks can be used in order to assist the attacker in performing successful fault attacks.

Fault attacks exploit the faulty behavior of an IC in presence of computation errors. One of the first kinds of fault attacks applied to smart cards was the glitch attack. A sharp glitch is applied either at the power supply, the ground or to the clock of the card. This way the circuit may be forced to malfunction and faults can be generated during its functionality. While the glitch attack can be performed easily, it affects the entire circuit. Thus it is very difficult to inject faults to selected parts of the circuit [2].

Electromagnetic (EM) attacks consist in affecting the functionality of an IC by means of a high frequency EM pulse [3]. Such EM pulses generate eddy currents at the surface of the IC which in turn leads to faulty computations. Their advantages are that they can be implemented with low cost equipment and that they can affect even packaged ICs. Even though EM attacks can attempt to affect a specific part of an IC they cannot be focused so well as other types of attacks.

Light based fault attacks were introduced in [4]. Either intense sources of light, like a flash, or lasers can be used to disturb the functionality of an IC by inducing to it a current due to the photoelectric effect.

Between all the above different approaches to perform fault attacks, laser fault injection has proven to be a very effective form of attack [5], [6], [7]. Its effectiveness stems primarily from the excellent accuracy and control that it provides to the attacker during a fault injection campaign. More precisely, laser attacks can be used to inject faults characterized by high locality and timing accuracy, either on the input of a single combinatorial gate (Single Event Transient - SET) or in a single flip-flop (Single Event Upset - SEU). Parameters such as the

spot size, the pulse duration, the energy and the pulse repetitiveness can be controlled accurately to achieve a high level of precision of faults injected into a circuit. With proper settings, they can also be used to induce into the circuit multiple faults, besides single faults. Such faults can either be multiple event transients (MET) or multiple cell upsets (MCU) [8], [9].

Multiple different kinds of lasers can be used to perform a laser fault injection attack on an IC. Amongst these laser types the most expensive ones have powerful capabilities especially regarding the controllability over location and timing characteristics. Such laser are capable to greatly focus the laser beam and thus affect specific parts of an IC and furthermore to have very short pulse durations and rapid repetitiveness (order of picosecond). On the other hand, less expensive equipment have lower capabilities but their low cost may allow attacking ICs for which the investment of an attacker on fault injection equipment is small. Low cost lasers also have the disadvantages of smaller pulse duration and repetitiveness as well as larger spot sizes. Furthermore, attacks can be performed either at the front-side, through the interconnection layers, or backside, through the substrate of the IC.

The cost which is involved during the design and fabrication of a prototype IC enforces the need to evaluate secure ICs on various design level before fabrication. Such an evaluation must start early in the design flow in order to reduce design costs through the reduction of design re-spins due to the performance of an IC under fault injection attacks. Although fault injection campaigns, at design time, can be performed on various abstraction levels to evaluate a secure IC, the evaluation will be useful only if the injected faults are representative of an actual attack after fabrication. State of the art design-time evaluations are performed at the Register Transfer Level (RTL) descriptions and they usually assume a specific fault model involving either single or multiple erroneous bits after an attack. While single faults have an inherent locality during the injections, on the contrary for multiple faults the locality of the concurrently injected faults is usually not taken into account. The reason is that if placement and routing is not completed it is difficult to have available information about the multiple fault combinations which may occur concurrently during an attack. Layout information is important because a laser spot may affect more than one logic gate and/or flip flop and therefore lead to multiple injected faults, especially for recent deep sub-micron technologies.

1.1.2 Evaluation of secure implementations inside the design flow

While the most important and final evaluation of a circuit against laser attacks should take place after its fabrication (by performing experimental laser fault injection campaigns), it is not the optimal stage of the design flow to start considering the effects of such attacks. The main reason for this is that if the circuit is fabricated, to correct its weaknesses against laser attacks, which will be revealed during the experimental fault injection campaigns, will need the enhancement and re-fabrication of the IC.

The most important aspect of a laser injection is its ability to perturb only a very specific part of an IC. This can provide the attacker the capability to inject faults with very good location controllability which translates into affecting very specific functionalities of an encryption algorithm. This property is a very important and at the same time a challenging aspect of

a high level laser fault model. The main reason of the difficulty to model locality early in the design flow, i.e. at Register Transfer Level (RTL), is due to missing low level information (synthesis, placement and routing).

During the design process of an IC, designers start by describing hardware functionality by using a hardware description language (HDL), at RTL design stage. Afterwards the HDL design passes firstly through synthesis and then through the placement and routing steps of the back-end design, so as to obtain the layout of the circuit. Each consecutive stage is one step closer to the circuit that will be ultimately fabricated. Concurrently every stage towards the layout contains an increasing amount of information about the design that may prove very useful in the effort to model laser attacks. An inherent problem of low levels of abstraction is that their increasing complexity forces the evaluation duration to grow very rapidly, especially in relation with the size of a design and the information amount that needs to be simulated. This can prevent extensive evaluations. Furthermore, it has to be taken into account that these successive design stages are completed at the expense of design effort and cost. Indeed, design weaknesses discovered late in the flow require very expensive re-spins. An evaluation does not target the single purpose of characterizing the resilience of a circuit under laser fault attacks, but also to provide useful feedback to the design engineers so as to enhance the resilience of the circuit.

RTL is a very important stage, since it takes place early in the design flow but it already defines the functional characteristics of the final circuit. Furthermore, RTL designs can be easily simulated or emulated at design time in order to perform extensive evaluations. Therefore, evaluating the effect of fault attacks using only RTL information can be very useful in order to take the first decisions about appropriate countermeasures, able to efficiently detect or mask local hardware fault attacks. At the following design stages, results obtained at RTL can be confirmed by taking into account the more detailed descriptions of the circuit. The fact that each circuit will pass through all design stages until it is fabricated provides the opportunity to utilize all of them in order to verify the accuracy of the RTL evaluation. Using information from different design stages can be useful, except if we engage into too many feedback loops between design stages (move back from lower levels to higher levels of the design flow). Therefore an evaluation flow can maximize its effectiveness if it starts from the high levels of abstraction (RTL) and continues the evaluation along with the natural design flow, until it reaches the final layout. On the contrary, if we are interested in countermeasures defined at RTL, an evaluation at a low level of abstraction will certainly involve costly feedback loops.

An RTL evaluation requires an RTL laser fault model able of bridging the gap between the RTL and the final layout and therefore modeling the locality characteristics of laser fault injection. This gap is caused by the synthesis, placement and routing steps in the flow, through which a hardware description becomes a finalized layout. The aforementioned procedures are mainly targeted at the optimization aspects of the design flow and they do not have the chip's resilience against hardware fault attacks as a primary goal [10].

The goal of this thesis is to implement and validate a laser fault model at the Register Transfer Level of abstraction, which can be used during early evaluations of secure designs.

Such a fault model can contribute to the reduction or even elimination of the design re-spins among different abstraction levels so as to implement ICs resilient against laser attacks.

This work is a part of the ANR project “LIESSE” (Laser-Induced fault Effects in Security-dedicated circuitS). The main goals of this project are the study and modeling of the effect of laser attacks on deep submicron circuits and to provide efficient CAD tools to circuit designers to predict the effects of such laser attacks during the circuit design. Other partners work at lower levels than RTL, including the experimental testing and characterization of basic circuit blocks as well as the implementation of TCAD models concerning laser attacks on secure integrated circuits. Interaction between the partners and the results in all these abstraction levels are useful especially for the validation of the developed tools and fault models. For instance experimental laser fault injections have proved very important so as to validate the RTL approach presented in this thesis. Additionally evaluation of a design with the fault models which were developed in the framework of the “LIESSE” project can further strengthen the conclusions about the design’s resilience against laser attacks.

This way the designers will be able to validate their designs against laser injections without the need of expensive laser equipment or to evaluate the ICs after their fabrication. The designers will thus benefit from the possibility to evaluate early in the design flow the behavior of their designs against laser attacks. Another goal is propose countermeasures to strengthen further secure implementations against such attacks.

The next sections of this thesis are organized as follows. In the second part of this chapter, we present fault models and tools which are used in the literature to perform evaluations. Additionally, we point out why existing RTL tools are not sufficient to be used to perform fault injection evaluations considering the locality characteristics of laser attacks. In Chapter 2, an RTL laser fault model which considers the locality characteristics of laser attacks is presented. Chapter 3 applies the developed RTL fault model to state of the art designs of the Advanced Encryption Standard (AES) and discusses the results. Chapter 4 includes the validation of the RTL fault model through a layout-based validation flow and experimental laser fault injection results. In Chapter 5 we present and evaluate a novel RTL countermeasure for the protection of ICs against laser attacks. Chapter 6 includes the general conclusions of this thesis as well as some perspectives.

1.2 State of the art

In this chapter we begin by summarizing the basic physical effects by which a laser can induce faults inside an IC. Then we present the effects that a laser attack may have on combinational or sequential circuits and their classification. Next, we present the fault models which exist so far in the bibliography. We start by summarizing low level fault models and we focus to the gate level and mainly RTL existing fault models. The third part of the current chapter describes the various different platforms which can be used to perform fault injection campaigns which evaluate designs against laser attacks. Additionally we present the main methodologies and fault models which are so far used to model laser attacks. In the last part we present the project of which this thesis was part of and we explain its goals as well as the different objectives and partners of the project.

1.2.1 Laser fault injection

1.2.1.1 Summary of laser physical effects

Illumination of a semiconductor device by means of a laser source can cause high levels of ionization [11]. The photons which impact the semiconductor device cause ionization and generate free electron-hole pairs. This translates to an electric charge which can be collected by junctions of the circuit.

Reversely biased p/n junctions are capable to collect this charge, induced by the laser mainly through their depletion regions and the high electric fields which exist there [8]. Such reversely biased diodes form for example between the drain/source of an NMOS transistor and the substrate. This charge collection finally results in a transient photocurrent at the junctions which are illuminated by the laser. This photocurrent may affect either combinational or sequential logic cells and lead them to function erroneously.

In Figure 1.1 we can see a laser beam incident at the back-side of an NMOS transistor. The beam is focused at the drain of the NMOS and it induces electron-hole pairs in the region. This charge is collected by the space-charge region around the drain. This can be translated to a disturbance of the current at the drain of the transistor, as in the middle part of the figure. The transient current can be viewed as a perturbation at the drain node of the NMOS transistor and modeled as a current source as in the third part of the figure.

1.2.1.2 Laser Effects on combinational and sequential logic

When a laser attack affects combinational logic it may induce transient perturbations which in turn may lead to errors at the outputs of combinational gates.

- Single Event Transient faults (SET) are transient disturbances which appear at the output of an affected logic gate.
- Multiple Event Transient faults (MET) appear when multiple logic gates are affected concurrently.

Either single or multiple event transients may propagate and get captured at a sequential logic element (e.g. flip-flop) or they will not affect the operation of the circuit. To be captured

they need to appear at the input of a FF when e.g. a rising clock edge latches the input data to the FFs at their fan-out network.

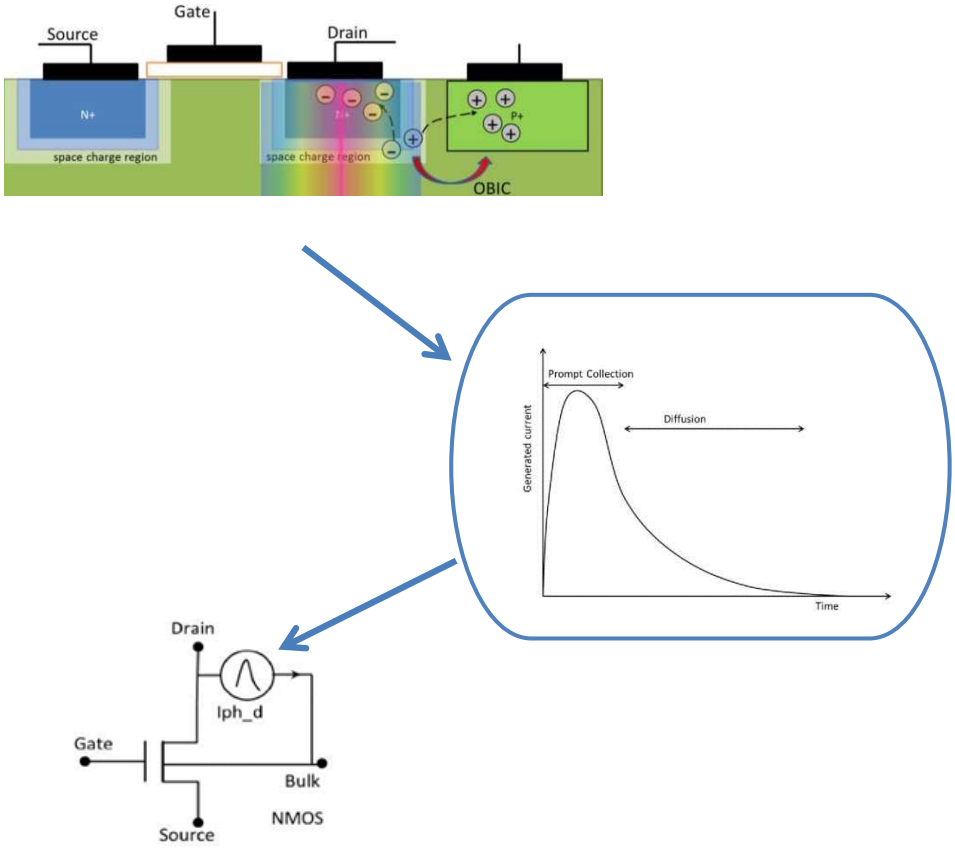


Figure 1.1: Laser induced photocurrent in microelectronic circuits – [Feng Lu – PhD, Simulation de fautes par laser dans les circuits cryptographiques]

Also SETs or METs do not lead to an error at the FFs of a circuit if they are electrically or logically masked. Transient faults are electrically masked if they do not have enough current and they fade out before reaching the cells in their fan-out. Logical masking occurs when even though the faults propagate to the inputs of other gates they cannot alter the outputs of these gates.

Sequential elements besides the fact that they may be affected and capture errors due to SET or MET faults, they can also be affected directly if the laser beam is directly focused on them.

- Single Event Upsets (SEU): occur when a laser shot affects a memory element (e.g. flip-flop) and this leads to an erroneous value.
- Multiple Event Upsets (MEU): occur when multiple memory elements are affected at the same time.

Even though due to logical and electrical masking it may seem that sequential elements are most prone to radiation induced faults, transient faults cannot be neglected. In [12] experiments have shown that error rates may be dominated by errors in the combinational logic

(e.g. NAND gates) especially as the technology nodes decrease and the operating frequencies rise.

Furthermore, in [13] it was observed that, for a 40nm technology node, at low frequencies the main contribution to the global error rates was due to errors in the sequential elements of the design. On the other hand at the 1.5–5 GHz frequency range, the error rates were dominated by combinational logic errors for a commercial technology. Experiments have shown that the use of hardened FFs lowers the range where combinational logic errors are dominant to the 1–3 GHz range. Thus, it is equally important to consider the effects of both transient faults and upsets.

1.2.2 Fault modeling

Fault models have been used for many years in order to predict the behavior of circuits under faults. There exist multiple models in all abstraction levels so as to provide the capability to the designers to evaluate the circuit's behavior under faults before fabrication.

1.2.2.1 TCAD modeling

At the TCAD abstraction level we encounter the most elaborate close-to-reality fault models. This is mainly due to the fact that TCAD is the abstraction level which is closer to the physics of the semiconductors and thus it can directly describe the phenomenon of the induced charge in the semiconductor. Then differential equations can be solved so as to determine the collection of this charge by the reversely biased p-n junctions of the circuit. It can perform either 2D or 3D analyses using finite elements and provide accurate input to spice-level fault simulators [14]. The drawback of such fault models at the TCAD level is that simulation times are very long even for low complexity circuits.

1.2.2.2 SPICE level modeling

Spice-level fault models can be used so as to provide accurate results and increase the speed of simulation in comparison with TCAD level simulation. Furthermore Spice level models are closer to the design procedure especially for analog circuits. In digital circuits spice-level simulations can characterize the behavior of logic gates under faults. For example in [15] the authors describe a transient fault spice model within the framework of a multi-level fault simulator. Even though spice level simulation is faster than TCAD simulation it is not practical for digital design validation.

1.2.2.3 Gate level modeling

Since the early days of integrated circuit manufacturing, several fault models have been used to describe faults, originating either from fabrication issues or from upsets caused by the interaction of high energy particles with an integrated circuit [8]. For particle fault analysis, classical approaches include the utilization of single or multiple bit-flipping and stuck-at models at the gate level [16]. Other utilized gate fault models include: bridging faults, transition faults as well as gate or path delay faults.

As described in [17], a bridging fault occurs when two signals are unintentionally shorted. Some of these faults can be represented by stuck-at behavior. The most common model used

for logical faults is the single stuck-at fault which confirms that a fault results in one of its input or outputs being fixed to either logic 0 (stuck-at-0) or logic 1 (stuck-at-1). This type of fault model offers a good representation for the most common types of failures. For this reason, it is one of the most utilized faults at the gate level mainly so as to model IC fabrication issues, if for example after fabrication one specific logic gate was stuck at zero or one. Furthermore, other kind of gate fault models are the delay faults and they appear during the manufacturing process that can influence the performance of a circuit without changing its functionality. There are two general types of delay faults, the gate delay fault and the path delay fault model. The most utilized gate delay fault model is the transition fault model, according to which every line in the circuit is associated with two transition faults, the rising fault and the falling fault.

The previously mentioned models, if defined with the capability of being transient, can be used to model the effects of a laser on an integrated circuit [18]. Gate level fault models can support simulation speeds which are a lot faster than lower level models even including timing information.

The major drawback of gate level fault models is that concerning multiple fault injections the fault space especially for multi-million-gate designs quickly explodes even for small multiplicities of faults. Furthermore, at gate level if there is no placement information it is difficult to predict which logic gate may be simultaneously affected by a laser spot incident on the IC.

1.2.2.4 Layout level modeling

When the finalized layout is available fault models which are defined in other levels of abstraction may be enriched with placement and routing information. For example in [19], the authors emphasize that the final layout is an effective source of information for the identification of adjacent cells and therefore the locality of an attack. Therefore the layout is a very valuable source of information regarding fault models attempting to describe faults which are induced by radiation. For instance it is possible to combine a gate level fault model with layout information so as to model the locality properties of a laser attack [15]. A major drawback of layout based fault models is that they are completed only during the final stage of the design flow. Therefore any critical problems which are discovered by such a model must certainly involve costly re-spins to enhance the robustness of a design.

1.2.2.5 Register transfer level (RTL) modeling

Many different models have been proposed in order to evaluate IC functionality under erroneous conditions. These include stuck-at faults as well as single transient faults of different types: bit-flip, bit-set, bit-reset, transients in combinatorial gates. Initially the size of technology nodes permitted erroneous IC functionality, due to either manufacturing problems or hard radiation, to be modeled by the single fault types mentioned above even at the RTL. In the current section we will describe the different fault models which have been used at the Register Transfer Level to model fault attacks. Afterwards we will highlight the drawbacks of these fault models to efficiently describe localized fault attacks.

- Single bit – flip

The single bit-flip is defined as the fault where the value of a memory element (e.g. flip flop) takes the opposite value of the on that it was supposed to have (in fault-free conditions) for a duration of one clock cycle. It was initially defined so as to describe the effects of an SEU fault of latching a wrong value in a memory element [20].

- Single bit – set

The single bit-set fault model is equivalent to a bit flip only if the flip flop has a value of zero. Therefore it is used to model the circumstances were a fault occurs only if the correct value of the flip flop is equal to zero. Otherwise (if its value is equal to 1) it does not affect the functionality.

- Single bit – reset

It is the opposite of a bit-set and therefore during a bit-reset fault, an error occurs only if the correct value of the memory element is equal to one.

Other RTL fault models combine the above basic fault models in addition to the gate level fault models with information from the HDL description language (e.g. VHDL) so as to inject fault on expressions and statements the HDL description [21].

Along with the IC technology advancement, for the recent deep submicron node sizes, the need to revisit these fault models arises. This is mainly due to the increasing need to utilize multi-bit faults to model erroneous behaviors since they become more and more relevant with each technology node release [9].

This transition to multi-bit faults requires more attributes to be included in a fault model; besides the type of faulty behavior of each cell (e.g., bit-flip) we also need to take into account the cells which may be disturbed simultaneously. Therefore the fault space is generated by the sets of cells that can be affected concurrently. Multi-bit fault models expand the corresponding fault space drastically. Therefore, it is necessary to generate a multi-bit fault space which will be realistic according to the considered fault attack or disturbance.

- Register multiple fault model

The register fault model enforces to multiple fault combinations which will be injected during a fault injection campaign to be concurrently inside the same register, as a register is defined at RTL. This multiple fault model makes the assumption that flip flops which belong to the same register will be placed in close proximity and therefore more than one of their bits may capture a fault concurrently.

- Random fault model

The random fault model considers that from all the flip flops of a circuit the candidates to capture concurrently a multiple fault are uniformly distributed. Therefore, for this fault model, it is equally probable for a multiple fault, of a certain fixed multiplicity (where by multiplicity we refer to the number of concurrently injected faults) to affect any possible combination of FFs of the design [22].

1.2.3 Fault injection evaluation platforms

In order to evaluate a design with respect to laser attacks through fault injections, there are the options of either simulation or emulation. Simulation based fault injection campaigns can take place at any level of abstraction or even on multiple levels. One state of the art multi-level simulation platform for laser attacks is tLIFTING, which will be summarized in the next section. On the other hand emulation based campaigns usually take place at RTL by using of FPGAs.

1.2.3.1 Multi-level fault simulation

In [23] the authors present the implementation of a layout-aware multi-level laser-induced transient fault simulator (tLIFTING). With the laser's geometry information and circuit layout as starting point, tLIFTING simulates the effect of faults with an electrical-level fault model.

tLIFTING is a fault simulator based on the open-source simulator Lifting [24], as in Figure 1.2. It is capable to perform 0-delay and delay-annotated simulations of digital circuits described in Verilog. For delay logic simulation, it models each circuit gate as a C++ class, while for timing simulation it implements an event driven simulator engine to perform delay-annotated simulations. The simulator can read the delay-annotated file, which provides information related to the delays of each gate, and the fault list, the netlist of the circuit and the input test sequence. Additionally, tLIFTING enables multi-level fault simulation in order to analyze the effect of transient faults due to a laser pulse at the electrical level. The electrical level drives the simulation of the circuit at gate level before the fault appearance and enhances the simulation run time compared to entirely transistor-level fault simulation.

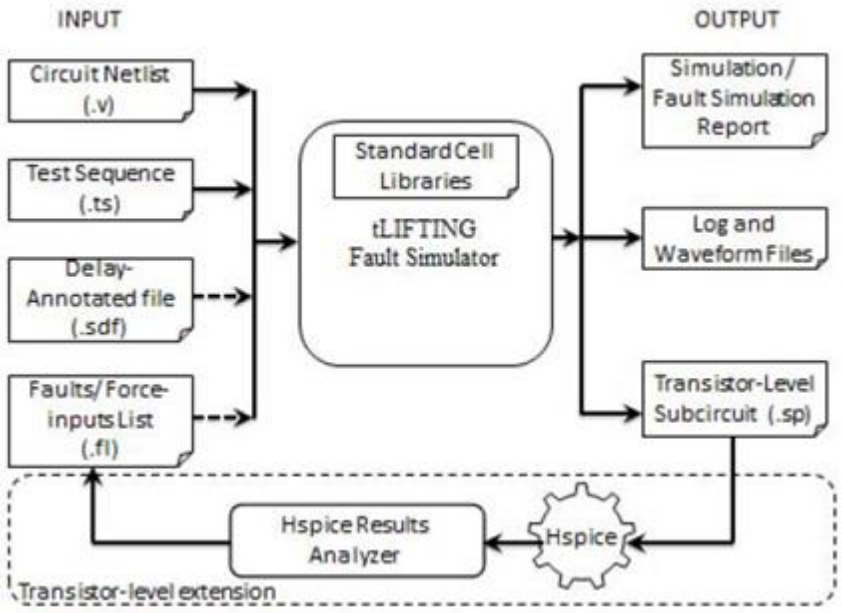


Figure 1.2: tLIFTING fault simulator [23]

Besides the acceleration which is achieved by such a multi-level approach the simulation speed is not comparable with RTL based simulation and especially emulation. Furthermore, this tool can be used at a late design stage since it takes as input the netlist of the design and in order to inject localized faults it also needs layout information.

In the next section we present state of the art fault emulation techniques and platforms which minimize the duration of RTL evaluations.

1.2.3.2 Fault emulation

In order to accelerate the performance of extensive fault injection campaigns multiple techniques have been proposed in the bibliography. Even though fault models defined at RTL allow a reduction of the duration of fault injection campaigns through simulation, the transition to the injection of multiple concurrent faults generates the need of even faster techniques. Fault emulation by making use of state of the art FPGA platforms allows a drastic acceleration of fault injection campaigns. Some of these techniques, used to instrument the device under test (DUT), are called HDL code modification techniques [25]. They involve the utilization of saboteurs or mutants in order to perform the injection of faults. Saboteurs are special HDL components which are added to the original design. When the circuit is operating under fault free conditions saboteurs are not affecting the circuit. On the other hand when the saboteurs are activated they inject specific faults at the locations they are placed. Mutants are components which replace components of the original design. When mutants are activated they alter (mutate) the functionality of the component so as to inject the desired faults, according to a specific fault model. Such techniques can be used either for fault simulation or if they are implemented in FPGAs they can perform fault emulation [26].

Fault emulation techniques as the above have the advantage that they can be very fast and inject faults in any desired node of the design which is well defined at RTL. Their main disadvantage is that they are increasing the complexity and the overhead of the instrumented design considerably. One way to solve this issue is by performing compile-time-reconfiguration (CTR) of an FPGA. This way only part of the design is instrumented and a bit-stream is generated to inject the required faults to this part. Then the design is re-instrumented and a different part of the circuit is injected with faults. The disadvantage of this technique is that CTR take large amounts of time in order to perform synthesis and place & route for each instrumented version of the design. In [27] the authors proposed an FPGA run-time-reconfiguration (RTR) methodology to solve the problems associated with CTR based fault emulation. During RTR based fault emulation the modification of HDL descriptions is replaced with modifications directly on the bit-stream of the FPGA. Therefore, there is the need to perform one reconfiguration for each fault combination that has to be injected but the procedure of synthesis, placement, routing and bit-stream generation is performed only once.

1.2.3.3 Random statistical fault injections

When the fault models which were described in the previous section are used to inject multiple faults the issue of having a huge fault space always arrives. This leads to very long durations of fault injection campaigns even with state of the art fault emulation platforms. One methodology to circumvent this issue is to perform statistical fault injection. In [22] the

authors present a random statistical fault injection methodology, based on statistics of proportions, and results of random fault injections on secure implementations. In the current section we will summarize this approach since it constitutes a fault injection approach and also since it will be utilized in the following chapters.

The main assumption of the random approach, which we will also refer to as random fault model, is that the characteristics of the populations (all possible errors at any clock cycle) follow a normal distribution. Also in order for the equations to hold every member of the initial population must have an equal probability to be sampled. This is equivalent with using a uniform distribution during random sampling from the population.

The population is constituted by all the members of interest of the design to be injected with faults at any possible clock cycle of the operation of the circuit. Even though the size of the population is huge even for medium sized designs, the approach does not approximate it with an infinite population. Sampling from the population is performed without replacement. Additionally the authors emphasize that when this statistical approach is used in the context of fault injection (instead of polls for example) the advantage is that the margin of error only accounts for the random sampling error and that there exists no other source of error.

The sample size determination for finite populations is based on a collection of random samples and on a confidence interval which is a equation of interval estimation of the characteristics of a population.

In this case, we can obtain the sample size n by the following equation:

$$n = \frac{n_o N}{n_o + (N-1)}$$

where n_o is the sample size without considering the finite population correction factor and it is given by the following equation:

$$n_o = \frac{z^2 p(1-p)}{e^2}$$

z is the cut of point which corresponds to the confidence level. This level is the probability that the exact value is in fact within the error interval. Usually, the confidence level chosen is 95%. The cut of point is computed with respect to the normal distribution according to the assumption of the method.

N is the initial population size. N mainly depends on the circuit (number of memory elements that can be modified by the source of fault injection). Furthermore it depends on the fault model which is also related to the multiplicity of the concurrent multiple faults as well as on the duration of the circuit's functionality in clock cycles.

p is the estimated proportion of individuals and the population having a given characteristic (it defines the standard error). It corresponds to an estimate of the true value being searched (for example the percentage of errors resulting in a failure). Since this value is not unknown before the fault injection campaign (between 0 and 1), the authors use a conservative approach which maximizes the sample size. Thus, the sample size will be chosen to make

sure that the expected margin of error with the expected confidence level no matter the computed value of the proportion. This is achieved for: $p = 0{,}5$.

From equations (1) and (2), we obtain:

$$n = \frac{N}{1 + e^2 \cdot \frac{N-1}{z^2 \cdot p \cdot (1-p)}}$$

and for the error:

$$e = z \cdot \sqrt{\frac{p \cdot (1-p)}{n} \frac{N-n}{N-1}}$$

The Margin of error e is the error on the computed proportion, $P_{computed}$ obtained during the fault injection campaign using the sample. The probability that individuals have the desired characteristic should be the interval $[P_{computed} - e, P_{computed} + e]$.

1.2.4 Conclusions

State of the art RTL fault injection tools and fault models such as [26] and [28] are general and do not consider any specific characteristic that can be used to describe the locality of laser attacks. The flip-flops (FFs) of the design are in general the targets of the fault injections, using the previously mentioned fault models. Usually, a maximum multiplicity is set in order to approach actual fault injection effects depending on the perturbation source. In order to choose the FFs which may be affected concurrently, usually statistical fault injection methodologies are used as in [22].

Complete randomness in selecting the FFs to inject faults is not suitable to describe the scope of a laser attack. This is mainly due to the fact that the scope of attack of an adversary in possession of state of the art laser equipment is to take full advantage of the accuracy of the laser. Therefore, random fault injection is not representative of good locality properties on the layout since they may lead to injecting multiple concurrent faults to FFs which are far away from each other on the layout. Additionally, it cannot guarantee functionally meaningful properties for an attack originating from a single laser fault injection. Thus even though random fault injection methodologies are necessary to cope with the huge fault spaces for multi-bit campaigns, it is also very important to select fault scenarios which are representative of an attack. Failure to achieve this may lead to unrealistic fault spaces which in turn can generate complex fault scenarios, not relevant with the effects of an actual attack. Such an evaluation can mislead the design team and lead to integrate to the design over-constrained countermeasures, with the associated overheads in terms of area and performance. Furthermore, on the contrary, a countermeasure which is efficient against random fault attacks can lead to a false assurance of security. This may occur since usually attackers do not try to inject random faults but rather very well controlled faults. Additionally depending on the utilized countermeasure random faults may be easier to be detected than well controlled faults.

In the case of faults caused by a laser and especially in the security context, the fault analysis should deal efficiently with the added complexity imposed by the laser characteristics and with the purpose of fault injection, which is the intention to extract hidden information. The complexity rises from the fact that a laser attack, especially in recent manufacturing technologies, provides to the attacker the flexibility of an excellent controllability over location and timing. Even a minimum spot size of the order of 1μm would affect several elements. Single bit flipping in registers does not describe the phenomenon accurately and multiple bit flipping fault models have to be used [6]. Additionally, the use of laser technologies that produce pulses with low jitter and high repetition rate influences the time domain aspects of an attack.

As far as we know, in the literature there exists no comprehensive RTL Laser Fault Model. In multiple different approaches generic fault injection platforms are used, as in [26], [28], with the capability to introduce multiple faults, either by simulation or emulation, but without any correlation with the capabilities of a laser. Fault modeling at RT Level has the benefits of occurring early in the design flow and of accelerating the analysis with respect to gate level models. Besides these advantages, it has the disadvantage that optimizations and technology mapping taking place in later steps of the synthesis flow, as well as placement, are unknown at this level of abstraction. Therefore, the registers and the important nodes of a design, for which we know in advance that they will not be affected by the synthesis flow, play a crucial role in the analysis. In this way, the transient multi bit stuck-at and bit-flip fault models can be used as the basic elements to implement a model of the effects of a laser in both the combinational and sequential parts of a circuit [29]. On the other hand, the complexity of such a fault injection campaign under exhaustive analyses can create an enormous fault space. The fault space derived by such an approach may lead to impractical computational durations, which make the simplification of the models a necessary step, in order for simulation or emulation to be completed within reasonable time. Furthermore, a large percentage of these faults will not correspond to possible fault attacks.

2 Modeling of localized attacks at RTL

Modeling of localized fault attacks early in the design flow is a valuable asset for the development of secure hardware. A fault model able to model localized injection of errors can be used to predict the effect of laser fault attacks. The Register Transfer Level (RTL) is a very important stage, since it takes place early in the design flow but already defines many characteristics of the final circuit. Therefore, evaluating the effect of fault attacks using only RTL information can be very useful in order to take the first decisions about appropriate countermeasures, able to efficiently detect or mask local hardware fault attacks. The main challenge of developing an RTL fault model concerning local fault attacks is the fact that a lot of information is missing during early design stages. These are primarily synthesis optimizations and mapping as well as placement. Furthermore, even though there also exist physical effects which cannot be modeled accurately by an RTL evaluation, it is nonetheless important to perform evaluations early in the design flow so as to avoid unwanted feedback loops.

At the following design stages, results obtained at RTL can be refined, by taking into account the more detailed descriptions of the circuit. An early stage fault model, able to model the locality of laser fault attacks, should provide the faults to be injected into the elements known at RTL so as to perform a corresponding fault injection campaign. Then, with the use of a fault injection platform, a fault injection campaign can be performed in order to characterize the robustness of an RTL implementation and potential countermeasures implemented at RTL according to the fault model.

As a consequence, the accuracy of the results of the fault injection campaign depends on the accuracy with which the fault model describes a fault attack. The first part of this chapter presents a fault model, describing localized fault attacks at RTL. Then, the second part of this chapter presents the use of the RTL laser fault model so as to validate the robustness of secure ICs with fault injection campaigns. Additionally, the same secure ICs are validated by means of the statistical random fault models so as to compare the results.

The work presented in this chapter has been published in the articles [P1], [P3], [P4] and [P5].

2.1 Laser fault attack properties

As already explained in the state of the art section, lasers prove to be very powerful means that perform precise fault attacks. Some key properties of a laser that also define the injection capabilities of an attacker are the following:

- Location
- Time
- Pulse duration
- Repetition rate

From the above properties, the injection location is one of the most important as it describes the excellent locality controllability that a laser possesses so as to attack a very specific part of the layout of an IC. For a laser source operating in the infrared range for example, the laser spot can be focused up to the diameter of 1 μm, limited by the diffraction limit, and target multiple standard cells especially for recent semiconductor technologies [5], [8]. Such laser spots can be used to inject faults with very good location controllability which translates to affecting very specific functionalities of an encryption algorithm implemented on an integrated circuit [30]. This property is therefore very important and the most challenging part of a high level laser fault model due to missing low level information (synthesis, placement and routing).

On the other hand, the time related properties listed above can be used in an RTL fault model in a straightforward way. The only limitation that RTL imposes on these properties is that they can be modelled only in multiples of the system clock period since gate level timing information (propagation delay) is not yet available at RTL. Therefore, according to the specifications of a particular laser source, such time related properties can be very accurately modelled and applied directly during a fault injection campaign. This is why in the following sections we will focus on modeling localized fault attacks. All timing related parameters will be addressed during the fault injection campaign.

2.2 Elaborated RTL netlist abstraction level

In the past, VHDL or other similar hardware description languages have been used to model fault attacks at RTL [31]. Modelling fault attacks at this level of abstraction embeds the difficulty that it is rather complex to accurately describe the locality of an attack. Furthermore, given such an RTL fault model, it is very complex to evaluate it with corresponding fault models at other levels of abstractions. Thus an important drawback of modeling laser fault attacks at the hardware description level is that such a model is very difficult to be validated in comparison with other lower levels of abstraction, which contain more accurate information. In order to circumvent this issue, a netlist which is representative of RTL and the corresponding hardware description language can be very useful. Such a netlist must represent the same functionality as the RTL description of the circuit without involving optimizations that take place during synthesis. The netlist which is generated immediately after the elaboration of the RTL code fulfills the above criteria and it can prove as a common ground between RTL and gate level.

The elaborated RTL netlist, from now on referred to simply as RTL netlist [32] is comprised by generic RTL cells not belonging to any specific manufacturable IC technology. These cells can be either functional RTL operators or generic basic gates and other RTL elements (e.g., multiplexers). One of the advantages of the RLT netlist is that it contains exactly the same functionality as the RTL hardware description of the design in a straightforward way since no gate level optimization steps have yet taken place. At the same time it has a netlist structure from which useful functional properties of the design can be easily extracted in an automated and fast way by means of an Electronic Design Automation (EDA) tool. The elaborated RTL netlist is a very suitable choice to model local attacks at RTL since it can provide the connection of RTL with lower levels of abstraction. This way it will be possible to perform an evaluation with an RTL fault model and validate it with regards to lower levels, as we will see in following chapters.

2.3 Locality properties of a laser fault model

In order to model the locality of laser fault attacks at RTL and at the same time to compare with other fault model and levels of abstractions, this section defines some important properties of an attack. These properties are: the origin, the recipients, the capture and the effect of an attack. After their definition they are intuitively explained below with some generic examples on abstraction levels following RTL.

- Attack ***origin:*** the area, space or elements of the considered level of abstraction which are potentially influenced (in comparison with normal operation) by the interaction of laser radiation with the IC at the physical level.

- Attack ***recipients:*** the elements of the considered level of abstraction, either in the origin of the attack or in the rest of the design, where the attack will be ultimately modeled.

- Attack ***capture:*** the propagation of the erroneous behavior of the attack origin to the attack recipients.

- Attack ***effect:*** the effect of the attack at its recipients, at the considered level of abstraction.

For each abstraction level the contents and the nature of the origin of the attack are different, but they express the location where the attack is instantiated. As we can see in Figure 2.1, in the physical and layout levels, the attack origin is naturally defined by the laser spot. In the case of TCAD level, it is defined as the volume in which the laser radiation is interacting with silicon and generates charge carriers (electrons and holes). At the transistor (spice) level netlist, the attack origin mainly contains transistors, while at the gate level it contains either combinational gates or sequential elements of the mapped technology. In the case of the RTL netlist, the origin contains either RTL combinational elements (e.g. generic gates, multiplexes, adders) or sequential elements (e.g. FFs).

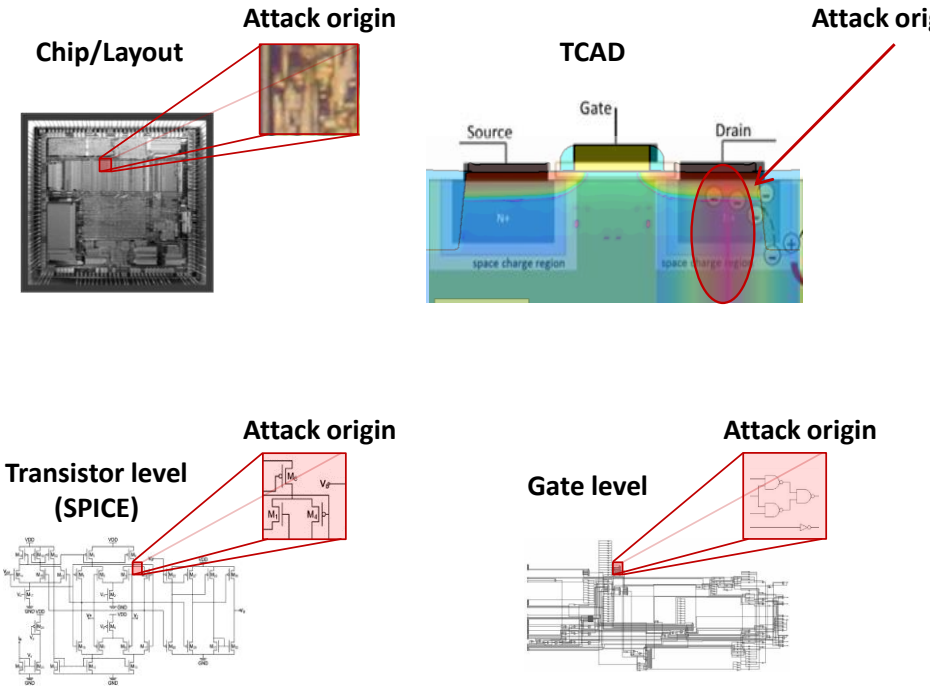


Figure 2.1: Attack Origin and different abstraction levels.

Depending on the level of abstraction used to model an attack, the attack recipients are the elements where the erroneous behavior will be captured after a laser attack at the attack origin. As an example at the gate level, they are the combinational nodes where faults have to be injected in order to model an attack in the attack origin. They may include elements of the circuit which exists inside the attack origin or in its vicinity. The attack recipients may even be defined in a different abstraction level than the level used to define the attack origin. Figure 2.2 depicts another example where a lower abstraction level, such as TCAD, is used to model a laser attack. The attack origin is defined at the TCAD level, as in Figure 2.2, and differential equations are used to determine the effect of a laser attack so as to determine the effect at the transistor level. In this case, the attack recipients have to be defined at the transistor level in order to simulate the behavior of the entire circuit [15].

Concerning the attack capture, the mechanism that propagates the induced faults from the attack origin to the recipients is in this case the generation of charge carriers, their absorption by sensitive (reversely polarized) nodes and the instantiation of a disturbance current (glitch) at the transistor level. The attack effect is the erroneous behavior of the attack recipients in comparison with normal operation.

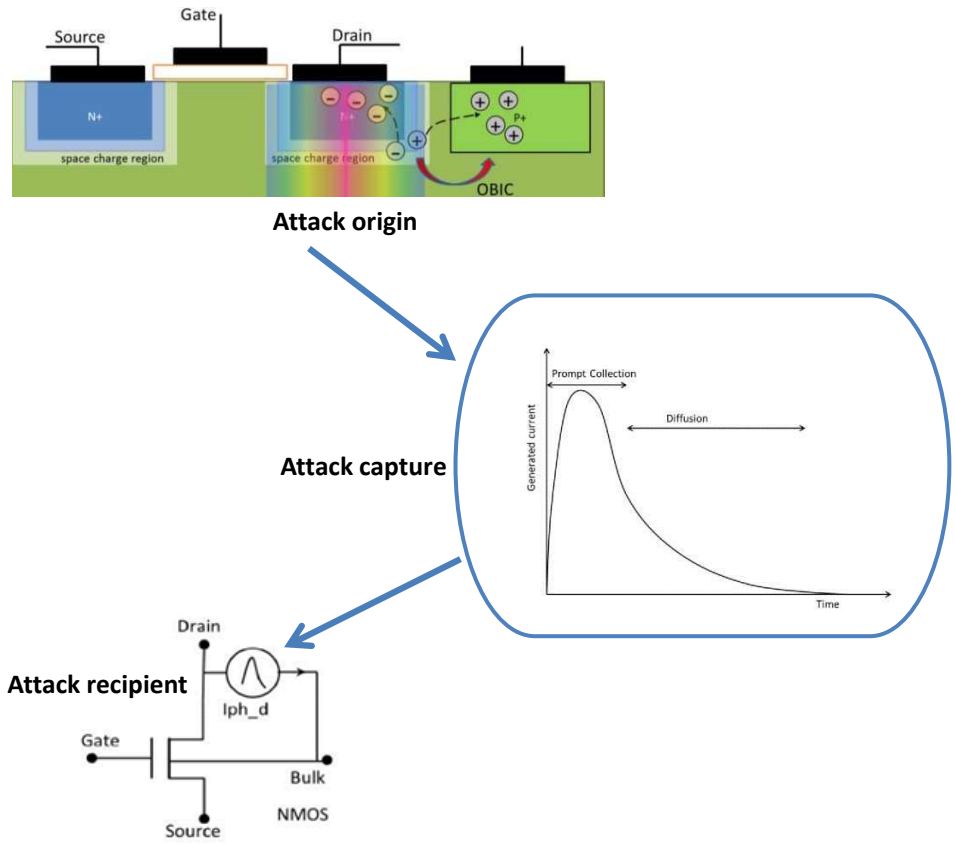


Figure 2.2: Attack origin, capture and recipient [Feng Lu – PhD, Simulation de fautes par la-

2.3.1 Locality properties definition at the RTL netlist

In this section the previously defined attack properties will be defined and explained at the RTL elaborated netlist abstraction level. These properties will be used later in order to define the fault model. Therefore it is important to define them in detail at RTL and explain them through some simple examples.

2.3.1.1 Attack origin

The attack origin is defined as the area, space or elements of the considered level of abstraction which may be influenced by the interaction of laser radiation with the IC at the physical level. For an RTL fault model and particularly for the RTL elaborated netlist, the attack origin must contain elements (or instances) of the RTL netlist. They can be either sequential or combinational elements belonging to the RTL netlist's library. Even though a local attack is well defined at the layout level, each RTL attack origin is supposed to contain elements which are more probable to be placed in the same neighborhood of the final layout. For each attack, and thus for each attack origin, these elements of the netlist will be considered as potentially erroneous due to a laser shot affecting their corresponding gate level elements, as in Figure 2.3.

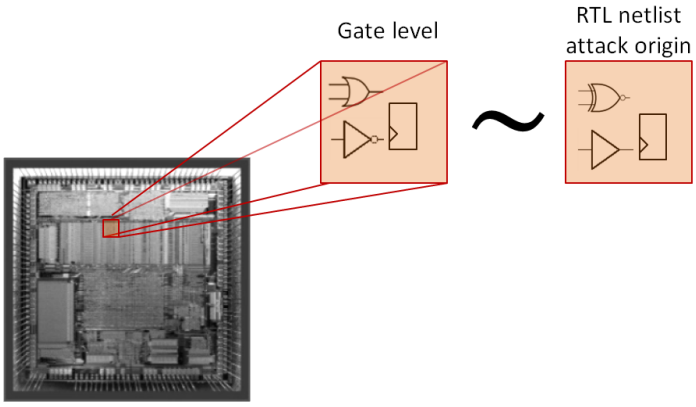


Figure 2.3: Attack origin at RTL and its corresponding origin at the gate level.

Even though there is currently no connection between the gate level and the RTL netlist corresponding attack origins, we will consider that the functionality of the contents of an attack origin at RTL may become erroneous due to an unknown corresponding attack origin at the gate level (or layout). Therefore as in Figure 2.3, when a laser shot affects a localized area of the layout of an IC, this translates into affecting specific gates of its gate level netlist. This set of gates corresponds to a set of generic gates of the RTL netlist where any erroneous behavior can be contained [33]. Comparing attack origins between RTL and gate level can be useful only in terms of validating the results of the RTL analysis with lower levels of abstraction when the design has naturally reached gate level and a completed layout. Therefore the attack origins of laser attacks need to be defined directly at RTL (in a top-down approach) by determining which of all the possible attack origins possess high locality characteristics, compatible with the capabilities of a laser source.

2.3.1.2 Attack recipients

Even though it is the elements which belong to the attack origin which are initially injected with faults, the circuit's erroneous behavior will not be modelled by simply injecting faults to these elements. Excellent candidates as attack recipients are the FFs of the design as they are the design's elements which are most likely to remain after gate level synthesis [18]. In general if an attack recipient is located inside its attack origin then we consider that this FF is directly attacked, which may result in a Single Event Upset (SEU) fault [8]. If on the other hand an attack recipient is not contained inside its attack origin, then we consider that this FF is indirectly attacked by a Single Event Transient (SET) fault [9].

Among the elements located inside an attack origin, only the FFs will be injected with faults according to our fault model. On the other hand, combinational elements located inside each attack origin will not be injected with faults. This is due to the fact that the RTL netlist is derived by elaborating the hardware description of the RTL code without including any major optimization steps or mapping. In Figure 2.4 we can see the various design stages of the design flow that starts with the hardware description and finishes after obtaining a finalized layout. The elaboration phase is taking place very early in the flow and results in the generation of the RTL netlist. As can be seen in the block diagram, heavy optimizations and mapping

take place only after RTL elaboration. This means that an RTL attack origin merely contains functional logic elements which will be totally changed during gate level synthesis, placement and routing. More importantly, the gate level netlist will contain completely different elements since the RTL netlist will be mapped to a specific technology library. For this reason the erroneous behavior of the design under laser attack cannot be modelled on the combinational elements of the RTL netlist as they will be completely replaced during gate level synthesis.

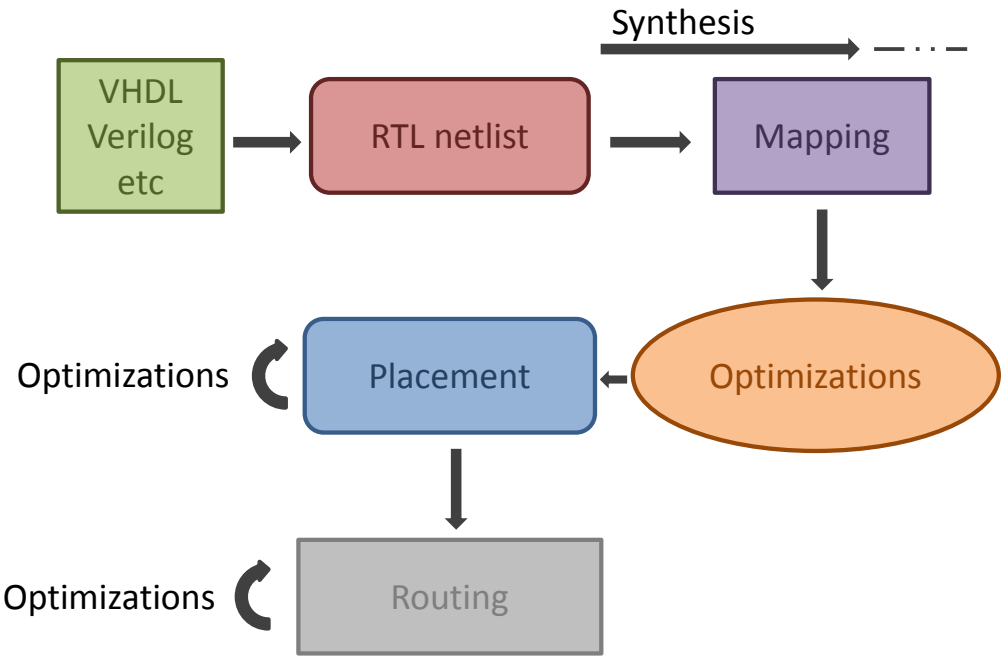


Figure 2.4: Design stages, from RTL to Layout

This fact makes it necessary to use elements of the RTL netlist which will remain after gate level synthesis in order to model the faulty behavior of an attack. Therefore, to avoid this obstacle, we define the attack recipients as the elements of the considered level of abstraction, either inside the attack origin or the rest of the design, where the attack will be ultimately modelled. This means that for each attack origin at the RTL netlist there exists one corresponding set of attack recipients. Figure 2.5 depicts an abstract RTL netlist where the attack origin contains several abstract generic gates and FFs (F1 and F2). As we can see F1 is potentially directly attacked by a Single Event Upset (SEU) fault since it is located inside the supposed RTL attack origin. On the other hand F2 is potentially indirectly attacked by a Single Event Transient (SET) fault, originating from the combinational logic located inside the attack origin and part of its fan-in network at the same time. The combination of these two potential faults may generate a multiple fault of a double multiplicity to the FFs F1 and F2. Therefore these two FFs can be used so as to model any erroneous behavior which may propagate to them, and which comes as a result of an attack inside this particular RTL attack origin.

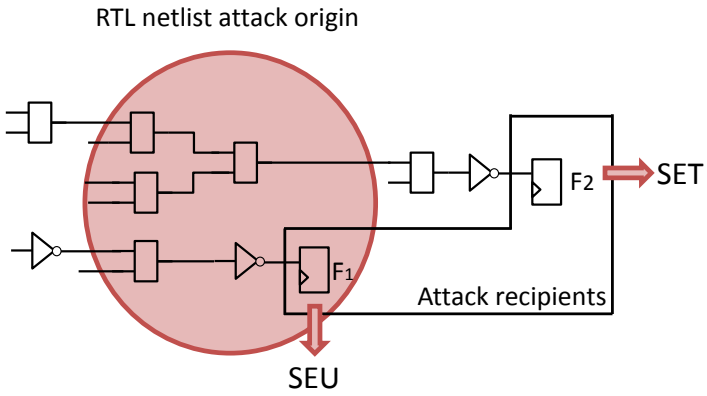


Figure 2.5: RTL Netlist and Attack Recipient Categorization

2.3.1.3 Attack capture

In order to derive which are the attack recipients which correspond to an attack origin we need to define the attack capture property. The capture property defines the propagation of the erroneous behavior of the attack origin to the corresponding attack recipients. This property therefore defines the way that faults originating from combinational components inside an attack origin can propagate to the attack recipients (the design's FFs). Therefore the origin and capture of an attack lead to the determination of the corresponding attack recipients.

The injection of multiple faults to combinational and sequential elements of a circuit by means of localized laser radiation can be achieved in two distinct ways as depicted in Figure 2.6 (a) and (b). As in Figure 2.6 (a), a laser shot may inject transient faults to combinational elements, which in turn may propagate to the FFs of the design (combinational attack) within one clock cycle. Otherwise, as in Figure 2.6 (b) it may modify the values of several FFs of the design because they are placed nearby on the layout, which we will call direct attacks (direct attacks on sequential elements) [P3]. Even though injection of faults can be due to combinational and direct attacks at the same time, this may be considered as a superposition of the two previously mentioned mechanisms. Let us now connect these fault injection mechanisms with the previously defined attack properties. For the combinational case the origin of the attack is a set of gates impacted by the laser. In the case of a direct attack the origin is a set of FFs, placed in the same neighborhood of the layout. In both cases the capture of the attack will lead to FFs as the recipients of any possible attack.

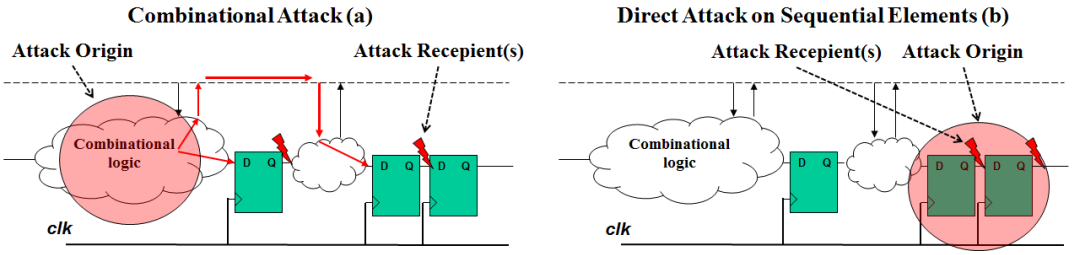


Figure 2.6: Combinational sequential attack capture mechanisms

Therefore concerning an attack capture involving combinational logic, faults induced inside the attack origin will be considered to propagate in the forward direction only (from the output(s) of a cell to the input(s) of another cell). One problem that this assumption entails is that it cannot model faults in the fan-in of elements of the attack origin. This issue will be dealt with in the next section when we will define better the basic assumptions of our fault model. The above attack capture mechanisms are also imposing some restrictions on the capabilities or the fault model. The fault model can predict attacks on combinational logic elements and FFs. Even though it is true that a laser attack may also affect the power and ground networks of an IC, modelling such faults at RTL is not meaningful since these networks are not yet defined at RTL. Therefore, our fault model does not consider such attacks as they are better addressed when the layout is completed.

2.3.1.4 Attack effect, fault type and fault multiplicity

Besides the origin and the recipients of an attack a fault model also needs to provide information about what kind of faults will have to be injected to the attack recipients so as to model an attack. This information is critical since the accuracy of the evaluation by means of a fault injection campaign depends highly on it. We define the attack effect as the result of the attack on its recipients. Therefore in the case of the RTL netlist the attack effect is the actual faults which get captured to the FFs of the design as a result of the erroneous functionality of the attack origin.

On the other hand the fault type is the fault which is used to instantiate the effect of the attack during a fault injection campaign. The fault types which will be used to emulate the effect of either a combinational or a sequential attack will be the classical bit-flip, bit-set and bit-reset faults but the methodology is not restricted in the utilization of any particular fault type [P4]. Furthermore it is a fact that an RTL fault model cannot predict the attack effect, since this is not accurately known at RTL as lower level information are required. However its definition is very useful in order to differentiate between the faults actually induced by a laser attack and the faults which will be used to emulate them.

Determination of the fault recipients and the required fault type is not enough so as to emulate a fault attack at RTL. Each specific attack will correspond to a set of attack recipients but due to the functional nature of the attack capture, any fault combination is possible inside each set. Therefore the number of the recipients of an attack can be very large and it should not be confused with the fault multiplicity during fault injections. As in the case of the fault

type, the fault multiplicity for each fault will be determined during the fault injection campaign.

2.4 Logic cone abstraction and functional dependencies

To define an attack, one option is to select specific cells of the elaborated RTL netlist as its origin. Even though the RTL elements describe the functionality of the circuit, later steps of optimizations and mapping will completely alter this netlist. Therefore, the origin of an attack may not be well defined across the design abstraction levels. In order to model the locality of laser attacks at RTL and to provide a good approximation of the origin of the attack, we perform logic cone partitioning of the design. Logic cone partitioning at gate level has been used in the past to perform automated fault diagnosis and locate the origin of one or more multiple faults, given some faulty outputs [34], [35].

For our analysis we consider cones which are bounded by a starting flip-flop (father) and expand backwards, from the outputs toward the inputs until we encounter either FFs or primary inputs of the circuit. This partitioning aims at modeling concurrently the direct and combinational laser attack mechanisms of the attack capture property. Since we intend to model both mechanisms with fault injection inside FFs, we must stress here that choosing relevant FFs is very important. After the partitioning, the elaborated RTL netlist (design) consists of FFs and the corresponding logic cones (which contain the combinational logic). Figure 2.7 depicts a logic cone and illustrates its boundaries, which are comprised by a single FF at its fan-out net and one or more FFs and/or primary inputs at its fan-in nets.

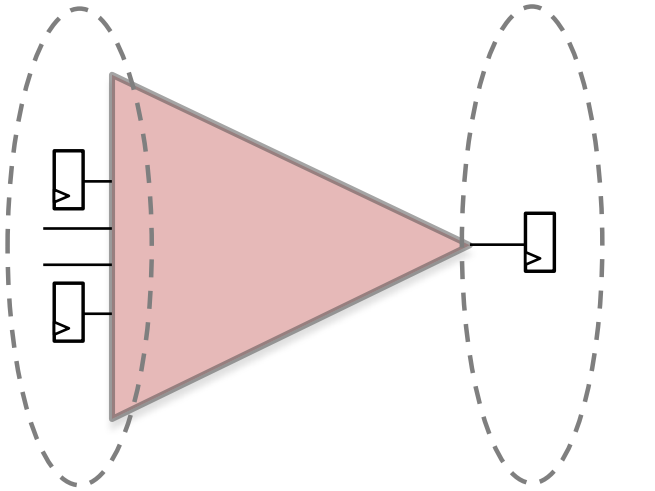


Figure 2.7: Logic cone fan-in and fan-out boundaries

Functionally dependent logic cones and FFs:

- Two logic cones will be defined as functionally dependent if they contain at least one common element or net (they intersect).

➢ Similarly two or more FFs are functionally dependent if their logic cones are also functionally dependent (they intersect).

As during synthesis the logic cones of the design will be altered (through mapping and optimizations), the content of each cone is not taken into account. For example two logic cones with different sizes and contents will be treated the same way by means of their functional dependencies. Figure 2.8 (a) shows two logic cones of different sizes, with their contents and their intersection. According to the previous definition, the two depicted FFs are functionally dependent. The same circuit is also depicted in Figure 2.8 (b) after abstracting the circuit in logic cones. This time the contents of the two cones are not illustrated since they are not taken into account and they are drawn having the same size. This way it is equivalent to view any design as a number of FFs and the functional dependencies between them. Thus, if we want to model an attack only on the combinational logic, the origin will be located inside one or more logic cones.

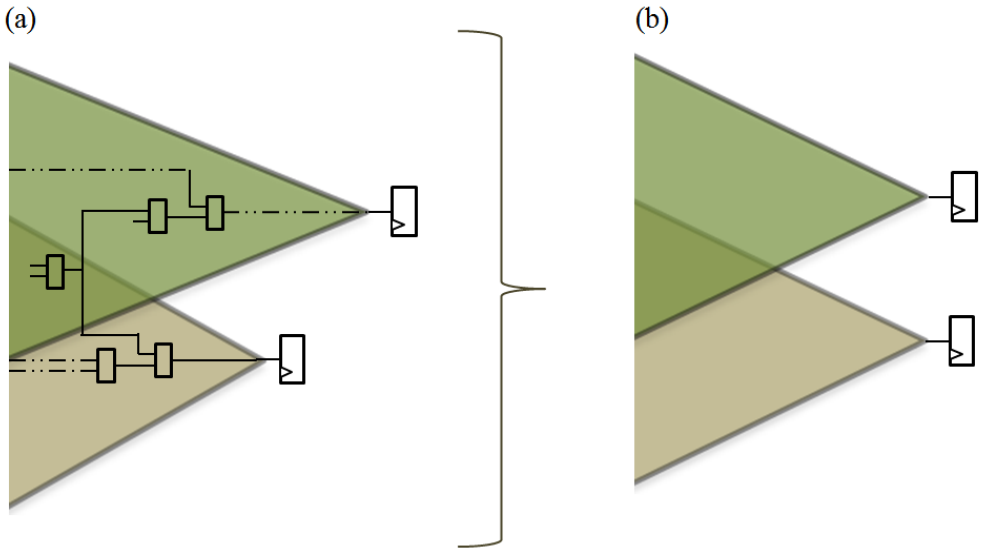


Figure 2.8: Logic cone abstraction. (a) Cones include elements and are drawn according to their size, (b) Cone abstraction according to functional dependencies (size is the same since the contents are not important)

On the other hand, the functional dependencies between the FFs can be used to model at the same time direct attacks. This is possible mainly due to the fact that FFs involving functional dependencies (of their fan-in cones) are considered more likely to be placed in a given neighborhood. Furthermore logic cones contain all the local critical paths of the design and any optimization taking place in the design flow will tend to shorten these paths so as to achieve timing closure.

2.5 RTL fault model description

In this section we will define the properties and the assumptions of an attack according to the proposed RTL laser fault model [P3].

2.5.1 Single affected cone assumption

In this section we introduce the assumption of the fault model regarding which are the possible attack origins that we consider during a laser attack. This assumption is very important since it defines the precision that the adversary has over the selection of faults he can inject to the circuit.

- Assumption 1 – Attack origin:

We define the origin of a laser attack to be contained inside an entire logic cone. Therefore each attack origin affects the chosen cone and at the same time, all the cones of the design that this cone intersects with. In the case of Figure 2.9 we can see in red one of the possible attack origins. For this attack origin we can see that also cone 2 is considered concurrently affected since the logic element i3 belongs at the same time to cones 1 and 2.

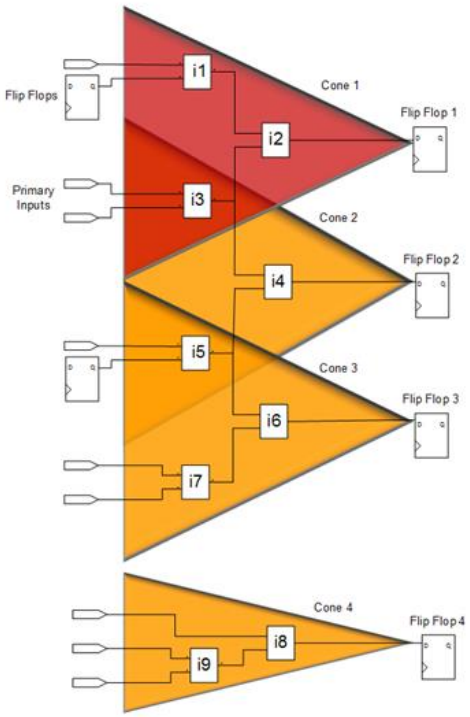


Figure 2.9: Single cone attack origin assumption

This assumption generates a discrete number of possible attack origins which is equal in number to the logic cones of the design (and thus equal to the number of FFs of the design). In terms of FF dependencies this is equivalent to assuming that an attacker may target concurrently one FF of the design and at the same time all the FFs which are functionally dependent with it. Additionally this characterizes the capabilities that the fault model will assign to an adversary so as to inject faults by means of a laser fault injection platform. According to this assumption the fault injection capabilities of the adversary is very precise as he is able to focus his attack to any single element of the design (combinational or sequential). This is a consequence of the fact that for any element of the design the attacker is able to target a logic cone that contains it. The restriction that this assumption imposes, involves which are the elements of the RTL netlist which can be simultaneously affected by an attack. Thus it enforces

to the adversary the restriction of not being able to affect at the same time elements belonging to two (or more) logic cones which do not intersect.

In section 2.3.1.3 where we defined the attack capture property, it was noted that the attack capture cannot take into account a fault induced in the fan-in of an element. In the case that this fan-in is connected to other elements outside the considered attack origin, it is true that a fault may affect other elements not located in the fan-out of the origin. The assumption made in the current section solves this issue since the logic cones are including the input nets. Therefore because of the assumption of the current section, that the attack origin is an entire logic cone, the aforementioned issue is eliminated. By including the input nets in each cone we obtain all the scenarios of possible attack recipients even if a fault is induced at the attack origin's inputs.

2.5.2 Combinational attack capture

- Assumption 2 – Combinational attack capture

A combinational attack is captured, within one clock cycle, through the logic cones (functional dependencies) to the recipients of the attack, which are the FFs located at the fan-out of each logic cone (father FF), involved with an attack, as in Figure 2.11.

Therefore for a given attack origin (cone), we can obtain the attack recipients, if we intersect the origin (cone) with all the remaining logic cones of the circuit. Then, the father FFs of all the cones intersecting with the origin generate a set of FFs which are the potential recipients of the faults injected in the origin of the attack. For example in Figure 2.10, when the origin of an attack is cone 2, is intersecting with cones 1 and 3, then the potential recipients of this attack are the FFs 1, 2 and 3. Finally, the effect of an attack is defined by the chosen fault type and a corresponding multiplicity. The fault multiplicity is not yet defined by the fault model at this stage. The fault model generates sets of FFs as potential recipients of localized attacks and thus imposes constraints on which are the FFs that can be simultaneously affected by a local laser attack. For the simple design example of Figure 2.10 this means that, according to the fault model, an attack that affects at the same time the FF1 and FF4 is not possible, for any chosen attack origin.

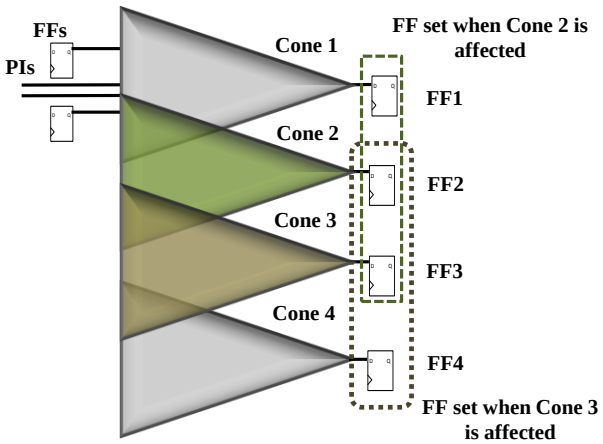


Figure 2.10: Another example of the determination of the attack recipient sets

The example of Figure 2.11 (a) illustrates the situation where cone 1 is considered as the attack origin (red cone). As we can see any fault induced inside this origin may propagate to the fan-out FFs of cones 1 and 2. Therefore, in order to determine the attack recipients for this attack, we need to extract all the logic cones that the attack origin intersects with. Then we may consider that cone 2 is the origin and determine the recipients for this case. The results for all the possible attack origins of this simple example are included in Table 2.1. In each row we can see the affected instances for each different attack origin, the cones that intersect with each origin and the corresponding attack recipients. Figure 2.11 (b) illustrated the situation where a localized attack is affecting only cone 4. In this case no other FF of the design can be affected and the attack will lead to a single fault being captured in FF4.

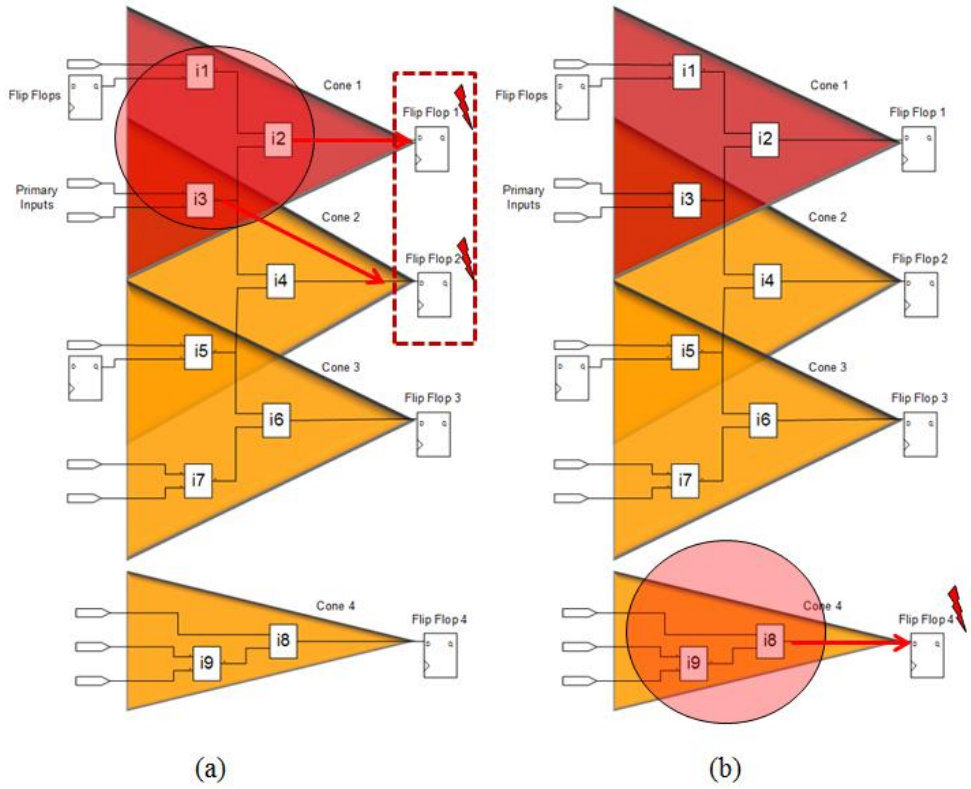


Figure 2.11: Attack recipient extraction given the assumptions for the origin and the capture of the attack, (a) Recipients when cone 1 is affected, (b) Recipients when cone 4 is affected. Both scenarios cover the cases where an attack (laser spot) is affecting only cone 1 or only cone 2.

Table 2.1: Attack recipients of the example of Figure 2.11

Logic Cone	Instances	Intersecting Cones	Attack Recipient FFs
1	i1, i2, i3	1, 2	F1, F2
2	i4, i3, i5	1, 2, 3	F1, F2, F3
3	i6, i5, i7	2, 3	F2, F3
4	i8, i9	4	F4

A clear disadvantage that arises from the assumption for the possible attack origins is that it is possible to have attack scenarios where a laser spot on the physical level is able to affect at the same time two non-intersecting cones as in Figure 2.12. Also from Table 2.1, in the attack recipients' column there exist no set involving F3 and F4 at the same time. Thus a concurrent fault in FFs 3 and 4, due to the attack depicted in Figure 2.12, will never be evaluated during a following fault injection campaign. Results about how often this situation occurs will be presented later in the validation chapter by considering various circuits and by comparing the RTL analysis with an analysis of the layout.

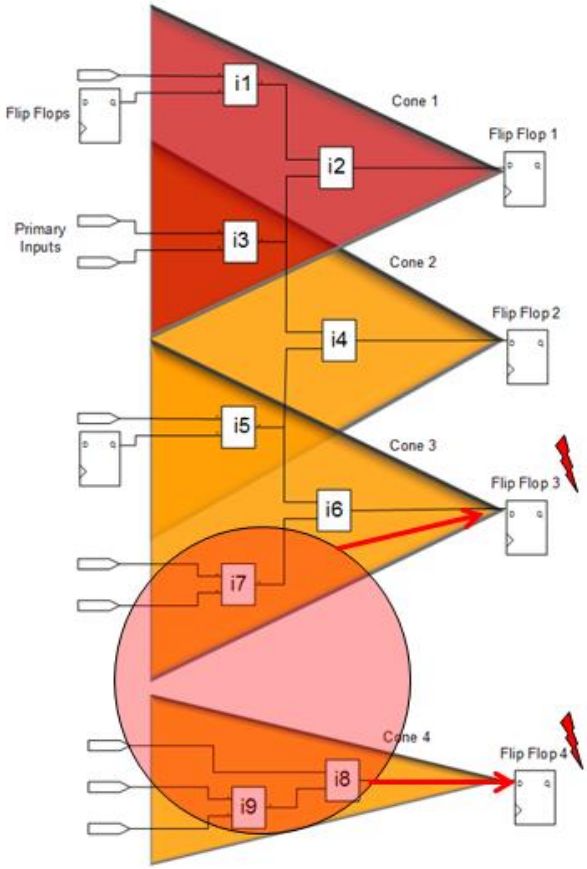


Figure 2.12: Attack scenario which is not compatible with assumption 2

Even though the single cone attack origin will be the main assumption of the fault model, which is going to be used and validated in the next chapters, it is possible to also use different assumptions for the RTL attack origin. Once the functional dependencies between FFs of the design are extracted, they can be used to easily obtain the attack recipients for any attack origin different than the single affected cone origin. In Figure 2.10 we can see that in case we want to change the assumption of the single affected cone attack origin and use multiple cones as affected by a laser attack, all we need to do this is to combine these results. For example if we want to assume that logic cones 1 and 2 are concurrently affected then we only have to consider that the possible recipients of this attack is the union of the recipient sets for

two independent attacks on cones 1 and 2. In the case of this example this leads to obtaining any combination of FFs as potential attack recipients.

2.5.3 Direct FF attacks

➢ Assumption 3 – Direct attacks on FFs

Direct attacks causing multiple faults on FFs are possible on FF combinations which are functionally dependent (their logic cones intersect).

Therefore, the fault model assumes that any direct multiple fault attack on FFs may occur strictly on FFs which involve functional dependencies, as in Figure 2.13. This stems out of the assumption that FFs that are functionally dependent are more likely to be placed inside the same local neighborhood of the final circuit layout.

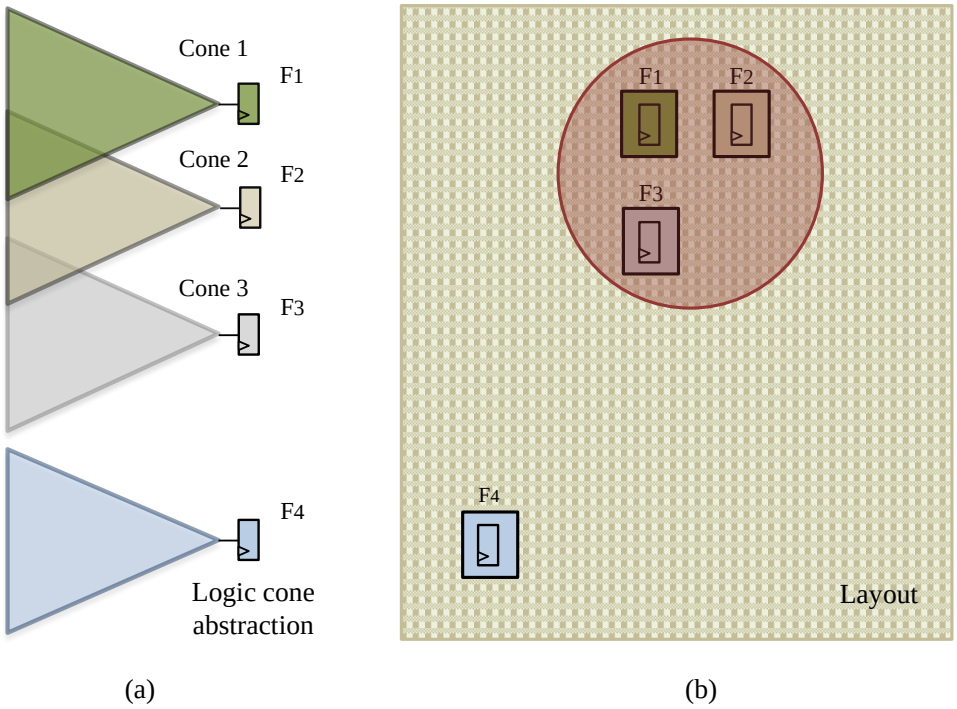


Figure 2.13: Logic cone abstraction versus layout placement of FFs assumption. (a) Logic cone abstraction, (b) Abstracted layout placement: functionally dependent FFs are expected to be placed in the same neighborhood.

2.6 Fault list generation

After the generation of the sets of recipients for all possible attacks, the next step is to generate the fault list which will have to be injected to the circuit so as to evaluate it through fault injection. The fault model so far is able to categorize the FFs which may be simultaneously affected by a localized attack by means of the attack recipient sets. Furthermore, it is compatible with any fault type which may be used to model the erroneous behavior. It is during the fault injection campaign when we will need to select the corresponding fault type (e.g. bit-flip, bit-set, bit-reset), so as to model the effect of the attack at the fault recipients. Even though the imposed constraints have dramatically reduced the corresponding fault space (as we will see in the next section) an exhaustive fault injection is still not possible due to the size

of the fault space. To circumvent this issue, a statistical approach can be used, so as to select samples within each FF set. As in [22], we fix a maximum multiplicity and we sample inside each fault set until we achieve the desirable margin of error, which is the maximum absolute value of statistical error.

2.7 EDA Tool implementation

In order to perform the partitioning, a VHDL and Verilog front-end, provided by Verific Design Automation Inc., was used [36]. An abstract VHDL or Verilog design is first analyzed, followed by RTL elaboration. The result is a netlist database containing basic "Verific" primitives. The C++ API of the front-end is then used to implement the analysis algorithms, of the flow of Figure 2.14. In order to extract the logic cones, we implemented a function that takes as argument one FF of the circuit; its input net is then used as the starting point to perform a depth first search on the netlist [37]. This function is recursive in nature and it explores each branch, until it encounters a primary input or a FF, before backtracking (return to the previous recursion) so as to cover the entire logic cone. Explored nets and combinational instances are stored in memory, having as a reference the father FF of the cone from which the extraction function started.

This procedure is then repeated for all the FFs of the design. After the cone extraction, the sets containing the nets and instances of the cones are processed by means of an intersection function. Each cone is considered as the attack origin of the analysis and the output of this function is the intersection of one cone with all the remaining ones. Since the cones are characterized by their father FFs, this algorithm provides all the sets with the possible attack recipient FFs for the entire design.

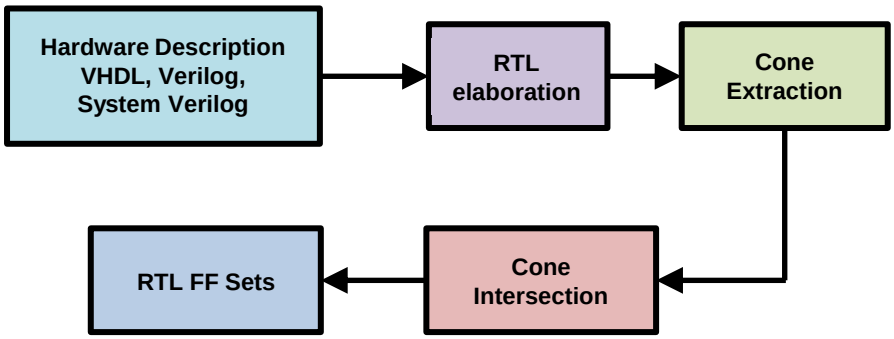


Figure 2.14: EDA tool flow for the extraction of the attack recipient sets (change RTL FF sets with Attack recipient sets)

2.8 Results

As explained in the previous sections the presented methodology results in the categorization of the design's FFs in attack recipient sets. Since the fault model considers that concurrent multiple faults may be selected only from the same attack recipient set, it also leads to characterizing many multiple fault combinations as not possible. In this section we provide results of the methodology for various circuits and the corresponding attack recipient sets for each circuit. Furthermore we will calculate the amount of all possible faults according to our method-

ology (cone method – exhaustive fault list). This will be done to compare the resulting fault space size with the amount of all possible faults when there is no restriction for the selection of multiple faults (exhaustive fault list of any possible fault combination). The target of this analysis is to compare the initial population from which we perform statistical sampling, for our methodology and for the totally random fault injection approach as presented in the state of the art chapter.

In order to generate the fault space that corresponds to an exhaustive analysis with our method, we must take into account each attack recipient set of varying sizes. Thus for each attack recipient set we calculate all possible multiple fault combinations and add all of them all together. In the following, we will use multiple bit-flipping as fault type, to count the size of the fault space, but the methodology can make use of any other fault type that can be injected into FFs. For a circuit containing N FFs, and therefore N cones, we can calculate the size of the fault space of each FF set with exhaustive bit-flipping fault injection as follows:

$$BitFlips_{per\ FF\ set} = (2^k - 1)$$

Where k is the number of FFs in each FF set and $1 \leq k \leq N$.

Since we are considering the fault space for an exhaustive analysis, care has to be taken in order to avoid counting the same fault scenarios (single or multiple) more than once. This may occur since the FF sets may overlap with each other. To avoid this situation, we check all the combinations of these sets. If two sets contain exactly the same FFs, we call them duplicates and consider only one of them. If we encounter one or more sets that are subsets of another larger set, then we reject all the subsets and keep their superset. After this processing, even though we do not remove any possible duplicate fault scenario, it forms a pessimistic upper boundary for the total amount of faults according to the cone method. In the end the implemented tool is able of building the fault space of the method for any given HDL description without any user effort.

At this point we have to emphasize that, as explained in the previous sections, our methodology is targeted to model localized fault attacks, either in the combinational parts of the circuit or directly in FFs. With our analysis we are not attempting to model attacks that are occurring to logic directly related to the clock tree and reset or enable signals. This is due to the fact that such kinds of attacks can affect a very large part of a circuit simultaneously and they are incompatible with the purposes and benefits of laser fault injection. Furthermore, the exact configuration of the clock tree (or other amplification trees) is unknown before placement and routing. Therefore, we have enhanced our tool with the capability to remove from the analysis all the logic that was directly related to reset and enable signals.

Table 2.2 presents the computed fault space sizes for several VHDL designs: a sequential multiplier, the encryption data path of an AES design protected by parity-based error detection [38], the entire AES design, as well as one design from the ITC-99 benchmarks (standard version) [39]. Columns two and three in Table 2.2 contain the number of gates and FFs respectively. The fourth column indicates for each circuit the largest attack recipient set size. The remaining two columns contain the total number of faults for an exhaustive campaign with the RTL cone and the random approaches. We achieve with our method a very large re-

duction of the fault space, of several orders of magnitude compared to the random multi-bit, for all the analyzed circuits. This is dove via considering many multiple fault combinations as not relevant to localized laser fault attacks. In the case of the data-path of an AES implementation an exhaustive analysis needs ~10E+19 fault injections for the cone method and ~10E+154 for the totally exhaustive case. As previously explained for the cone method, these results are upper boundaries for the fault population while for the totally exhaustive case they are the fault populations. For large circuits the sizes of the attack recipient sets may still be large even with the cone method, but the advantages of our method are twofold. Initially we introduce a locality more suitable to describe a localized laser attack and we obtain a large reduction of the population of the fault space which in fact represents this locality. Therefore, if we consider that the assumptions of our methodology are accurate, by statistically sampling our fault space we will obtain results more representative of laser attacks since a huge amount of non-realistic faults has been excluded from the population.

Table 2.2: Fault space sizes

Circuit	Gates	FF	Cone method max recipient set size	Cone method total faults	Random multi bit total faults
Multiplier	3,244	71	17	1.31E+05	2.36E+21
AES parity Data Path	32,312	512	62	2.77E+19	1.34E+154
AES parity	42,156	936	196	1.00E+59	5.81E+281
B18	114,621	3.320	560	7.55E+168	2.63E+999

Figure 2.15 shows the sizes of each of the sets obtained after analyzing the AES Data Path of Table 2.2. The horizontal axis contains the attack recipient FF sets which have remained after checking for duplicate sets or subsets, and the vertical the corresponding set sizes. After this processing only 64 FFs have remained out of the total 512 FFs of the design.

After the extraction of the cone method's fault space the design evaluation can be performed afterwards with a fault injection campaign, either by simulation or emulation. While the cone method fault space has a smaller size than the random fault model, we may still need to utilize statistical fault injection to perform the evaluation within a reasonable simulation or emulation duration. The sets of the fault space of the cone method, has the advantage of containing possible fault combinations with higher locality properties. Thus sampling inside these sets will provide samples which will maintain these locality properties.

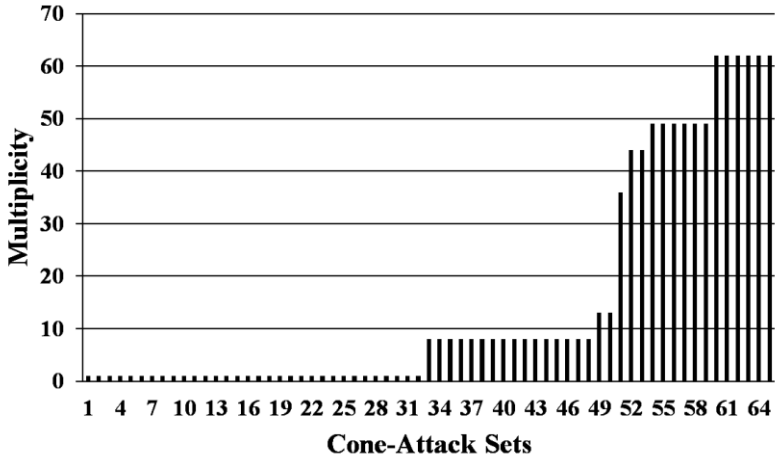


Figure 2.15: Reduced attack recipient sets of the parity protected AES data-path

2.9 Conclusions

One of the main benefits of the methodology presented in this chapter is to compare the described RTL fault injection methodology with the totally random fault injection approach in the context of predicting the locality of laser attacks. The random fault injection approach uses a fixed maximum multiplicity and the set of all the FFs of the design to randomly choose fault samples. This is exactly the same approach which is applied to each of the sets of attack recipients (FF sets) for the RTL cone-based laser fault model. The difference of the random method is that it does not have any attack origin at all. Furthermore, the way that faults get captured to FFs is not defined. Thus, the random method models localized attacks relying exclusively on setting a maximum multiplicity for the fault samples. This does not guarantee that each chosen sample can have any locality relationship. Any locality by random sample selection of even a small multiplicity highly depends on the total number of FFs of a design, since the larger the design the less localized a random sample will be. On the contrary, for the RTL cone-based fault model the set size is dependent on the functional dependencies of the logic of the circuit. This way even if two designs have a big difference in their total number of FFs, the locality is maintained since it depends on each circuit's functional dependencies between its FFs. The difference of the locality characteristics of the two methods will be shown by means of several different implementations in the next chapters.

We have to point out the fact that even though the random method generates a fault space several orders of magnitude larger than the cone method (for exhaustive fault injection), in fact it needs less samples to perform a statistical evaluation. This is due to the fact that the cone-based model includes many sets from which samples have to be chosen, according to a desirable margin of error [22], while the random method has only one. Additionally to achieve a specific margin of error, the sample size is not greatly influenced by the size of each set when the initial population is large [22]. This is a tradeoff of the method, when used in conjunction with statistical fault injection. Besides this, modern fault emulation platforms can easily handle the extra fault samples.

Finally let us first recall the TCAD level example of section 2.3. In this case of low level of abstraction, the attack origin is easier to be defined. At the semiconductor physical level (TCAD level), the origin of an attack is the volume inside the silicon where the charge is induced as the result of laser irradiation. Then, differential equations have to be solved to determine the fault propagation (attack capture) to elements in other levels of abstraction (attack recipients). Besides the fact that this kind of fault injection can be very accurate to model laser attacks, the main goal of the above example is to show that the RTL fault model can have all the attack properties (origin, capture, recipients) defined the same way as they are at lower levels of abstraction. Furthermore, as the RTL design stage occurs early in the design flow, it can assist in avoiding costly feedback loops in the process of evaluating a design, or relevant countermeasures, against laser attacks. In this sense lower levels are very important to validate the RTL model with a bottom-up flow. Furthermore, when the design naturally passes into the other levels of abstraction, it is useful to use the corresponding fault models and cover the situations which are not visible at RTL.

2.10 Contributions

- Definition of a novel RTL fault model describing the locality of laser attacks
- The fault model allows generating multiple fault combinations which are more representative of the locality of laser attacks, especially when compared to the totally random fault model. The fault model therefore manages to guide fault injection campaigns with representative faults, while on the contrary the random approach mainly generates fault combinations which are not possible by means of laser attacks.
- Definition of the locality properties and main assumptions of the fault model
- EDA tool development for the automation of the application of the fault model
- Application of the fault model on multiple secure implementations

3 Fault model evaluation based on RTL fault injection campaigns

In this chapter, we present the results of emulated fault injection campaigns according to the cone fault model and the random fault model. The main goal is to compare the two fault models based on the evaluation of two AES architectures with different countermeasures. Furthermore, we will use known properties of the countermeasures in order to compare the utilized fault models in terms of predicting faults representative of localized laser attacks.

Initially, we will briefly describe the emulator developed at TIMA laboratory to perform the fault injections. Then we will provide further information on the generation of the fault samples according to the applied fault models. Furthermore, we will make use of experimental results, which will be presented in detail at a next chapter in order to determine the fault type which is more appropriate for the evaluation as well as the fault multiplicities for each considered attack.

3.1 Emulated Fault Injection Campaigns at RTL

The main goal of a fault injection campaign is to compare the normal functionality of a design accepting some specific inputs with the functionality of the same design with the same inputs in the presence of faults according to a specific fault model. The two different approaches to perform fault injections at RTL is either by simulation [40], [41], [42], [43]. The main difference between the two approaches is the speed with which faults can be injected and evaluated. Besides the difference, it must be also noted that even though simulation is slower it has the advantage of being able to provide detailed information about the fault propagation inside the circuit. On the other hand emulation based fault injection is a lot faster since the circuit is operating in real time, but it is more difficult to obtain information about the fault propagation inside the circuit. The fault injection emulation platform developed at TIMA laboratory utilizes run-time reconfiguration of an FPGA device in order to inject faults to the normal operation of the circuit.

3.2 Fault types and multiplicities

For all the evaluations we will use the bit-flip fault model. Because of experimental results which will be presented in the next chapter, we have chosen the bit-flip fault type for the fault injection campaigns presented in the next sections. Concerning the multiplicities we will use a maximum multiplicity between 5 and 8 concurrent faults.

The injected faults will be characterized as follows:

- Detected: The output cipher text contains errors and they were detected by the countermeasures.
- Undetected: The output cipher text contains errors but they were not detected by the countermeasures.
- Silent: The output cipher text is correct and the injected errors were not detected by the countermeasures.

- False Positive: The output cipher text is correct but the injected errors were detected by the countermeasures.
- AES Crash: The injected errors lead to a crash during the computation of the cipher text.

3.3 Statistical Sampling of Faults

For complex designs it is true that it is impossible to perform exhaustive fault injection campaigns, even using emulation. Therefore, Statistical Fault Injection (SFI) is mainly used to cope with the huge fault spaces. Reference [22] presents a rigorous approach so as to apply SFI with a quantified margin of error and confidence interval. This approach will also be used for the fault injections presented in the current chapter.

3.3.1 Random Approach Fault Sampling

The random fault injection approach, detailed in [22] uses statistical fault injection by considering that any combination of FFs are possible uniform candidates to capture faults during an attack. Initially a maximum possible multiplicity M_{max} of faults is defined. Then for each multiplicity between one and M_{max} a pseudorandom sampling algorithm (twister) is used to choose an appropriate amount of fault samples (FFs). The fault injection platform must then inject these FF combinations with the fault type that remains to be defined. Furthermore, there is the need to define the injection instants (clock cycles) that each fault must be injected at, as well as the input vectors to the circuit.

We use random selection of the injection instants. The same pseudorandom algorithm is used to select for each fault combination sample the clock cycle it will be injected at (from clock zero up to the last clock of the computation). Concerning the input vectors, the standard test vectors described inside the AES standard have been used. Given the great diffusion properties of AES [44], this does not constitute a limit of the approach.

Because of the random nature of the selected samples, the only way to localize the injected faults is by means of using a small fault multiplicity. Therefore, if for example we limit the fault multiplicity to two FFs being concurrently attacked, we can expect to have a better modeling of the locality than if we use a fault multiplicity of eight.

3.3.2 Cone Method Approach Fault Sampling

As explained during the previous chapter, after the processing of the RTL design by the RTL EDA tools and the application of the cone based approach we obtain all the attack recipient sets according to the cone method's fault model. Besides the huge reduction of the fault space, it was shown in the previous chapter that an exhaustive fault injection campaign is still not possible. Once again we are obliged to make use of statistical fault injection in order to cope with the huge fault space. Even though the cone method also uses statistical fault injection, it cannot be considered as a random method since it does not allow any possible fault combination. This time the statistical methodology, which was earlier applied to the set of all the FFs of the design, is applied inside each of the attack recipient sets. This will inevitably lead to increasing the samples which we will have to evaluate and there will be a tradeoff of the methodology. It is important to note that any success of the cone method to describe local-

ized fault attacks should not be attributed to the increased number of samples as the increase in samples leads only to the reduction of the margin of error. An increase of the random method's samples may lead to the reduction of the margin of error but this will not mean that the random method describes better localized attacks. It will simply mean that we will obtain results with a smaller margin of error concerning random attacks. The final goal of the current chapter is to compare the two fault models in terms of which one is more representative of localized fault attacks.

3.4 Design and countermeasure evaluation results at RTL

In the current section we present the fault injection results on two AES implementations which are protected by different countermeasures. The basic properties of these countermeasures will be presented in the next subsection. We will present first the evaluation results according to the random fault model and then according to the cone fault model for each design. In the last subsection we will present and discuss the comparison of the two approaches.

Since the countermeasures of the two designs under evaluation are already developed and since they were not designed with a focus on protecting the AES specifically against laser fault attacks, we will use the results for two different purposes. We will use the results to compare the two fault models without considering localized attacks.

The main purposes of this evaluation are the following:

- To identify the capabilities of the two fault models of generating multiple fault combinations that may lead to a successful attack.

  The designs will be evaluated by characterizing the nature and the criticality of the errors induced to the circuits due to the faults instructed by the two fault models without any consideration of how well these faults describe localized attacks.

- To utilize properties of the two countermeasures so as to evaluate and compare the capabilities of the two fault models to generate fault samples representative of localized attacks.

3.4.1 Designs under evaluation

The designs which will be evaluated are two AES implementations with different countermeasures. The first is protected by parity code generation with prediction (AES Parity) [45] while the second includes hardware redundancy and several countermeasures against Electromagnetic side-channel attacks (AES Morph) [46].

The AES Parity associates one parity bit for each byte of the AES state matrix. Thus, the parity groups of the parity code are the bytes of the state matrix. Furthermore, a parity prediction rule is implemented for each AES transformation. This translates to 16 parity bits for an AES of 128 parity bits and each parity bit is calculated so as for the parity of a state array byte and its corresponding parity bit to be even. This protection scheme can certainly detect any single bit fault as well as any multiple fault of odd multiplicity, inside the same byte of the AES state matrix. On the contrary it cannot detect an attack if its result is the injection of an even number of faults restricted inside the same parity group. Furthermore, if there is at least

one parity group injected with an odd multiplicity of faults, then the countermeasure will detect the attack. This shows that an adversary who wants obtain the AES output under undetected faults has an interest of attempting to inject faults inside the same parity group so as to bypass the protection. Therefore a fundamental property of this parity based countermeasure is that it is capable to have a better detectability for the odd multiplicities than for the even ones, especially for localized faults.

The basic building block of the AES Morph design is the AES round instance which performs all the round transformations. In order to increase the throughput, the design implements four round instances which are hierarchically equal and connected to a common bus. These round instances can be also used in a hardware redundancy mode. For our analyses only two round instance were used to compute the result of one round. Therefore after the computation of the round these two results were compared to verify if the computation was fault-free. The remaining two round instances were not used and thus they were not active.

The design also includes several countermeasures in order to provide protection against Electromagnetic side-channel attacks. These countermeasures consist in Dynamic Column Relocation (DCR), Dynamic Block Relocation (DBR), Dynamic Mapping (DynMap) and Linear Masking. Additionally since the implemented countermeasures require different random values, the design includes a pseudo random number generator (PRNG).

3.4.2 Design evaluation results based on the random approach

This section presents the results of the emulated RTL fault injection campaigns which are based on the random fault injection approach. Even though the main focus is on multiple fault injections, we have also included exhaustive single fault injection results. The multiple fault samples are selected from the list of all the FFs of the design with multiplicities between one and eight. The limit of a maximum of eight concurrent faults in the presented results was set for the random method. The fault type used is the bit-flip. For all the statistical injection campaigns the margin of error is 5% and the confidence level 95%.

3.4.2.1 Random fault injections on AES Parity

The results of the fault injections are summarized in Table 3.1. The first two rows include single-bit fault injections. The first row includes an exhaustive analysis of all 92.610 faults concerning all FFs and all injection clock cycles.

The second row presents the results of a statistical fault injection campaign, with a random selection of a set of 385 targets and injection clock cycles. For single fault injection we can see that the difference of the percentages of error characterization between the exhaustive and the statistical fault injection are well below the margin of error of 5%. The statistical evaluation of single fault injection indicates that 2.09% of all the injections were undetected while 12.79% were detected. The largest percentage of 62.14% was characterized as silent, while 20.89% of the injections were false positives and 2.09% lead to a crash of the AES.

As concerns the transition to multiple faults, we can see that a multiplicity of two concurrent faults, has a big impact on the percentages of detected and false positive errors by more

than 10% each. At the same time we notice that the silent errors drop from 62.14% for single errors, to 36.88% for a multiplicity of two.

Table 3.1 Injections for single-bit and multiple-bit bit-flips in AES Parity

Injected errors	Number of injections	Silent	Undetected Error	Detected Error	False positive	AES Crash
Exhaustive (multiplicity=1)	92 610	54 590	2 308	13 527	20 936	1 249
		58.94%	**2.49%**	**14.61%**	**22.61%**	**1.35%**
Statistical (multiplicity=1)	383	238	8	49	80	8
		62.14%	**2.09%**	**12.79%**	**20.89%**	**2.09%**
Statistical (multiplicity=2)	385	142	13	98	123	9
		36.88%	**3.38%**	**25.45%**	**31.95%**	**2.34%**
Statistical (multiplicity=3)	385	118	9	132	112	14
		30.65%	**2.34%**	**34.28%**	**29.09%**	**3.64%**
Statistical (multiplicity=4)	385	85	2	171	115	12
		22.08%	**0.52%**	**44.41%**	**29.87%**	**3.12%**
Statistical (multiplicity=5)	385	65	3	172	116	29
		16.88%	**0.78%**	**44.68%**	**30.13%**	**7.53%**
Statistical (multiplicity=6)	385	42	8	181	120	34
		10.91%	**2.08%**	**47.01%**	**31.17%**	**8.83%**
Statistical (multiplicity=7)	385	35	5	202	116	27
		9.09%	**1.3%**	**52.47%**	**30.13%**	**7.01%**
Statistical (multiplicity=8)	385	23	0	211	107	44
		5.97%	**0%**	**54.81%**	**27.79%**	**11.43%**

As multiplicities continue to increase, silent errors decrease as we can also see in Figure 3.1 and from 36.88% (for a multiplicity of two) they drop to 5.97%, for a multiplicity of eight. This shows that the larger the multiplicity, the more likely it is for a fault to impact an active region of the circuit and succeed to propagate an error to the output of the AES computation.

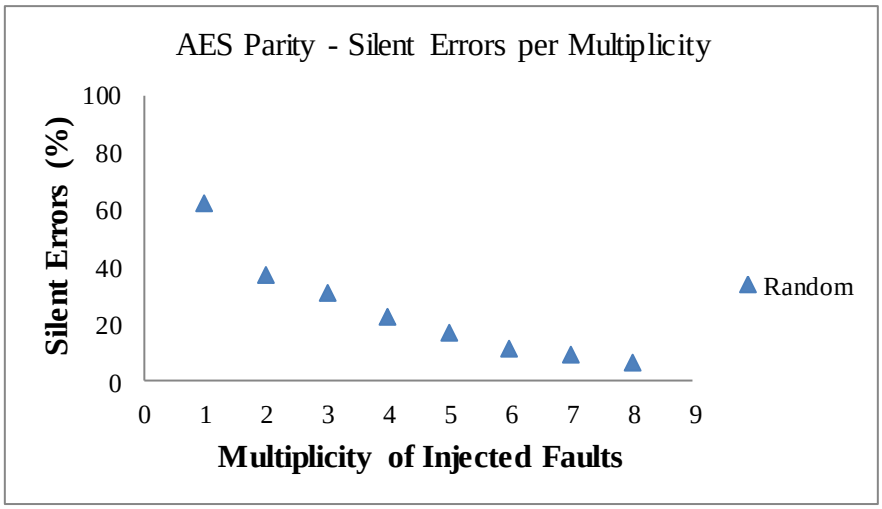


Figure 3.1: AES Parity, silent errors per multiplicity

Undetected errors on the other hand do not exhibit a large variation due to the increase of the multiplicity as in Figure 3.2. Undetected errors increase when we inject faults of a multiplicity of two up to 3.38%. While this is reasonable due to the nature of the countermeasure (group parity) we have to note that this is not the case for the remaining even multiplicities of four and six injected faults. Even though it is expected of even multiplicities to consistently lead to higher multiplicities of undetected faults, this is not the case as multiplicities of 4 and 8 have undetected rates close to zero. One explanation is due to the fact that these differences in the percentages are small in comparison with the margin of error of 5%. Another explanation can be that the random selection amongst all the FFs of the design, leads to the injection of a single bit error to at least one parity group, as the multiplicity increases. If the latter is the case, then we can see that such an evaluation may lead to a false assurance of security.

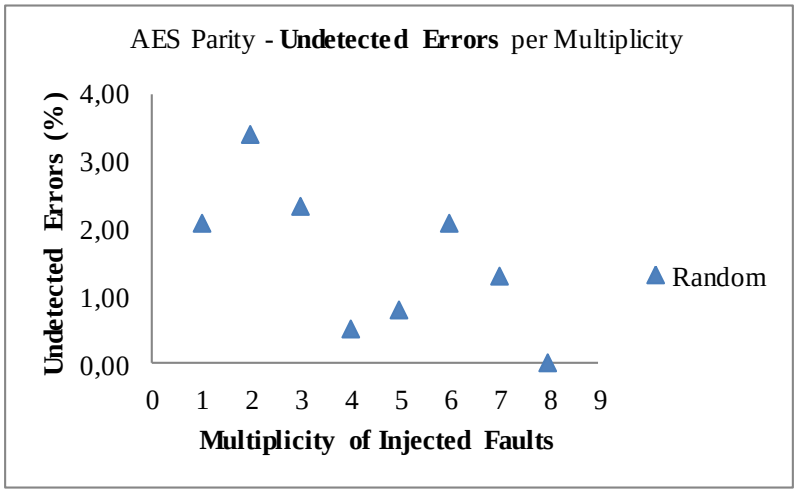


Figure 3.2: AES Parity, undetected errors per multiplicity

This conclusion is further supported by the increase of the detection rates with the increase in multiplicities shown in Figure 3.3, as it is apparent that even or odd multiplicities do not affect the detection rates differently. In fact the dominant tendency is that the larger the multiplicity the easier the detection is. This can be explained since the bigger the injected multiplicity, the higher the probability is to affect one protected register with an odd number of errors.

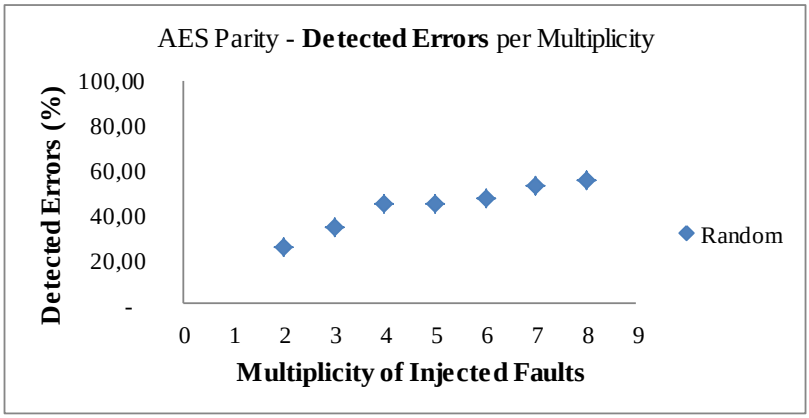


Figure 3.3: AES Parity, detected errors per multiplicity

The false positive rates of Table 3.1 occur due to the fact that when errors are injected during the inactive clock cycles of a protected pipeline, it leads to the detection of an error even though the fault will not be able to propagate to the AES output. As we can see in Figure 3.4 false positive errors remain constant as multiplicities increase.

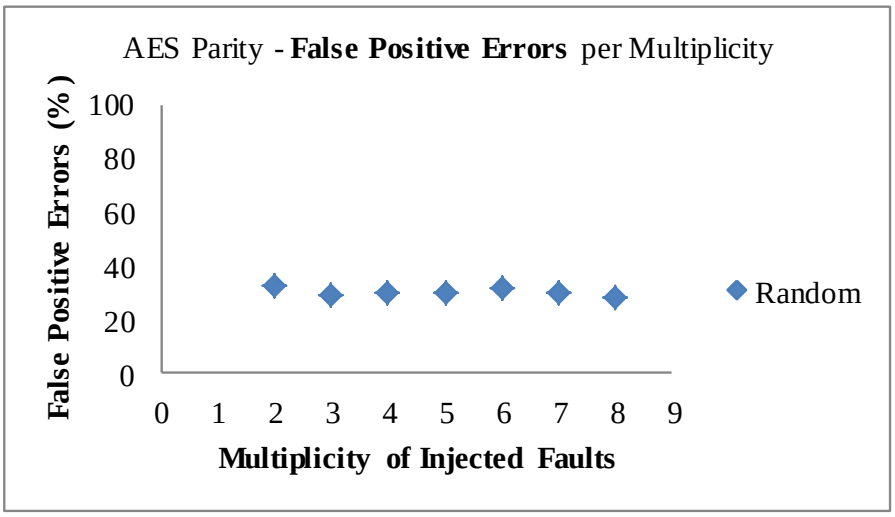


Figure 3.4: AES Parity, false positive errors per multiplicity

The last column of Table 3.1 contains the number of crashes, mainly due to injections in control logic. As we can see in Figure 3.5 crash errors have the tendency to increase with the multiplicity since, due to the random selection of the targets, the bigger the multiplicity, the higher the probability to include one critical control flip-flop.

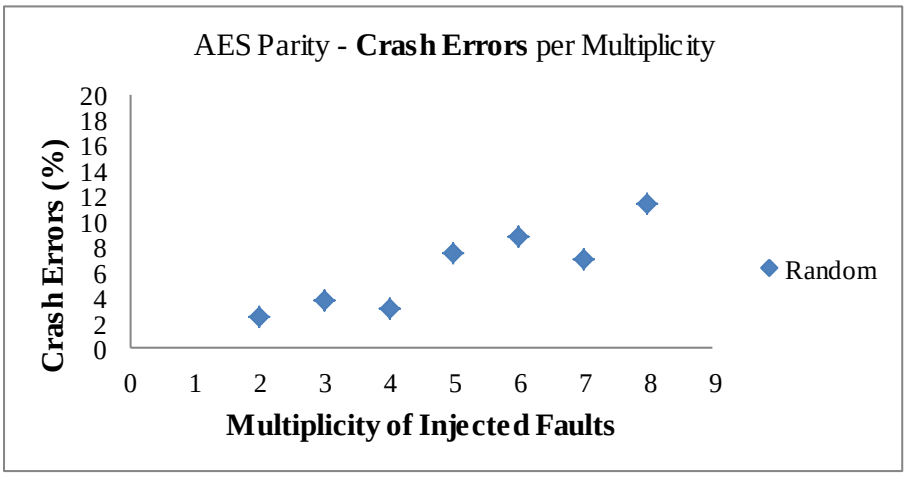


Figure 3.5: AES Parity, crash errors per multiplicity

The results of the random fault injection campaign on the parity protected AES architecture showed that the undetected errors are very few, if we take into account that parity cannot detect any even multiplicity injected in the same parity group. However, we have to emphasize the fact that the faults are injected randomly with a uniform distribution across all the design's FFs. Therefore, the probability of generating multiple errors of strictly even multiplicities inside any of the parity groups is very small. Even though there exists a decreasing trend in the percentage of undetected errors for increasing multiplicities (which is compatible with the uniform injection model), there are some exceptions for some of the even multiplicities of injected faults. This means that the design is robust against random faults since such faults cannot highlight the weakness of the countermeasure, which is the injection of strictly even faults inside the parity groups.

The largest percentage of single bit errors lead to a correct output of the AES (error-free = silent + false positive = 83.03%). This is mainly due to the fact that many of the single faults have a small lifetime since the elements they are injected in are not active at the time of the injection. This percentage of faults that lead to no error can form a metric of the robustness of a design against fault attacks early in the design flow. Since the error-free percentages are mainly affecting inactive components, then a multiple fault which does not affect the computation is more likely to be confined in one specific region of the design. On the contrary random multiple faults with a large dispersion between different RTL parts of a design have a higher probability to affect at least one active component. Due to the dispersion of random faults, a multiple fault may affect at the same time an active and an inactive part of the design. This leads to the possible propagation of the injected fault only in the fanout of the active region.

In Figure 3.6 we can see the rapid decrease of the error-free percentages as the multiplicity increases and from 83.03% for single bit errors, it falls to 33.76% for eight concurrent faults.

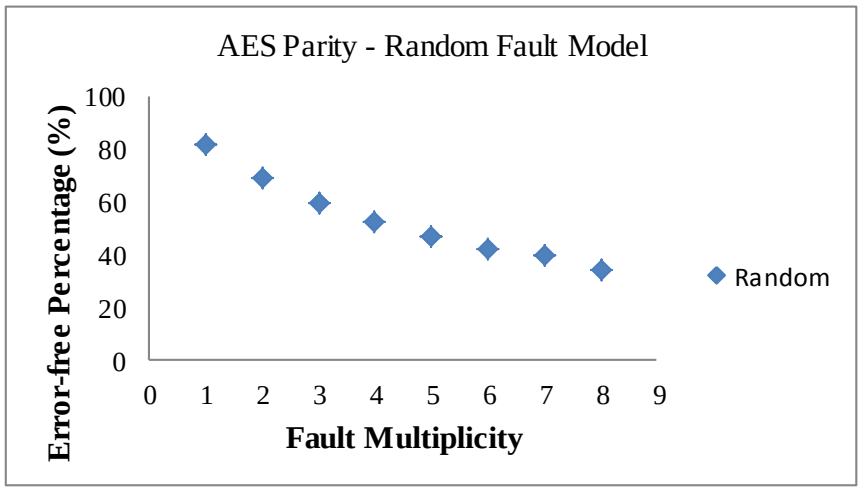


Figure 3.6: AES Parity, error-free injections per multiplicity

3.4.2.2 Random fault injection on AES MORPH

For the AES Morph the fault injection campaign was performed similarly to the campaign described in the previous sections. Once again the fault type used was the bit flip and the multiplicity varied between one and five. The injection time was also chosen randomly for each of the fault samples. For this injection campaign only one data block was loaded to the AES and therefore it was allocated randomly on two (out of the four round instances the design contains) independent instances for verification. All results based on statistical fault injection have been obtained for a margin of error of 5% in order to limit the emulation duration and a confidence of 95%.

Table 3.2 presents the results of the fault injection campaign. Single bit injections have been performed both exhaustively and according to the statistical fault injection approach. All the error percentages of these two (single bit) campaigns have produced similar results, within the margin of error, besides the silent errors which differ approximately by 10%. This can be explained though by the small number of injections during the statistical injections (~385 injections) and the confidence level. During the single bit fault injection campaign once again we can notice that the silent class of errors is the dominant with a 75.85% for the exhaustive campaign (or 65.97% for the statistical campaign). This large percentage is mainly due to injecting faults to inactive regions of the design.

Table 3.2 Injections of bit-flips with multiplicity up to 5 in AES Morph with the erroneous bits uniformly spread into all flip-flops

Multiplicity	Silent	Undetected error	Detected error	False positive	AES Crash
1 (Exhaustive)	75.85%	3.85%	10.1%	9.39%	0.81%
1 (Statistical)	65.97%	8.82%	9.87%	13.66%	1.68%
2 (Statistical)	50.4%	9.44%	21.89%	17.27%	1%

3 (Statistical)	**40.57%**	**8.49%**	**32.07%**	**17.69%**	**1.18%**
4 (Statistical)	**27.48%**	**9%**	**41.81%**	**19.63%**	**2.08%**
5 (Statistical)	**19.03%**	**10.62%**	**49.11%**	**17.92%**	**3.32%**

If we take into account the statistical injections we can see that as we increase the error multiplicity the percentages of silent faults is gradually decreasing as in Figure 3.7 from 65.97% (for single bit injections) to 19.03% (for injections of 5 concurrent faults). Since the targets are randomly chosen this behavior is normal because as multiplicity increases it is more likely to target at least one active region of the design. On the other hand as we can see in Figure 3.8 the percentage of detected errors is increasing as we move from smaller to higher multiplicities of injected targets. Detection rates start from 9.87% for single bit injections and reach 49.11% for a multiplicity of five. Thus as we increase the multiplicity we can see that the silent errors are becoming detected. This is due to the increasing probability of injecting an error in the instance actually used at the injection cycle to perform either the useful computation or its verification.

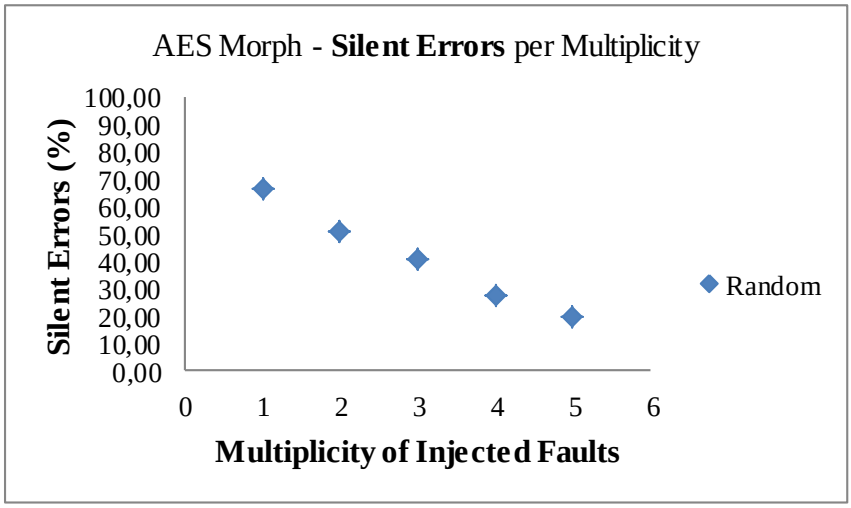


Figure 3.7: AES Morph, silent errors per multiplicity

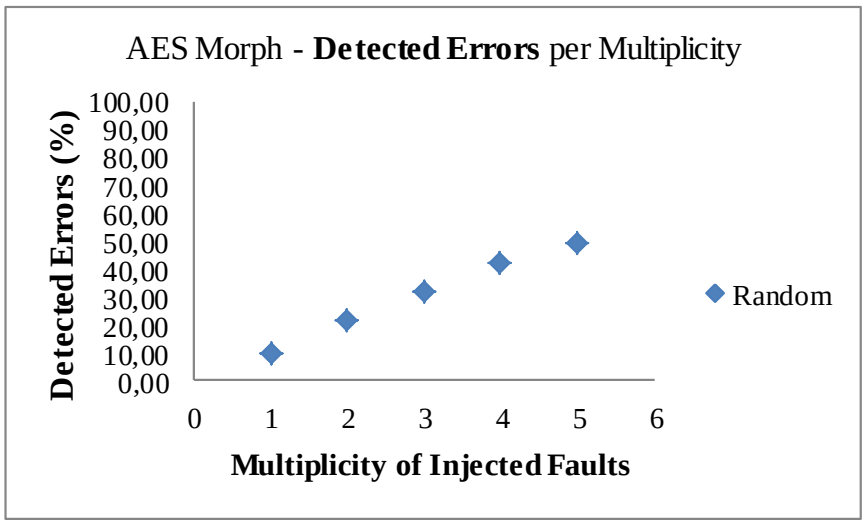


Figure 3.8: AES Morph, detected errors per multiplicity

Undetected errors, as can be seen in Figure 3.9, are stable and around 9% for all injection multiplicities. The same tendency holds for the false positive errors which are stable around 17%, as can be seen in Figure 3.10. Crash errors are also stable and they are included in Figure 3.11.

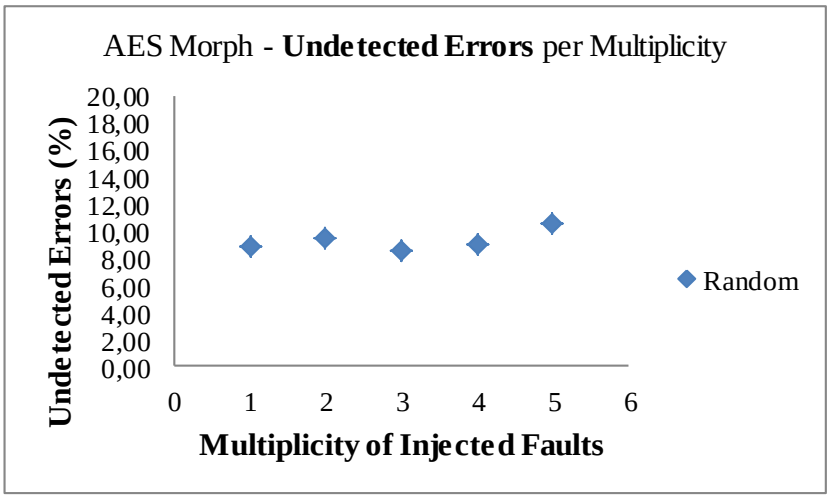


Figure 3.9: AES Morph, undetected errors per multiplicity

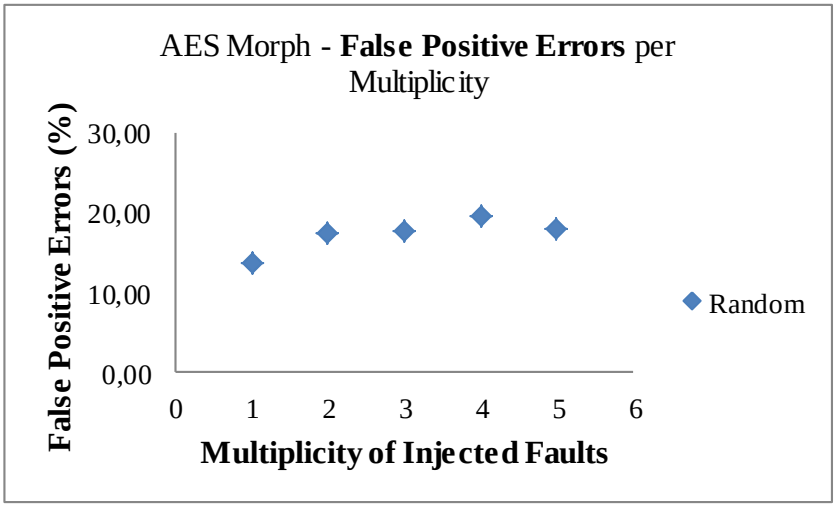


Figure 3.10: AES Morph, false positive errors per multiplicity

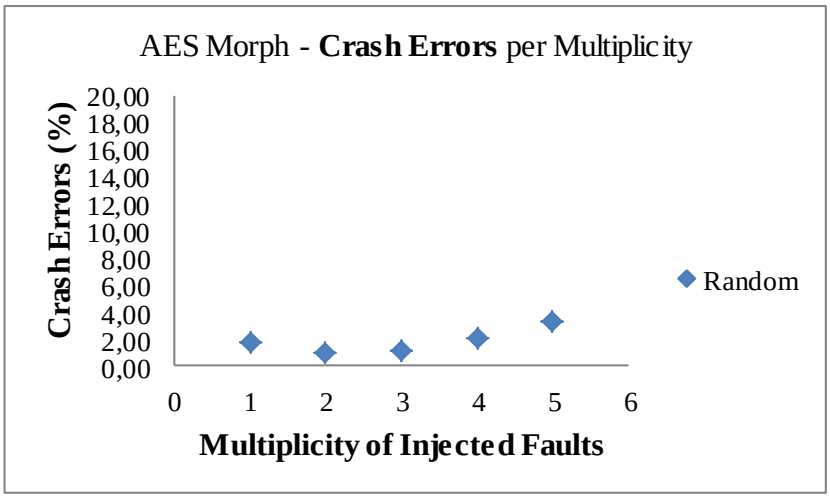


Figure 3.11: AES Morph, crash errors per multiplicity

As in the case of the AES Parity design we will also use for the Morph design the error-free percentages of the injections so as to discuss the capability of the random fault model to describe attacks which target specific functional blocks of a design. As can be seen in Figure 3.12 for single bit errors 79.63% of them are error-free. As multiplicities rise we notice a decrease of the error-free percentages up to 36.95%, for a multiplicity of five targets. Once again we can notice that single bit errors, which describe the largest possible locality at RTL, produce a high error-free rate of 79.63% while the transition to two concurrent injection targets reduces the error-free rate by 11.96% and this tendency continues almost linearly as multiplicities decrease.

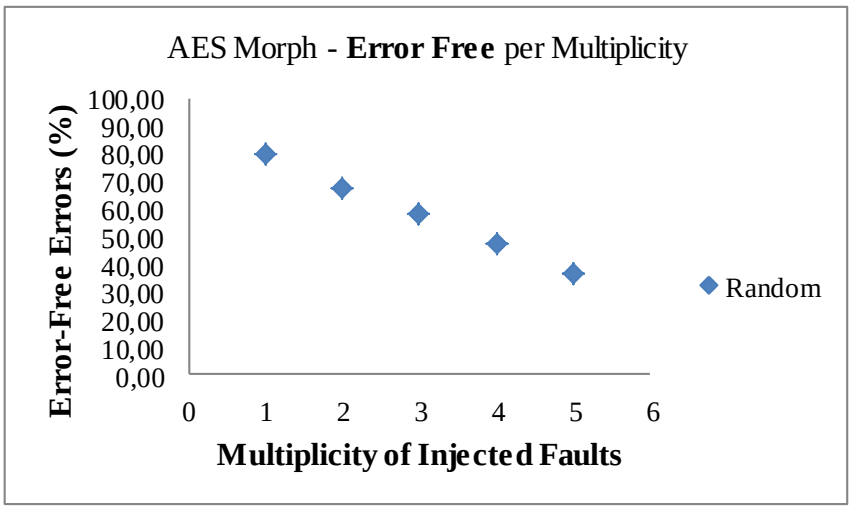


Figure 3.12: AES Morph, Error Free percentages per multiplicity

We notice from the above that since error-free percentages are mainly affecting inactive components, then a multiple fault which does not affect the computation is more likely to be confined in one specific section of the design. On the contrary random multiple faults with a large dispersion between different RTL parts of a design have a higher probability to affect at least one active component. Due to the dispersion of random faults, a multiple fault may affect at the same time an active and an inactive part of the design. This makes the propagation of the injected fault to the outputs of the computation more probable since the error is placed in the fanin of an active region.

3.4.3 Design evaluation results based on the cone fault model

For each of the designs presented in the previous sections (AES Parity and AES Morph) in this section we will present the fault injection campaign results according to the cone fault model. The multiple fault samples are selected from the list of all the FFs of the design with multiplicities between 1 and 8. The fault type used is the bit-flip. For all the statistical injection campaigns the margin of error is 5% and the confidence level 95%. As already explained, a statistical injection is taking place in each attack recipient set of the cone fault model and we calculate for each multiplicity the average for all the attack recipient sets.

3.4.3.1 Cone method fault injection on AES Parity

The results of the fault injections are summarized in Table 3.3.The first two rows include single-bit fault injections, which are the same as in the section presenting the random injections. The first row includes an exhaustive analysis of all 92.610 faults concerning all FFs and all injection clock cycles.

Table 3.3 Injections of bit-flips with multiplicity up to 8 in AES Parity with the erroneous bits limited to attack recipient FFs, modelling the attack locality

Multiplicity	Silent	Undetected error	Detected error	False positive	AES Crash
1 (Exhaustive)	58.94%	2.49%	14.61%	22.61%	1.35%
1 (Statistical)	62.14%	2.09%	12.79%	20.89%	2.09%
2 (Statistical)	52.24%	8.39%	12.38%	25.55%	1.44%
3 (Statistical)	42.23%	1.29%	20.07%	34.64%	1.77%
4 (Statistical)	50.93%	8.21%	14.32%	24.44%	2.1%
5 (Statistical)	41.63%	0.95%	21.67%	33.53%	2.22%
6 (Statistical)	49.61%	7.96%	17.82%	22.21%	2.4%
7 (Statistical)	37.92%	0.69%	23.68%	34.98%	2.73%
8 (Statistical)	27.62%	5.92%	29.33%	33.31%	3.82%

As can be seen in Figure 3.13 the silent errors for single bit errors start from 58.94% for the exhaustive campaign and 62.14% for the statistical. When the multiplicity is increased to two concurrent faults, silent faults are reduced to 52.24%. The percentages are quite stable as the multiplicities increase, with silent error rates being higher for even multiplicities. A gradual decrease takes place after exceeding six concurrent faults to 27.62% for a multiplicity of eight faults. This shows that as the number of concurrent targets increases, the injections do not necessarily affect active regions of the design. The cone fault model tends to inject faults inside either active or non-active functional blocks of the design. Therefore it is the assumptions of the cone fault model which are able to constrain fault injection to specific functional blocks rather than spread them in the entire design.

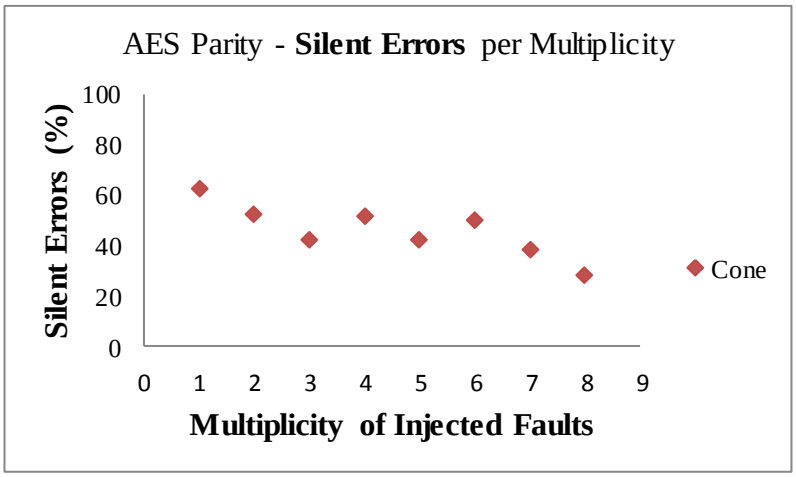


Figure 3.13: AES Parity, silent error percentages per multiplicity

Undetected errors are presented in Figure 3.14. It is noticeable that there exists a clear difference between the odd and even multiplicities, regarding these rates. Single bit injections lead to 2.09% of undetected errors. When multiple faults are introduced, we can see that for all even multiplicities we obtain larger percentages than for the odd ones. A multiplicity of two concurrent faults leads to 8.39% of undetected errors, while the remaining even multiplicities of 4, 6 and 8 concurrent faults lead respectively to 8.21%, 7.96% and 5.92% of undetected error rates respectively. On the other hand multiple odd multiplicities start from 2.34% for three concurrent faults and gradually decrease to 0.95% and 0.69% for multiplicities of 5 and 7 respectively. It is therefore evident that by injecting faults inside the same attack recipient sets we were able to characterize the parity-based countermeasure's weakness, which is its inability to perform equally well under odd and even error multiplicities.

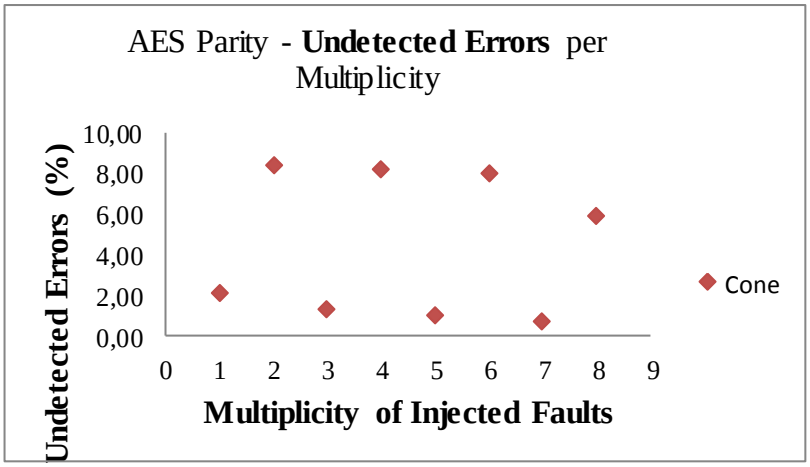


Figure 3.14: AES Parity, undetected error percentages per multiplicity

Detected errors are presented in Figure 3.15. The detection rate of single bit errors is 12.79% and they gradually rise to 29.33% for a multiplicity of eight concurrent faults. Once again we can see that odd multiplicities have a higher detection rate than even multiplicities. Also as multiplicities increase, detection is higher since more parity groups are involved and thus it is more probable for at least one of them to detect an error.

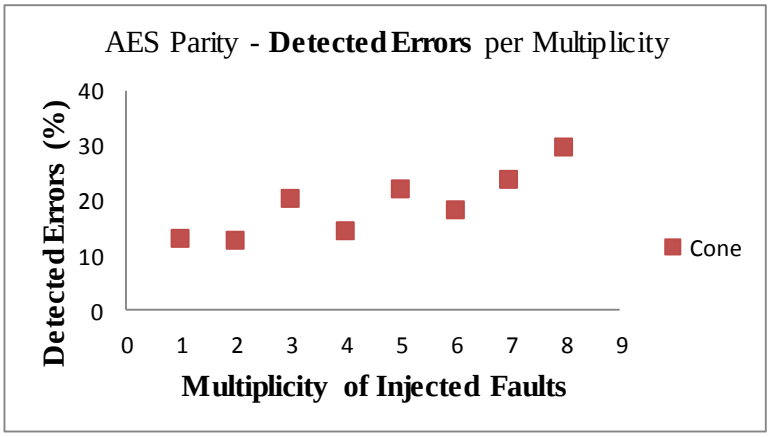


Figure 3.15: AES Parity, detected error percentages per multiplicity

False positive errors are in general stable with increasing multiplicities and they are approximately 24% for even multiplicities and 34% for odd multiplicities as presented in Figure 3.16.

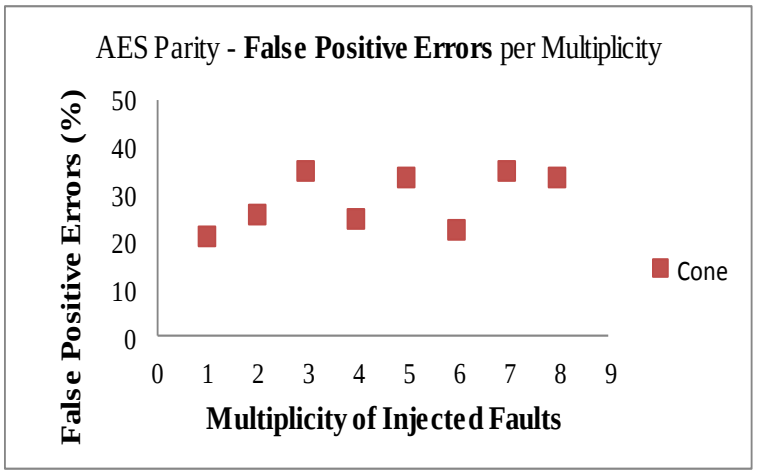


Figure 3.16: AES Parity, false positive error percentages per multiplicity

Finally, crash errors also exist and they have the tendency to increase with increasing multiplicities as in Figure 3.17. This tendency is expected as the higher the multiplicity the more probable it is to target control FFs, which may lead to a crash error.

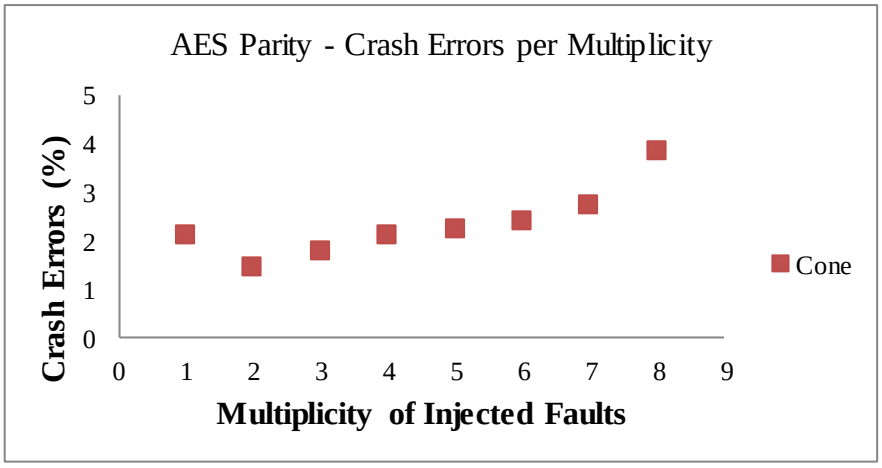


Figure 3.17: AES Parity, crash error percentages per multiplicity

Figure 3.18 presents the error-free (the sum of silent and false positive errors) percentages of the campaign. They have an average of 75% for the multiplicities of two up to seven and they are rather stable with the exception of the multiplicity of eight concurrent faults when the error-free rate drops suddenly to 60.93%. This result shows that the fault combinations generated by the cone fault model do not target both active and inactive regions at the same time, as multiplicities increase. If the target cone which generates the corresponding attack recipient set has a fanout which is inactive (during the selected injection clock cycle) then the faults

will not be able to propagate to the circuit outputs. This illustrates that the cone fault model is capable to contain the injected faults to local regions of the design.

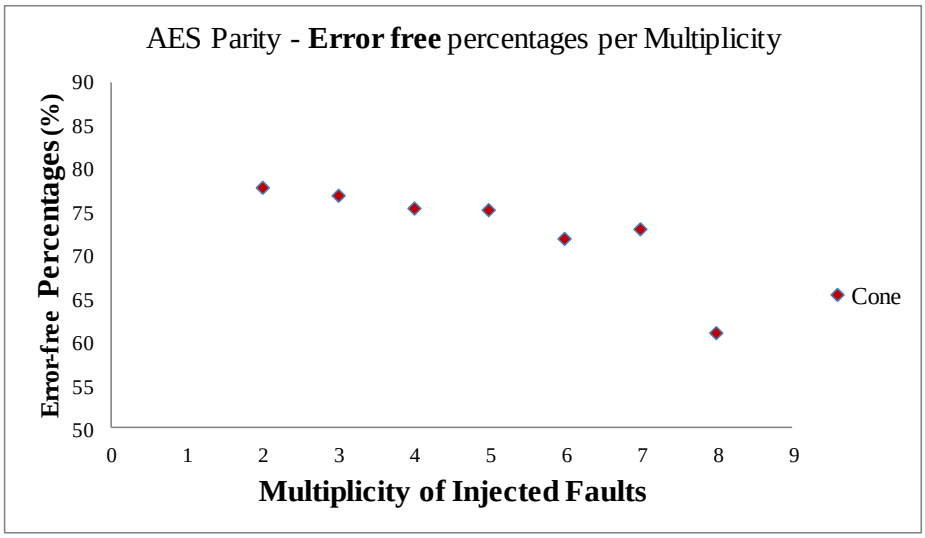


Figure 3.18: AES Parity, error-free percentages per multiplicity

3.4.3.2 Cone method fault injection on AES Morph

This section presents the fault injection campaign performed on the Morph AES design according to the cone fault model. The results are included in Table 3.4, including multiple fault injections from two up to five concurrent faults, as well as the results of the exhaustive and a statistical campaign regarding single bit faults.

Table 3.4 Injections of bit-flips with multiplicity up to 5 in AES Morph with the erroneous bits limited to the attack recipient sets modelling local attacks

Multiplicity	Silent	Undetected error	Detected error	False positive	AES Crash
1 (Exhaustive)	75.85%	3.85%	10.1%	9.39%	0.81%
1 (Statistical)	65.97%	8.82%	9.87%	13.66%	1.68%
2 (Statistical)	63.09%	6.79%	15.4%	13.13%	1.59%
3 (Statistical)	58.21%	6.02%	19.79%	14.54%	1.44%
4 (Statistical)	54.04%	5.76%	24.7%	14.11%	1.39%
5 (Statistical)	52.3%	5.6%	28.03%	13.35%	0.72%

Figure 3.19 presents the error percentages which belong to the silent class. We can see that the multiplicity of two concurrent faults generates 65.97% of silent errors while for five faults it drops to 52.3% and therefore we obtain a 13.67% decrease.

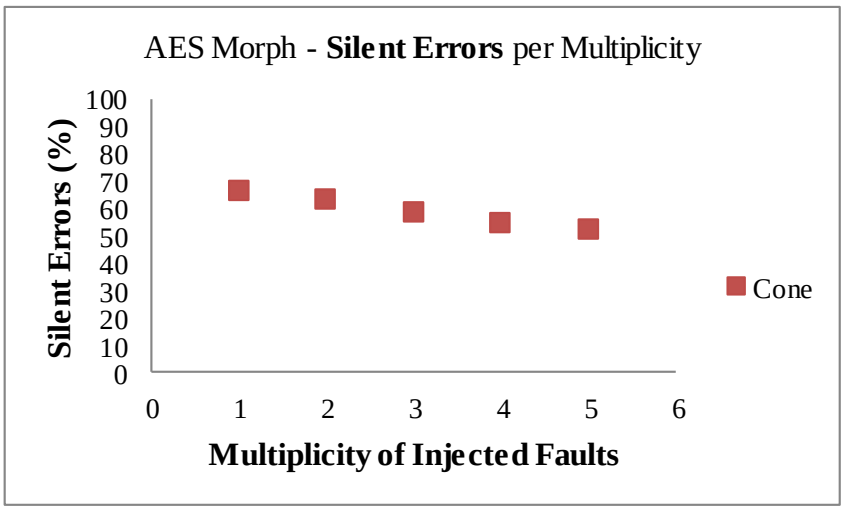


Figure 3.19: AES Morph, silent error percentages per multiplicity

Undetected errors are rather stable with an average percentage of 6.6%, as can be seen in Figure 3.20.

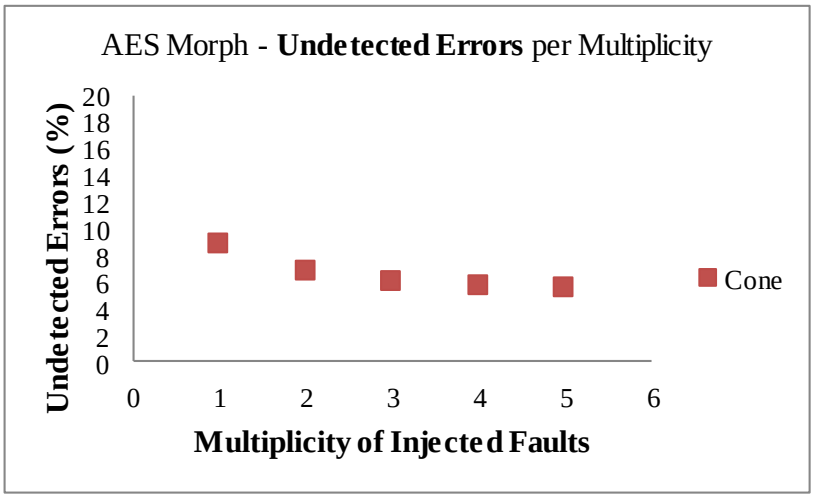


Figure 3.20: AES Morph, undetected error percentages per multiplicity

The detection rates start from 10% for the single bit fault injection and they have the tendency to increase as multiplicities rise. Single bit injections generate a detection error rate of 9.87% while for the injections of five concurrent faults we obtain 28.03% of detected errors. By combining these results with the results of silent errors we can deduce that as the multiplicity increases silent errors tend to become detected.

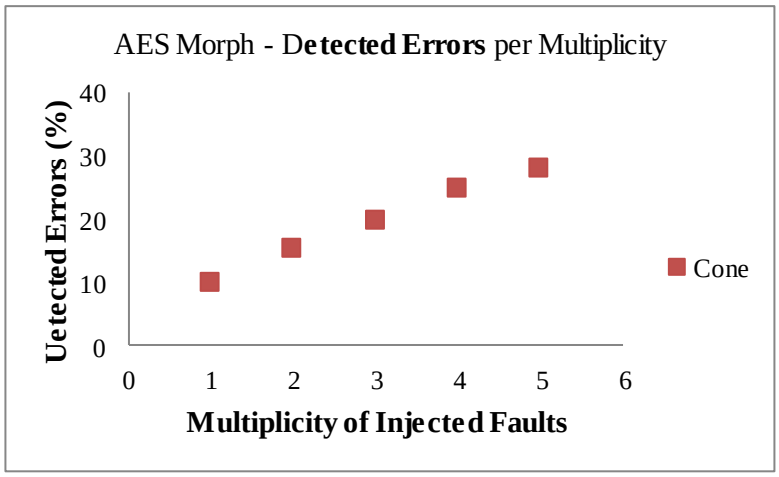


Figure 3.21: AES Morph, detected error percentages per multiplicity

False positive and crash errors are graphed in Figure 3.22 and Figure 3.23 and they are very stable with increasing multiplicities. They generate 13.76% and 1.36% of false positive and crash, average error rates respectively.

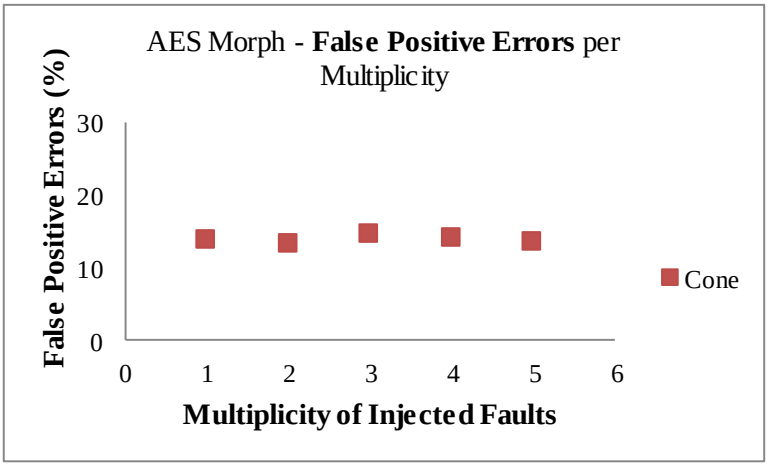


Figure 3.22: AES Morph, false positive error percentages per multiplicity

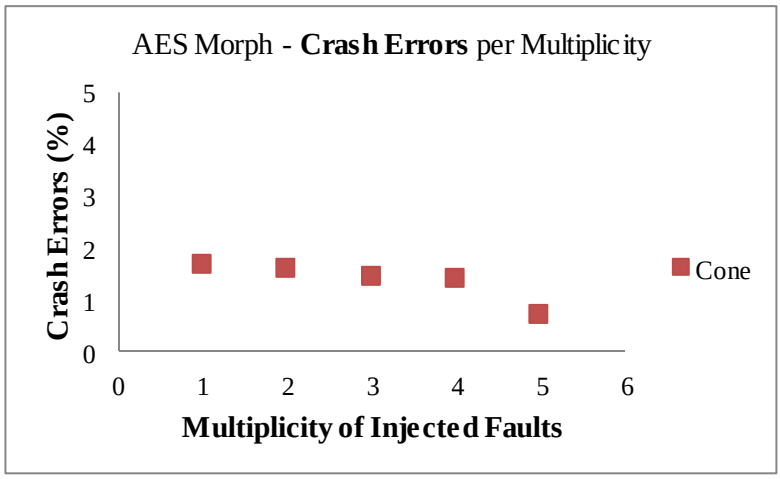


Figure 3.23: AES Morph, crash error percentages per multiplicity

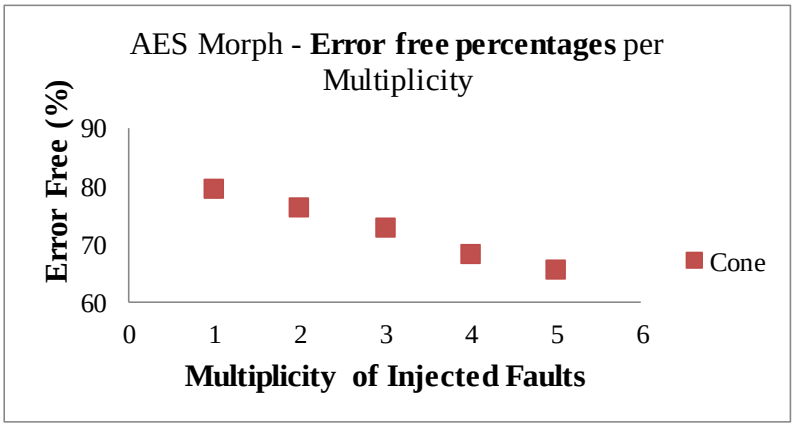


Figure 3.24: AES Morph, error free percentages per multiplicity

When we consider the error-free percentages, in Figure 3.23, we can see that single bit injections generate a percentage of 79.63% while a multiplicity of five concurrent errors 65.65%. Therefore we notice a drop, between multiplicities one and five, of 13.98%.

3.4.4 Comparison of the two approaches

In the current section we will use the results presented in the previous sections to qualitatively compare the two fault injections campaigns and discuss their differences in terms of evaluating the designs.

3.4.4.1 Comparison of results for AES Parity

In Table 3.5 we present at the same time the results obtained with the random and the cone fault models for the AES Parity design. Furthermore in Figure 3.25 we have included the graphs of the results according to the random and cone fault models so as to compare the results.

Table 3.5 Injections of bit-flips with multiplicity up to 8 in AES Parity with the erroneous bits uniformly spread into all flip-flops or limited inside the attack recipient sets modelling the attack locality

Multiplicity	Injection campaign	Silent	Undetected error	Detected error	False positive	AES Crash
2	All FFs	36.88%	3.38%	25.45%	31.95%	2.34%
	Local groups	52.24%	8.39%	12.38%	25.55%	1.44%
3	All FFs	30.65%	2.34%	34.28%	29.09%	3.64%
	Local groups	42.23%	1.29%	20.07%	34.64%	1.77%
4	All FFs	22.08%	0.52%	44.41%	29.87%	3.12%
	Local groups	50.93%	8.21%	14.32%	24.44%	2.1%
5	All FFs	16.88%	0.78%	44.68%	30.13%	7.53%
	Local groups	41.63%	0.95%	21.67%	33.53%	2.22%
6	All FFs	10.91%	2.08%	47.01%	31.17%	8.83%
	Local groups	49.61%	7.96%	17.82%	22.21%	2.4%
7	All FFs	9.09%	1.3%	52.47%	30.13%	7.01%
	Local groups	37.92%	0.69%	23.68%	34.98%	2.73%
8	All FFs	5.97%	0%	54.81%	27.79%	11.43%
	Local groups	27.62%	5.92%	29.33%	33.31%	3.82%

Figure 3.25(a) contains the results for silent errors, where we can see a clear difference in the two approaches. The random fault model has silent error rates depending on the multiplicity of the injected faults. On the contrary, when we used the cone fault model the design had more stable silent fault rates. Concerning the false positive errors (Figure 3.25 (d)) we can notice that for both fault models the rates are constant (within the margin of error) and independent of the multiplicity of concurrent injected faults. More precisely, the results show that the percentage of silent faults with respect to error multiplicity is much closer to the cone fault model than to the random fault model, even in the case of a quite large spot with respect to the transistor size.

Figure 3.25 (f), shows the error-free rates, which are the sum of these two classes of errors. The rates of the random fault injection follow the behavior of the silent class, while the cone fault model produced almost constant error-free rates. As already discussed in the previous sections this shows that the cone method is more capable to constrain the injected faults in a specific functional region of the parity design. This can be deduced by the fact that error-free rates are involving fault injection in inactive regions of the circuit. Therefore the constant rates show that the faults generated by the cone fault model do not increase the probability to affect an active target as the multiplicities rise.

Undetected error rates of Figure 3.25 (b), show that the cone fault model is able to clearly indicate to the designers that the countermeasure under evaluation has not the same capabilities of detecting odd and even injected faults. This is coherent with a protection based on parity. Furthermore the random approach, even though it also shows some differences between odd and even multiplicities (although not consistent), the error rates for many multiplicities are very close to zero. On the other hand the cone method clearly indicates that the undetected faults are in fact not very close to zero for both odd and even multiplicities of injected faults. This analysis shows that the random fault model may in this case lead to a false assurance of

security since it produces results which make the basic disadvantage (that the detection of odd and even faults is different) of the countermeasure disappear. Such a false assurance of security may lead the designers to believe that the implemented countermeasures are sufficient for a specific application and move on to the design stages which follow RTL (synthesis, placement, routing and finally fabrication). Such a decision may create an unnecessary design respin.

Figure 3.25 (c), shows that a clear difference between the two fault models also exists in the case of the error detection rates. The random approach indicates higher detection capabilities as the multiplicities rise while the cone method produces more stable results. This difference is important since it may also lead to a false assurance of security when using this specific countermeasure.

Crash error rates also have differences when they are determined by the two fault models. For large multiplicities the random approach manages to often affect control unit related FFs and thus it indicates that the design will often crash when attacked. On the contrary the cone fault model indicates that increasing the fault multiplicity increases the crash rates but by a maximum percentage of 3.82%, instead of 11.43% when the design is evaluated by the random fault model.

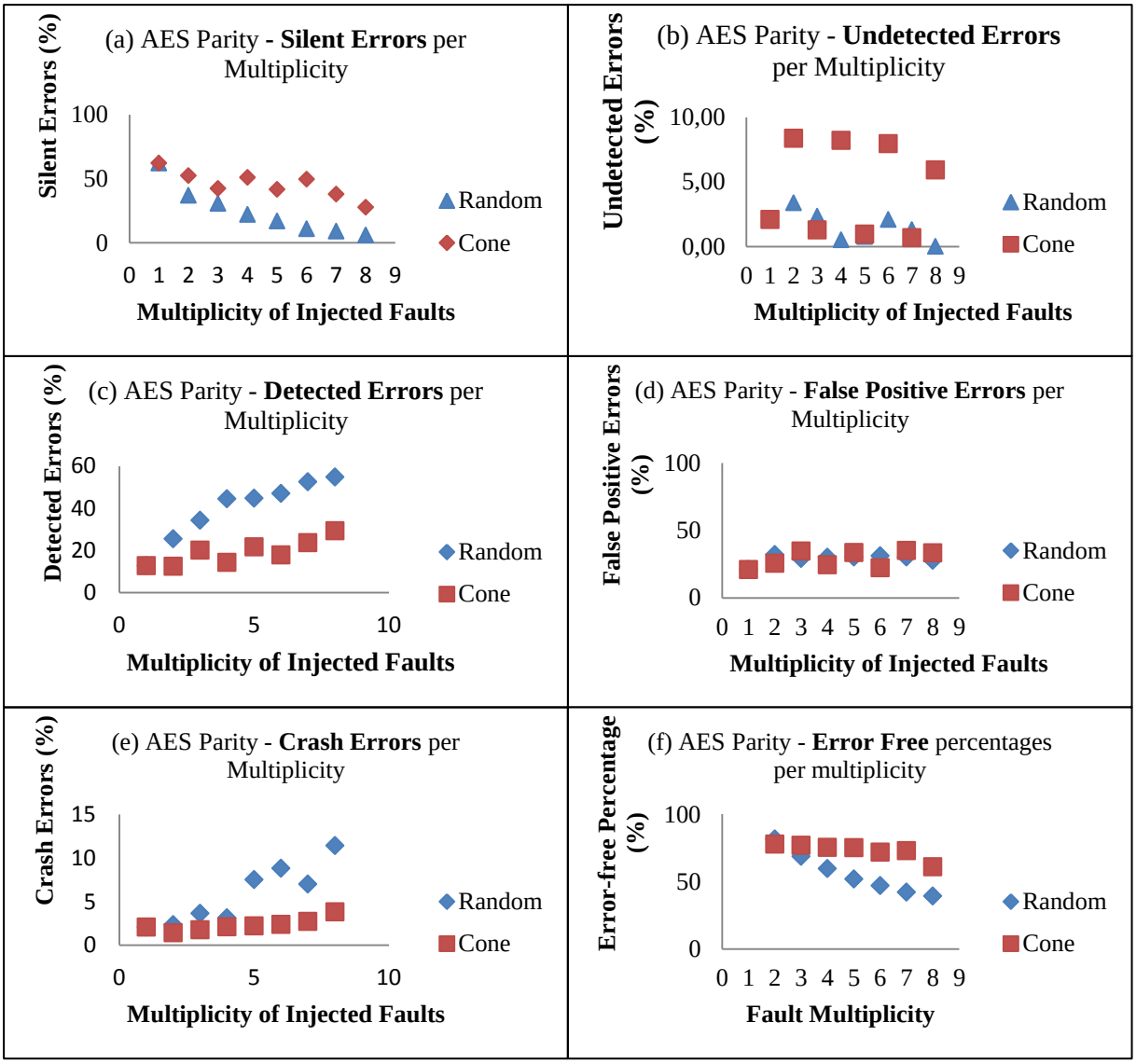


Figure 3.25: AES Parity, silent error percentages per multiplicity for the Random and Cone fault models

3.4.4.2 Comparison of results for AES Morph

In Table 3.6 we present at the same time the results obtained with the random and the cone fault models for the AES Parity design. Figure 3.26 contains the graphs of the error rates for the injections of the AES Morph design according to the two fault models.

Figure 3.26 (a) contains the silent error rates for the two fault models. The results have similar characteristics with the relevant results of the AES Parity design. We can clearly see that the cone fault model provides more stable results as multiplicities rise, while in the case of the random fault model silent errors are a lot more dependent on the multiplicity. Once again, this shows that the random fault model, for large multiplicities, has a bigger impact on the functionality of the design and it manages to impact active regions which propagate the injected faults to the output. Concerning the false positive error rates, included in Figure 3.26, we can also see that the cone method shows a more stable behavior.

These two error rates combined produce the error-free rates included in Figure 3.26 (f). The same tendency is again evident as, with increasing multiplicities, the random fault model's error-free rates drop faster than the corresponding rate of the cone fault model.

Table 3.6 Injections of bit-flips with multiplicity up to 5 in AES Morph with the erroneous bits uniformly spread into all flip-flops or limited to specific groups modelling the attack locality

Multiplicity	Injection campaign	Silent	Undetected error	Detected error	False positive	AES Crash
2	All FFs	50.4%	9.44%	21.89%	17.27%	1%
	Local groups	63.09%	6.79%	15.4%	13.13%	1.59%
3	All FFs	40.57%	8.49%	32.07%	17.69%	1.18%
	Local groups	58.21%	6.02%	19.79%	14.54%	1.44%
4	All FFs	27.48%	9%	41.81%	19.63%	2.08%
	Local groups	54.04%	5.76%	24.7%	14.11%	1.39%
5	All FFs	19.03%	10.62%	49.11%	17.92%	3.32%
	Local groups	52.3%	5.6%	28.03%	13.35%	0.72%

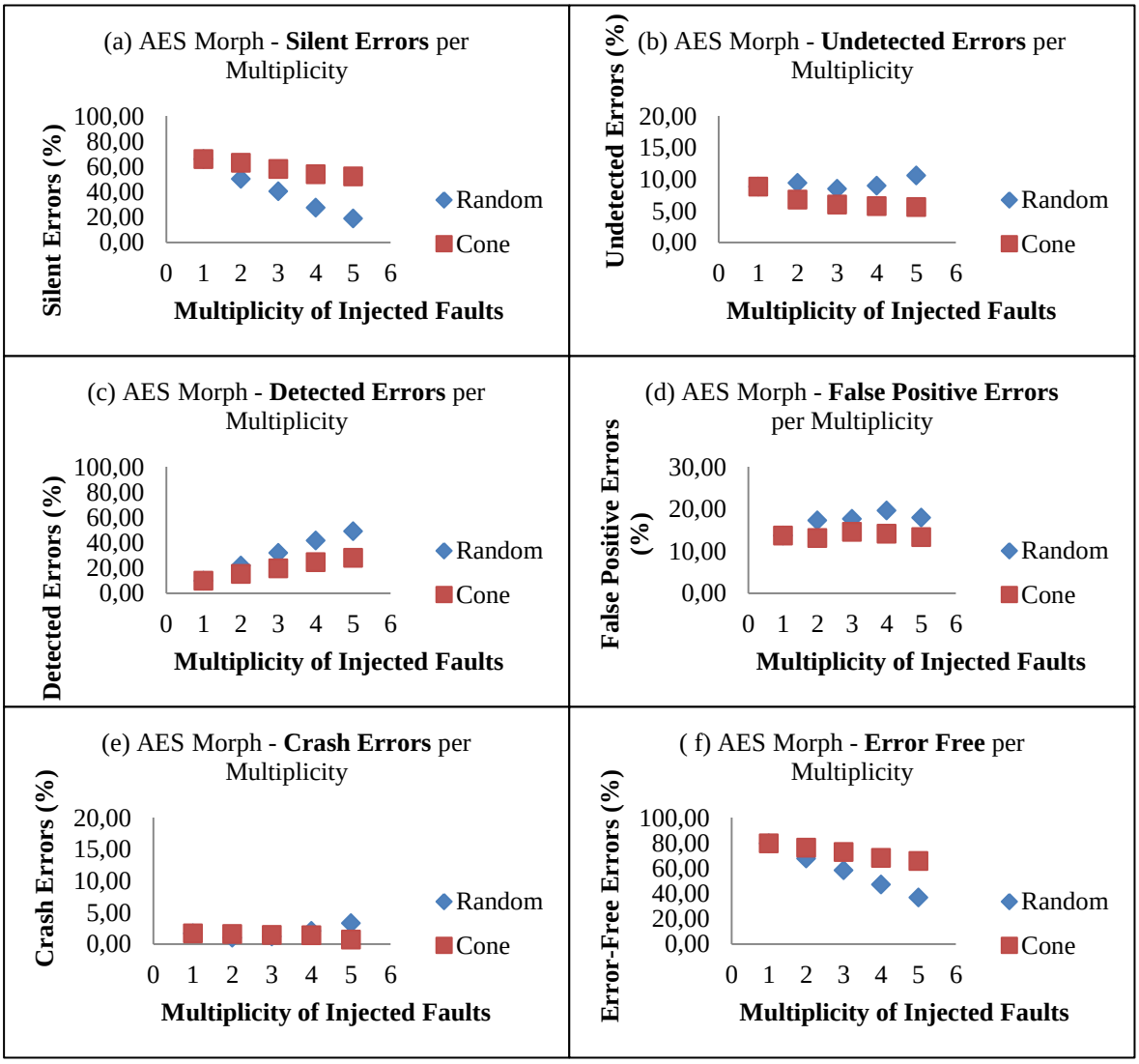


Figure 3.26: AES Morph, error percentages per multiplicity for the Random and Cone fault models

Once again, this shows that the random fault model is more capable to inject an error and propagate it to the outputs of the AES computation.

The comparison of the undetected error rates, included in Figure 3.26 (b), indicate that the evaluation according to the two fault models is similar, within the limits of the considered margin of error. The random fault model generates undetected errors in the order of 10% while in the case of the cone fault model the undetected errors are of the order of 5%.

Detected error rates, in Figure 3.26, show that faults generated by the random fault model are consistently higher than the faults generates by the cone model. For the highest considered multiplicity of five concurrent injected faults, the random model has a detection rate approximately double the rate of the cone model.

Crash error rates are low for both fault models and equivalent according to the considered margin of error.

3.5 Conclusions

The fault injection results presented in the current chapter show that the considered fault model has a big importance for the evaluation of secure designs early in the design flow. In this chapter we presented fault injection results on two circuits according to the random fault model and the cone fault model. These fault models are defined by different assumptions. On one hand, the random fault model assumes that it is equally probable for an attack to inject a fault to any FF of the design. On the other hand, the cone fault model assumes that only one entire logic cone can be affected at the same time (and all the cones the affected cone intersects with). Therefore, it assumes that only functional dependent FFs may be recipients of concurrent multiple faults. As we saw, the error rate results obtained by the two fault models are qualitatively and quantitatively different. Furthermore, the analysis of the results indicates that the cone fault model is more capable to keep track with the properties of each of the evaluated countermeasures. In the case of the AES Parity design, the cone fault model was able to clearly indicate the distinct ability of the countermeasure to detect odd and even faults. Additionally for both designs the cone fault model error-free rates are more stable and this shows that it can better constrain the fault injection to specific functional regions of the design.

It is important to emphasize that the highest degree of locality for RTL evaluations can be described by single bit fault injection as any fault either injected in combinational or sequential logic, will either become silent or it will affect at least one FF. The results show that both fault models provide results close to single bit fault injection when the multiplicity of the injected faults is equal to two. As multiplicities continue to rise we have a divergence from the single bit results, with the same tendency for both fault models. For all the error rates, the cone fault model is closer to the single bit results than the random fault model. Due to this fact we can firstly deduce that the multiplicity of injected faults is a way to localize the impact of the attack. Secondly, the cone fault model provides results with characteristics closer to single bit fault injection and therefore it seems more appropriate to perform early design stage evaluations of localized fault attacks compatible with the capabilities of laser fault injection equipment.

In the next chapter we will present results of the evaluation of the locality of the fault models themselves based on layout information of the same AES designs. Furthermore, we will include experimental fault injection campaign results performed on the AES parity design so as to compare them with the fault space of the cone fault model.

3.6 Contributions

- Comparison of the fault injection results of the cone fault model and the random fault model regarding injection locality
- Emulation based fault injection campaigns on two AES implementations with different countermeasures so as to compare the fault model capabilities to generate multiple faults compatible with localized laser attacks

4 Validation of the proposed fault model

4.1 Validation with respect to layout

The goal of the validation approach presented in this section is to validate the degree in which the locality that the cone fault model predicts at RTL is accurate when compared with the locality after the completion of the design's layout. RTL fault models are very useful in order to perform early evaluations against laser fault attacks and accelerate the duration of fault injection campaigns. A valuable source of information which is completely missing from RTL is the placement of the standard cells. On the other hand, the completed layout of the design is the abstraction level which has the most complete set of information before the actual fabricated IC especially as concerns locality characteristics. Therefore, during the design procedure and before fabrication the layout is the most reasonable abstraction level to use for validating a fault model defined at a higher level.

The work presented in this section has been published in the articles [P1] and [P6].

4.1.1 Validation approach

The main aspects of the proposed RTL fault model that need to be validated are arising from its assumptions, as presented in section 2.5. Therefore we need to validate if the assumption that only one logic cone can be under attack at the same time holds (assumption 1 of section 2.5.1) and in what degree. This assumption could not hold if for example a focused laser spot manages very often to inject faults concurrently in two (or more) non-intersecting logic cones.

The second assumption to be evaluated is if faults which have their origin inside one logic cone, will propagate only to the father FFs, as indicated by the elaborated RTL netlist (assumption 2 of section 2.5.2). This may not hold in the case where synthesis and optimizations occurring after the elaborated netlist level, alter drastically the intersections of logic cones (alter the functional dependencies).

Furthermore we also need to verify the assumption that FFs with functional dependencies are more likely to be placed nearby and therefore to verify in what extent the model is able to predict direct FF attacks (assumption 3 of section 2.5.3).

The validation procedure, detailed in [P6], starts with logic cone partitioning at the gate level. This allows the implementation of the same methodology, which was presented in section 2.5, this time at the gate level so as to compare the results before and after synthesis. Additionally, this way we can also obtain the gate level fault space according to the cone fault model. Thus we are able to obtain the potential attack recipients sets that may capture faults at gate level (gate level cone fault model). Furthermore, the gate level cone partitioning allows the connection between the RTL and the layout level as there exists a one-to-one correspondence between RTL and gate level logic cones.

In Figure 4.1, we illustrate a local attack constrained inside a square on the layout. For the purposes of the validation approach this square is defined as the attack origin at the layout level. The square shape was chosen mainly for technical reasons (due to the OPENACCESS interface). Furthermore it leads to more conservative (pessimistic) validation results as it is

used to describe the (smaller) inscribed circle, representing the actual shape of the laser spot. By extracting its contents (standard cells), we obtain the set of gate level cones which are affected by this attack. The gate level cone partitioning then provides the attack capture to the considered attack recipients. Once again the recipients of the attack are the FFs which may potentially capture an attack. Thus we are able to determine the set of FFs (attack recipients) that may contain erroneous values, after an attack inside this square. Since the attack capture considers any possible functional path to determine the attack recipients, the generated fault space does not depend on specific inputs of the circuit or its state during the time that the attack occurs. Furthermore, we do not consider any physical or structural effects that require information not available at RTL. Therefore we do not take into account effects on the clock and reset trees as well as on the power and ground distribution. Such structures are not defined at RTL and therefore it is a lot more meaningful if their effects are treated when the final layout is completed. Additionally, knowledge of such effects at RTL is not very valuable, concerning decisions that have to be made to improve high level countermeasures.

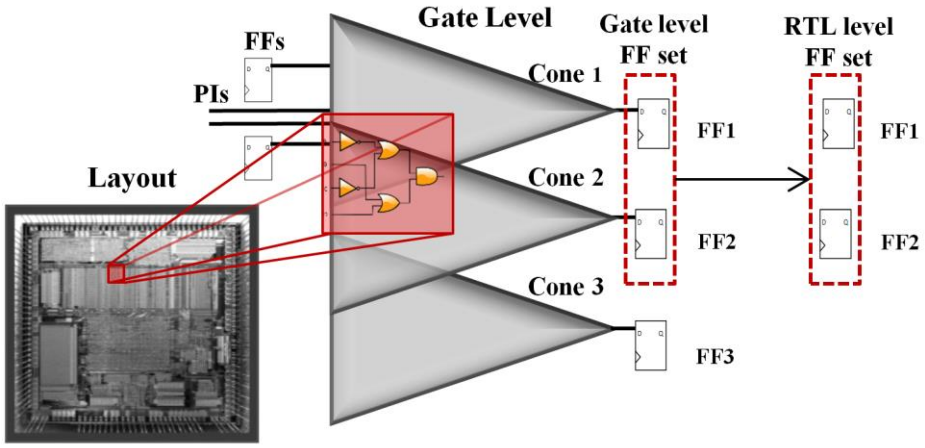


Figure 4.1: Extraction of the attack recipient sets (FFs) affected by a local attack.

The above procedure is repeated and a scan of the chip is performed, according to the size of the square (spot) and a step size. After scanning the chip we check if each attack is covered by the RTL and the gate level fault models. To determine if an attack is covered, we compare the possible attack recipients of the RTL and the gate level fault models with the ones that the layout validation tool indicates. Thus we use the attack recipient FF sets related to each square and we verify if this combination is fully contained (subset), in any of the RTL attack recipient sets. Then, we perform the same check for the gate level attack recipients. After this analysis, each box is categorized depending on if it was covered (or not) by the RTL and gate level fault sets. We end up determining the percentage of the layout area which was covered by the RTL and gate level analyses respectively.

We must stress the fact that this procedure evaluates at the same time all the assumptions of the RTL fault model. If a localized area on the layout (spot) generates fault recipients which are a subset of at least one RTL attack recipient set, this means that: (a) the layout attack has an attack origin affecting functionally dependent logic cones; (b) the fault capture

prediction at the elaborated RTL netlist also holds at the gate level, since it did not lead to faults being captured at FFs which are not functionally dependent in the elaborated RTL netlist; and (c), since the layout validation considers at the same time combinational and direct attacks, it is also able to evaluate direct attacks. In Figure 4.2, we can see an illustrative abstract example of the RTL coverage percentage on the layout of an imaginary circuit. For illustration purposes, we use spot size equal to step and arbitrarily a coverage of approximately 90%. This would mean that 90% of the IC area can be evaluated right after RTL design (white squares). On the contrary the remaining area (black squares), corresponds to fault scenarios that are unknown through the RTL fault model because they need to be evaluated during the post layout design stage.

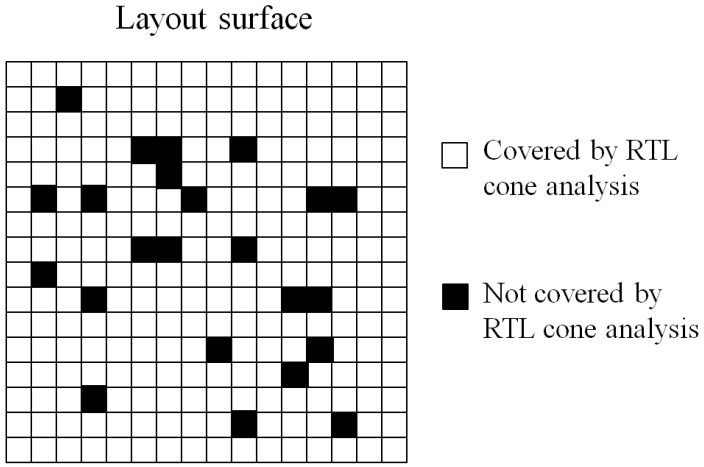
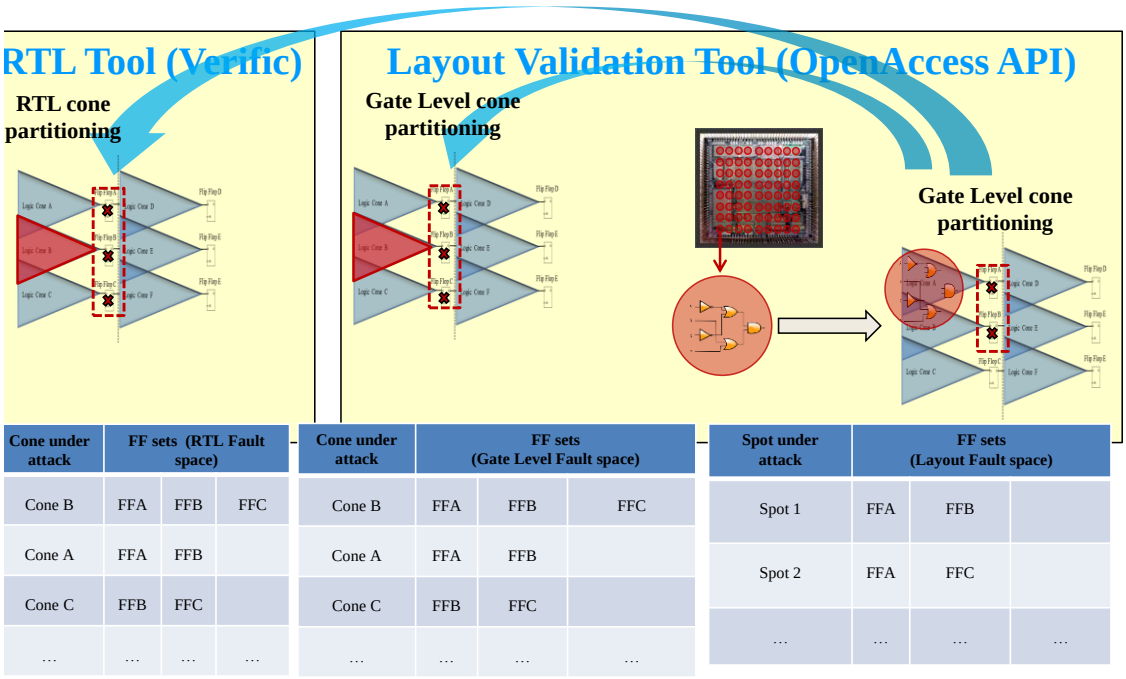


Figure 4.2: Illustration of coverage percentages; in this example, for spot = step we have coverage percentage ~= 90%)

An example of the validation procedure is also illustrated in Figure 4.3. On the left we can see a part of an abstract circuit and its partitioning in logic cones. In the depicted example logic cone B is the considered attack origin which generates an attack recipient set containing FF_A, FF_B and FF_C. Similarly in the middle of the figure we can see that the same attack origin at the gate level is generating the same attack recipient set. The right hand side of the figure illustrates the finalized layout and its partitioning in attack origins at the layout level (red spots). As depicted, the validation tool reads the contents of the layout spots, one by one and for each of them it extracts the logic gates and/or FFs it contains (e.g. spot 1). Then the tool extracts all the gate level logic cones where these gate level elements belong to. In our example they are located inside logic cones A and B. Therefore an attack on spot 1 dictates as an attack recipient set, cones 1 and 2. As explained previously, the next step of the validation is to verify if the attack recipient set of spot 1 is a subset of at least one gate level attack recipient set. For this example indeed the set containing FFs A and B is a subset of at least two recipient sets at the gate level, namely the sets generated by cones A and B. Therefore spot 1 is covered by the gate level cone fault model. Furthermore, spot 1 similarly is a subset of the recipient sets of cones A and B at the RTL level. Thus spot 1 is covered by the cone fault model either at RTL or gate level. In this example a spot of the layout is covered and additionally we can see that for this part of the circuit there is no difference between the RTL and

gate level cone partitioning. Naturally it is also possible for a spot to be covered by one of the cone fault models, either at RTL or gate level or it may not be covered by either of them.



Cone under attack	FF sets (RTL Fault space)		
Cone B	FFA	FFB	FFC
Cone A	FFA	FFB	
Cone C	FFB	FFC	
…	…	…	…

Cone under attack	FF sets (Gate Level Fault space)		
Cone B	FFA	FFB	FFC
Cone A	FFA	FFB	
Cone C	FFB	FFC	
…	…	…	…

Spot under attack	FF sets (Layout Fault space)		
Spot 1	FFA	FFB	
Spot 2	FFA	FFC	
…	…	…	…

Figure 4.3: Validation procedure

If a spot is not covered by either version of the fault model, gate level or RTL, then this means that the physical location of the spot (in combination with the design's placement) targets cones which are not intersecting, either at RTL or at gate level. In the case when the spot is covered by the gate level fault model but it is not covered by the RTL (due to alterations that synthesis enforces), this means that this spot cannot be evaluated correctly at RTL for the same reason. Nonetheless it can be evaluated when the design reaches the gate level. The reverse case when the spot is covered by RTL but it is not covered by gate level means that synthesis alters the RTL elaborated netlist in a way which separates logic cones which are intersecting at RTL.

If a spot is covered by the RTL fault model, then the statistical selection of faults from the RTL attack recipient sets, as presented in the previous chapter, will be capable to evaluate its behavior under an attack. Even though the statistical analysis cannot cover every possible local fault attack, the selected samples will have high locality characteristics. Such an analysis will provide the opportunity for designers to evaluate the resilience of cryptographic algorithms and relevant countermeasures with the simulation or emulation of fault injection campaigns, representative of local laser fault injections.

4.1.2 Validation flow implementation and results

For the implementation of the validation flow, the OpenAccess (OA) C++ API has been used [47]. Initially an RTL design is synthesized using Synopsys Design Compiler, by making use of the Nangate 45nm open cell technology library. The synthesized Verilog netlist is then passed to Cadence SOC Encounter, for placement and routing. For the purposes of the

evaluation, only timing constraints where used and no floorplan or placement constraints where provided to the layout tool. The combinational standard cells that are placed under the highlighted area of Figure 4.1 can be extracted by utilizing the OA API. In detail, a C++ function has been implemented: it instantiates a bounding box, by taking as arguments the coordinates of its lower left and upper right corners, and performs a query on the layout for the standard cells enclosed in this box.

The circuits chosen to be validated are the B18 design of the ITC99 benchmarks [48], as well as the two AES implementations which were introduced in the previous chapter. The first one (AES Morph) contains countermeasures based on hardware redundancy and data reallocation schemes [46], while the second is the parity protected AES, described in [45]. Figure 4.4 depicts the obtained characterization cartography of the layout for the hardware redundancy protected AES. Each spot corresponds to one potential attack origin at the layout and in this case for visibility purposes the step of the scan is equal to the spot size of 5 μm. It depicts all the spots which are not covered by the RTL fault model in black color, the spots which do not affect any combinational or sequential logic in gray color, and the spots that are covered by the RTL analysis in white color. From this figure we can see that the majority of the spots are covered by the RTL analysis (white spots). The black spots which the RTL analysis fails to cover are affecting very large multiplicities of attack recipients (FFs) of an order higher than five hundred. This is usually related to control logic driving several FFs at the same time. Spots in gray correspond to the layout area related to power and ground. These spots do not affect any combinational or sequential logic and their behavior under an attack are not meaningful for an RTL validation.

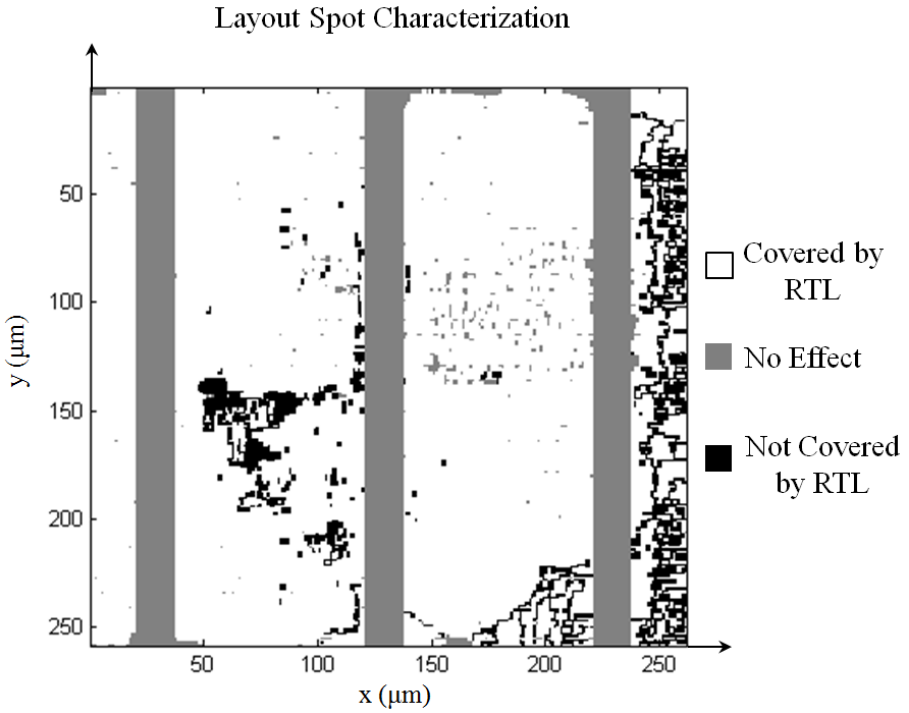


Figure 4.4: Layout characterization for the hardware redundancy protected AES design, spots of 1 μm covered at RTL (white), spots which have no effect (gray), spots not covered by RTL (black).

Figure 4.5 shows the same cartography results for the parity protected AES design.

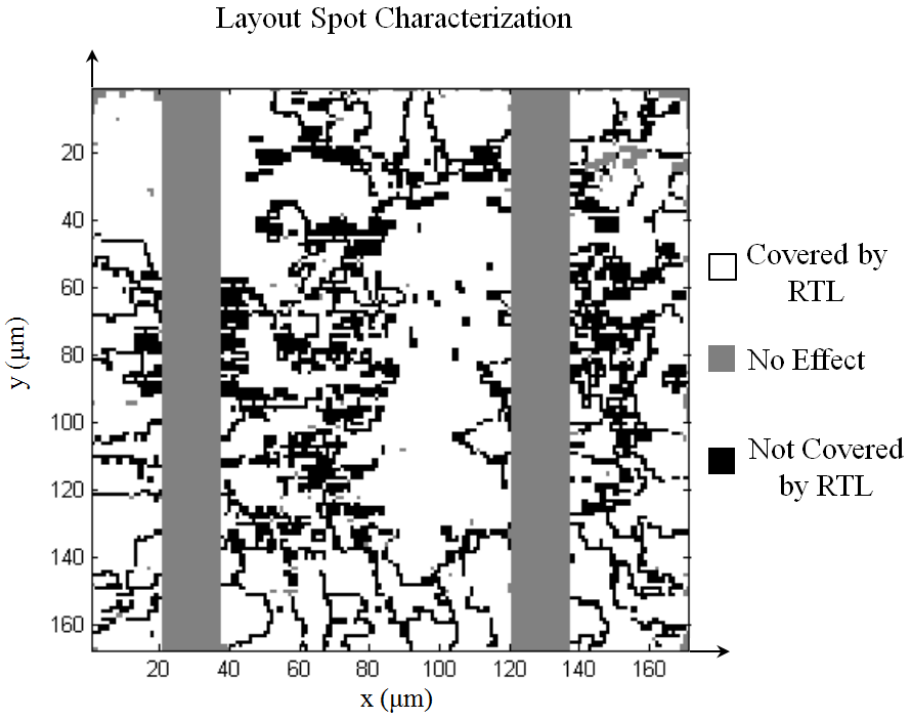


Figure 4.5: Layout characterization for the parity protected AES design, spots of 1 µm covered at RTL (white), spots which have no effect (gray), spots not covered by RTL (black).

It is important to mention that initially the RTL fault model can be used so as to validate the design. Then the validation approach can verify the parts of the layout that have already been validated by fault injection at RTL. The attack origins on the layout which are not covered are not known to the RTL design stage, but the validation tool can effectively assist to complete the evaluation. It can provide all the potential attack recipients which remain to be validated with fault injection campaigns at RTL simulation or emulation speed. This procedure can avoid unwanted and costly feedback loops between the layout and higher circuit design abstraction levels.

Table 4.1 summarizes the validation results for the chosen implementations. For each design we have chosen three different square sizes, 1 µm, 5 µm and 20 µm, (second column) and a step always equal to half the spot size (column three). The fourth column includes the percentages of the layout spot attack origins (out of all the considered spots) that are covered by the RTL analysis. Column five contains the percentage of spots that generate FF sets entirely included in one or more FF fault sets of the gate level fault space. The RTL and gate "direct attack" columns provide the percentages of direct attacks that are found inside one or more fault sets of the RTL and gate fault spaces, respectively. We have a "direct attack" when, in each layout attack origin, the only recipients of an attack are the FFs contained in it (SEU or MCU). Column eight includes the maximum number of FFs that were encountered inside one layout spot, in each analysis. This maximum attack recipient set size for each analysis is also defined as the maximum multiplicity for the purposes of the layout validation flow. The term maximum multiplicity is used to show that this is the maximum number of FFs that the corre-

sponding attack may induce to the design. Thus it is the number of concurrent faults that would need to be injected during a fault injection campaign, if there was no limitation on the amount of concurrent faults. The next three columns contain the maximum set sizes (or maximum multiplicities) of the attack recipient FF sets in the RTL, gate level and layout fault spaces respectively.

Table 4.1 Validation Results wth Respect to Layout – Covered Spots – Covered Direct FF Attacks – Fault Space Multiplicities

Design	Spot (um)	Step (um)	RTL Covered spots (%)	Gate Level Covered spots (%)	RTL Covered Direct Attacks (%)	Gate Level Covered Direct Attacks (%)	Direct Attack Max set size (Multiplicity)	RTL Max set size (Multiplicity)	Gate Level Max set size (Multiplicity)	Layout Max set size (Multiplicity)
AES HR (2369 FFs)	1	0.5	91.1	95.6	90.1	93.5	4	1152	2113	2065
	5	2.5	79.1	91.7	79.1	89.0	13			2066
	20	10	55.5	82.1	67.4	79.8	81			2087
AES Parity (936 FFs)	1	0.5	76.4	88.9	85.9	88.2	4	202	685	652
	5	2.5	35.7	68.8	60.4	68.6	11			655
	20	10	7.3	34.3	27.8	41.2	54			711
ITC99-B18 (3320 FFs)	1	0.5	86.9	90.3	96.5	99.1	4	561	447	478
	5	2.5	78.1	86.6	89.2	96.2	10			496
	20	10	61.5	71.3	66.9	82.6	55			633

In the third column, we see that for the case of a square (spot) of 1 μm, for all the designs, the RTL analysis covers more than 75% and 91% of the area of the layout, as we described it for the simplified example of Figure 4.2, for the parity protected and the hardware redundancy protected designs respectively. These are the results of the cartography of Figure 4.4 and Figure 4.5. When the spot size is increased to 5 μm the RTL coverage for the Morph design falls to 79.1%. For the AES parity a 5 μm spot reduces the coverage to more than half, to 35.7%. The RTL analysis manages to cover 86.9% and 78.1% of the layout area of the B18 design for 1 μm and 5 μm spots respectively. The coverage is quite high for spots of 1 μm, given the fact that it is achieved at RTL and so it allows reducing the re-design feedback loops between RTL and layout descriptions. We can also see that the coverage percentages for the gate level FF fault sets are higher than the RTL. This is due to the fact that the gate level fault space takes into account the synthesis of the RTL code (mapping & optimizations of the final post-layout netlist). For the AES Morph design the gate level fault model covers 95.6% and 91.7% for spots of 1 μm and 5 μm respectively. For the AES parity we obtained a coverage of 88.9% and 68.8%, while for the B18 90.3% and 86.6% respectively. This shows that, as expected, the gate level analysis exploits richer information about the final layout of the circuit and this translates to a higher coverage than RTL. Even though gate level has higher coverage, we can see that for spots of 1 and 5 μm the RTL and gate level results are very close. This shows that our assumptions hold in a large extent and that the optimizations are not affecting the validity of our assumptions a lot. As the spot size increases, we notice a decrease

in the coverages for both RTL and gate-level analyses. This occurs due to the fact that for large spots, clearly multiple non-intersecting cones are impacted concurrently. This translates in a failure of the assumption of the RTL methodology which states that each time only one entire cone is under attack.

Concerning the RTL coverage of direct attacks on multiple FFs (MCUs), we can see that for both designs the percentages are above 88% for the spot size of 1 μm. For AES Morph a spot of 1 μm covers 90.1% of direct attacks while the AES parity 85.9%. This verifies that the FFs that are functionally related will end up placed in the same localized area. Additionally as the spot sizes increase we maintain high coverages. Inside the IC design flow, the validation flow can take place when the design eventually reaches the post-layout phase. Then by performing the above analysis the FFs that are potential targets for MCU attacks can be extracted only to verify that the used countermeasures can cover all attack scenarios.

Random placement is sometimes used in secure products as a countermeasure to obscure the design for attackers. Such a countermeasure could reduce the efficiency of our approach. However it will have a very negative impact on the timing of the design and thus it can only be used for particular designs. Also most designs which can be attacked in the consumer market cannot afford such countermeasures.

Table 4.2 includes some additional results after analyzing the results of the previous table. The sixth column contains the percentages which are covered by the Gate level but at the same time they are not covered (missed) by the RTL. These percentages correspond to the scenarios where synthesis optimizations and mapping alter the cone intersections of the RTL netlist and at the same time are due to the spot affecting concurrently logic cones which do not intersect at RTL. Column seven on the other hand contains the percentages which are missed by the gate level. These are clearly involving situations where the laser spot is affecting at the same time non intersecting logic cones. For most cases, the failure to cover layout spots at RTL is twice the percentage of the failure according to the Gate level analysis. This shows that none of the assumptions fails consistently and non-coverage is partly due to netlist alterations during synthesis and partly due to placement (when placement layouts in the same neighborhood non-intersecting logic cones.

Table 4.2: Validation Results with Respect to Layout – Differences between RTL and Gate coverages

Design	Spot (um)	Step (um)	RTL Covered spots (%)	Gate Level Covered spots (%)	Covered by Gate – Missed by RTL	Missed by Gate	Missed by RTL
AES HR (2369 FFs)	1	0.5	91.1	95.6	4.5	4.4	8.9
	5	2.5	79.1	91.7	12.6	8.3	20.9
	20	10	55.5	82.1	26.6	17.9	44.5
AES Parity (936 FFs)	1	0.5	76.4	88.9	12.5	11.1	23.6
	5	2.5	35.7	68.8	33.1	31.2	64.3
	20	10	7.3	34.3	27	65.7	92.7
ITC99-B18 (3320 FFs)	1	0.5	86.9	90.3	3.4	9.7	13.1
	5	2.5	78.1	86.6	8.5	13.4	21.9
	20	10	61.5	71.3	9.8	28.7	38.5

4.1.3 Fault space analysis

In Figure 4.6 we graphically illustrate the fault spaces for the random, the RTL cone based and the layout fault models. The order of magnitude of the fault space for the cone method (at RTL or gate level) would be of the order of magnitude: $(2^{multiplicity})$. In the case of the random method it would be: $(2^{\#FFs} - 1)$. The maximum multiplicities of Table 4.1 explain the depicted difference between the random approach and the RTL cone based approach.

The results in Table 4.1 show that the random method creates a fault space many orders of magnitude larger than the RTL cone method does. Furthermore, the layout multiplicities show that there is no localized area on the final layout, which can potentially affect all the FFs of a design. This shows that the random method will also contain many multiple fault combinations which are not relevant to localized attacks. For all the designs the layout multiplicities are closer to the cone methodology applied at the gate level. For B18, a spot of 1~5 μm can affect a maximum of 496 out of the 3320 FFs of the design, since we consider all possible combinational propagations. On the other hand we can see that the fault spaces of the RTL and the Layout fault models overlap. To characterize the quality of the RTL fault space it is important to quantify this overlap.

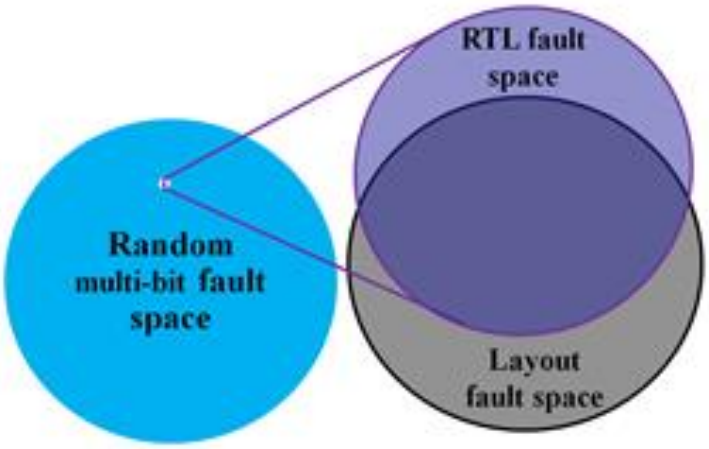


Figure 4.6: Graphical illustration of the relevant fault space sizes

Let us consider one random combination of FFs of the design (of any multiplicity). This combination will belong to the RTL fault space if it is a subset of at least one FF set of the RTL cone fault space. On the other hand the same FF combination will belong to the layout fault space, if it is a subset of at least one attack recipient set of the layout fault space. Therefore this combination belongs to the intersection of the two fault spaces if it is at the same time a subset of at least one attack recipient set for both fault spaces. The size of these fault spaces, even for medium sized circuits, does not allow quantifying analytically their intersection. So as to obtain an approximation of this intersection we apply the statistical methodology as described in [22]. Thus first we choose one specific multiplicity (number of FFs) and randomly select samples from one RTL FF set, until we achieve the specified margin of error. Then we check if each sample is at the same time a subset of at least one layout FF set. This way we calculate one percentage for each RTL set and we average them for each multiplicity.

In Table 4.3 we include this analysis for the AES parity design. The first column contains the sample multiplicities. Columns two and three contain the statistical analysis results of the RTL cone based fault space. The remaining two columns concern the same analysis for the random multi-bit fault model. For this analysis we have used a margin of error of 10% for both fault models and all multiplicities. For the RTL cone based fault space in Table III we present the average overlap for each multiplicity. On the other hand the random multi-bit fault space is composed by only one set containing all the FFs of the design.

We should emphasize the fact that the set of potential attack recipients for the random multi-bit fault model contains every possible fault combination. Therefore it also contains all potential layout fault recipients as illustrated in Figure 4.6. Our goal is to show that the fault space of the RTL cone method corresponds to those fault combinations which have better locality characteristics. To achieve that we also sampled from the FFs in cone attack sets, in the same way as the random multi-bit fault model does to select fault candidates in the whole set of FFs. For the RTL cone based fault space we can see that for a multiplicity of two, 95.7% of the selected samples were subsets of at least one FF set of the layout. As the multiplicity is increased we can see that the overlap percentages decrease to 77.4% for the rather

high multiplicity of 20 simultaneously erroneous FFs. The same analysis shows a completely different tendency for the random multi-bit fault space. For a multiplicity of two only 57.4% of the selected samples were subsets of at least one spot. Furthermore for a multiplicity of three, the overlap is reduced by half to 28.7%, and for a multiplicity of five it is only 17.1% of the samples belong to the layout fault space. Finally for a multiplicity of twenty there was no overlap at all.

Even though the random method samples from the fault space of any possible fault, it is not feasible for this approach to provide samples representative of local attacks especially when error multiplicity is increasing. Furthermore there is no point for an exhaustive fault injection (or selecting a very large number of samples) since such an analysis at RTL will also contain many non-relevant faults for a laser attack evaluation. Early stage fault injections containing non relevant faults for laser attacks could then lead to over-constrained countermeasures or biased evaluations. On the contrary, the proposed RTL cone based approach succeeds in selecting fault samples which possess high locality characteristics and thus are more representative of laser attacks.

Table 4.3: Statistical Analysis of the RTL and the Layout Fault Space Overlap

AES Parity				
RTL cone fault model			Random multi-bit fault model	
Fault Multiplicity	Average Overlap of RTL and Layout (%)	Max Error (%)	Overlap of RTL and Layout (%)	Max Error (%)
2	95.7	10	57.4	10
3	92.3	10	28.7	10
4	89.3	10	25.8	10
5	87.2	10	17.1	10
10	80.7	10	1.9	10
20	77.4	10	0	10

4.1.4 Conclusions

In the current section we have presented results validating the locality characteristics of cone fault model by means of finalized layouts of two AES implementations with different countermeasures. The results show that synthesis, placement and routing (due to routing placement optimizations) do not affect in a large extent the initial assumptions of the cone fault model, as they were presented in Chapter 2.

When we perform fault injection at RTL we are able to inject faults which will be constrained inside the covered layout spots. If we performed an exhaustive evaluation according to the cone fault model we would be able to evaluate any local attack on these spots. Furthermore we need to emphasize the fact that the evaluation will cover more spots than the ones which were characterized as covered. This is due to the way that we decide if a spot is covered or not. Even if it is not a subset of any attack recipient set of the cone method due to only one FF, then the spot is characterized as non-covered. The remaining minority of (non-covered) spots are not visible to the cone fault model at the RTL design stage. These spots can

be evaluated with the assistance of the layout validation tools presented in the current section when the design reaches the post layout design stage.

4.1.5 Contributions

- Layout-based validation of the proposed fault model
  - Definition of the validation flow
  - Implementation of an EDA tool to automate the validation flow

4.2 Validation with respect to experimental laser attacks

In order to validate the RTL cone fault model several laser fault injection campaigns have been performed at the ENSMSE (École Nationale Supérieure des Mines de Saint-Étienne). These campaigns were made on the Parity protected AES design which was also analyzed in the previous sections. In the next sections we will first introduce the experimental methodology which produces results which can be used to validate our RTL fault model in a straightforward way. Then we describe the experimental platform as well as the experimental results.

4.2.1 Experimental methodology and goals

The main goal of the experimental approach is to validate the extent in which the error combinations which are injected to an IC by means of an experimental laser fault injection campaign are covered by the results of the RTL analysis of the design's RTL code.

- A fault combination (captured at the FFs of the design), injected by means of a laser is defined as **covered by the RTL methodology** if it is contained inside (a subset of) at least one RTL attack recipient set.

Therefore in the current section the experimental results will be utilized so as to validate the capabilities of the RTL fault model. For this purpose two different laser fault injection campaigns were performed at CMP (Centre Microelectronique de Provence).

The AES Parity designed by TIMA laboratory, was implemented by ST Microelectronics on silicon by using the ST Microelectronics 28nm bulk technology. For this implementation no specific layout constraints were used.

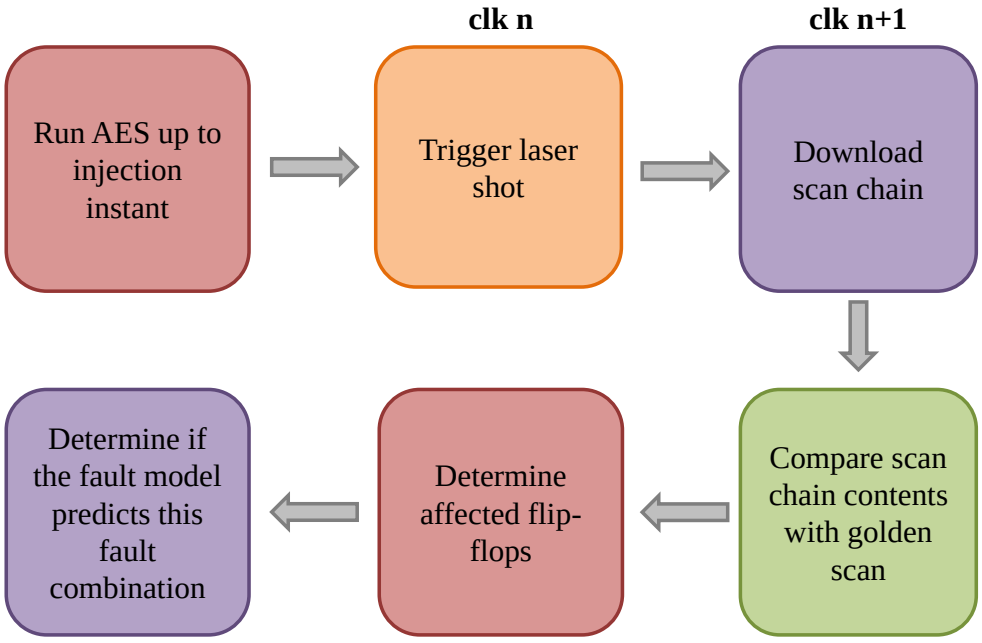


Figure 4.7: Experimental laser fault injection flow

The validation is based on the comparison of the recipients of the attack according to the experimental laser fault injection and the RTL fault model. The attack recipients of the RTL fault model are already known from the RTL design stage, as explained in the previous chapters. The experimental fault attack recipients on the other hand were obtained by utilizing the scan-chain of the Parity protected AES implementation.

During the experiments a clock cycle is selected (which we will also call the 'injection instant') to perform a laser attack, with a pulse duration of one clock cycle. The laser was instructed so as to target the selected clock cycle with an uncertainty (jitter) of plus or minus one clock cycle due to the capabilities of the used equipment. Then multiple attacks were performed with exactly the same injection parameters. Each time we shot a laser pulse, targeting the same clock cycle X and we performed the readout of the contents of the scan chain each time on the following rising edge of the clock, increased by one. This way all the rising edges from the "injection instant" up to the end of the encryption were read, as in Figure 4.8.All clock cycles: X-1, X, X+1, X+2, …, X+n (where X+n is the last clock cycle of the AES) were read after shooting the circuit with a laser pulse on clock cycle X (the injection instant remained the same). The above experiment was repeated ten times so as to obtain results on the repetitiveness of the fault injections.

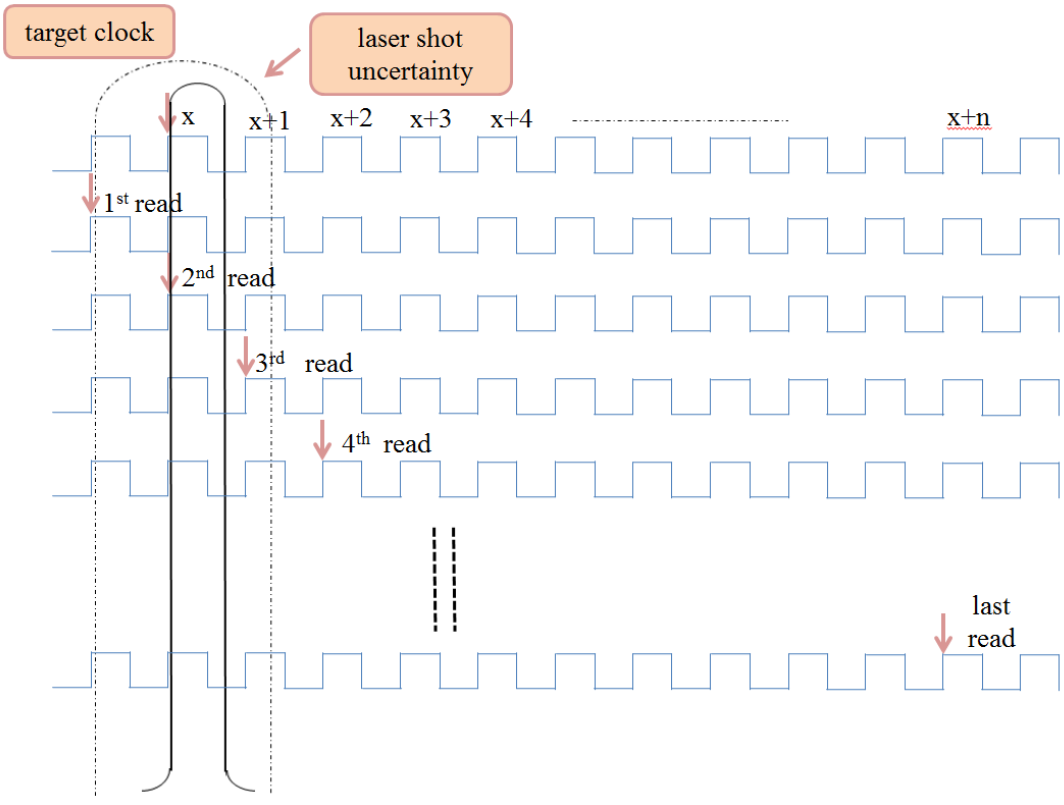


Figure 4.8: Experimental laser fault injection approach for one set of laser parameters

This strategy enabled us to cope with the uncertainty of the injection instant and additionally it allowed us to monitor the propagation of the faults during the operation of the circuit.

After the completion of the fault injection campaign we post processed the results by comparing the obtained results with the golden scan-chain contents. The golden results were recorded under fault-free operation for each clock cycle of the operation of the circuit. This analysis enabled us to know which FFs captured errors due to the laser injections for every clock cycle after an attack. Furthermore it allowed us to know the type of the injected faults among: bit-flip, bit-set and bit-reset. Out of all clock cycles of each experiment, the first one during which the scan-chain contained errors was considered as the clock cycle immediately after the laser attack. Therefore the erroneous scan-chain bits corresponded to the FFs which captured faults within one clock cycle of the laser attack injection instant regardless of the uncertainty on the injection instant (of one clock cycle). According to the definitions of the previous chapters, these erroneous FFs are attack recipients of the experimental laser fault injection campaign. Additionally we used the ten identical experiments with the exact same parameters so as to verify the repetitiveness of our approach. Therefore a set of identical experiments was considered as valid if more than 50% of them lead to faults being captured at the same clock cycle, without leading necessarily to the same fault combinations.

The obtained attack recipients were also confirmed by monitoring the multiplicity of injected faults on the next clock cycles. From this analysis the general conclusion is that after the first rising edge of the clock following each attack we noticed that with every consecutive clock the injected faults lead either to the same error multiplicity inside the scan-chain (the errors did not propagate further) or to increasing multiplicities (the errors propagated further inside the AES). These experimental results did not take into account the propagation of the errors to the circuit's outputs as well as if they lead to detected or non-detected errors, since the main goal was the validation of the attack recipients of the RTL fault model.

The laser fault injection parameters were then changed and the same procedure took place all over again. After obtaining the list of all the attack recipients of the campaign, each one of them was intersected with each of the attack recipient sets produced by the RTL fault model. If an experimental attack recipient was a subset of at least one attack recipient of the RTL fault model, then the experimental result was marked as covered by the RTL fault model. Finally the percentage of the covered experimental attack recipients is used as a metric for the validation of the RTL fault model. This way if the majority of the experimental errors are subsets of the RTL attack recipient sets, then this means that the categorization of the design's FFs according to the RTL fault model contains locality characteristics compatible to actual experimental laser attacks.

A high coverage will also indicate that RTL fault injection constrained inside the RTL attack recipient sets will be capable to reproduce the majority of the actual experimental laser faults which may be injected to the circuit after its fabrication. It is true that even with coverages of 100%, in order to simulate all the laser errors at RTL, the RTL fault injection campaign would have to be exhaustive. As explained in the previous chapters even though this is not feasible for complex circuits, a high coverage will still indicate that the RTL fault model manages to remove from the RLT fault space faults which are not relevant to localized laser attack on the actual circuit.

4.2.2 Experimental platform

The experimental setup included two Printed Circuit Boards (PCBs) so as to align the chip with the laser source and to interface it with an FPGA development board. In Figure 4.9 we can see the two PCBs, on top the motherboard, responsible for powering and interfacing the socket with the FPGA board. Below the motherboards we can see the socket board which was used to hold the ASIC. The sockets were needed to facilitate testing multiple different chips. Finally, at the bottom we can see the ASIC and its opening in order to be able to focus the laser beam on its back side.

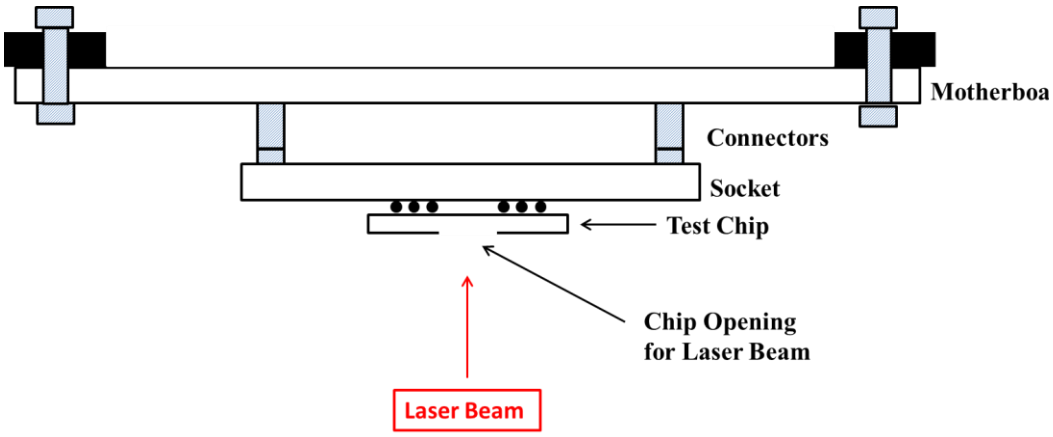


Figure 4.9: Schematic representation of the stack of the Motherboard, socketboard and chip for the experimental laser campaign

4.2.3 Experimental parameters and results

4.2.3.1 First Laser Fault Injection Campaign

For the first experimental laser attack campaign, the parameters involved a full area scan of the AES parity design with a laser spot size of 5 μm and a step of 4 μm. This scan allowed us to have a full scan of the circuit including some overlap between the spots. The parameters used were the power, and the injection time:

- Power: 0.6, 0.7, 0.8 W
- Injection Time: 18 different clock cycles were chosen to inject faults in all the rounds of the AES 128 bit encryption
- For all experiments the duration of the laser pulse was set to 10ns, equal to the period of the AES clock.

For each of the power parameters we performed fault injections on all the 18 chosen clock cycles and for each of them we performed a full scan of the circuit, as described above. Initially we will limit the acceptable fault multiplicities to 10 concurrent faults. In total 398 faults with a multiplicity less than or equal to 10, with distinct injection parameters, were injected during this campaign. Out of these injections 194 involved single bit errors and 204 multiple bit errors. One initial but important conclusion is that it is possible to inject multi-bit errors and that they account for approximately half the injected faults.

After post-processing (following the procedure described earlier) the results, we have obtained the RTL coverage considering all the obtained fault multiplicities which are greater that one and up to ten. This constraint was set since single faults are a priori covered by any categorization of attack recipient sets. Furthermore, our methodology targets specifically multiple faults. The coverage for this experimental campaign was equal to **9.3%**. This means that 9.3% of all experimental faults were a subset of at least one RTL attack recipient set. After careful examination of the results we discovered a systematic reason for this low coverage. In every experimental fault with a multiplicity greater than 3, there was always only one flip flop which was not included inside the same RTL attack recipient set. Actually this FF was always belonging to a different byte and also to a different submodule. For example when an attack was influencing multiple FFs of the data-path, there was one FF of the key unit involved and vice versa.

This systematic failure of the RTL fault model to cover many multi-bit faults (mismatch), lead us to provide to the RTL fault model the freedom of failing to cover at most one erroneous flip flop from each experimental fault. This means that if an experimental laser (multiple) fault is not a subset of any RTL attack recipient set, due to only one flip flop then we allow for this attack to be considered as covered. Then after recalculating the RTL coverage, we obtained a result equal to 91.18% that verifies the repeatability of the previously explained mismatch due to only one FF in most experimental faults with a multiplicity larger than one.

Additionally, the above mismatch was only present for the highest parameter of the power of the laser source, which also provided the majority of the injected faults. For smaller power values, the previously explained mismatches were not present. Therefore these observations can lead us to the conclusion that a high power parameter may affect concurrently parts of the circuit that are not located in close proximity. One way for the above to be verified is by repeating the laser fault injection campaign on the same location that produced faults but this time by carefully adjusting the power parameters.

Earlier in the current section we mentioned that during the layout of the AES Parity design, no placement constraints were followed. This means that cells (either combinational or sequential) of the data unit and key unit of the AES were allowed to be placed nearby. On the other hand the RTL analysis considered these two sub-circuits of the design as completely decoupled. Therefore there existed laser spots during the experimental fault injection campaigns which were able to target non-intersecting logic cones.

Table 4.4 and Table 4.5 include the total number of faults obtained per multiplicity as well as their percentage which was covered by the RTL fault model, at first without allowing any mismatch and then by setting the allowed mismatch to one respectively. In the case were we do not allow any mismatch we can see that the RTL methodology can cover multiplicities of up to three concurrent errors. Higher multiplicities have dramatically smaller coverages. On the other hand when we allow a single mismatch we can see that we obtain coverages of 100% for multiplicities up to 4 concurrent errors. Higher multiplicities show very high coverages above 83.3% for multiplicities up to nine concurrent errors. The lowest coverage in this case is 69.2 % for the case of ten errors.

Table 4.4: RTL fault coverage per multiplicity – no allowed mismatch

Fault multiplicity	Faults per multiplicity	Covered faults per multiplicity	Coverage per multiplicity
1	194	194	100.0
2	5	5	100.0
3	1	1	100.0
4	22	2	9.1
5	36	9	25.0
6	50	0	0.0
7	36	2	5.6
8	23	0	0.0
9	18	0	0.0
10	13	0	0.0

Table 4.5: RTL fault coverage per multiplicity – mismatch of one allowed

Fault multiplicity	Faults per multiplicity	Covered faults per multiplicity	Coverage per multiplicity
1	194	194	100.0
2	5	5	100.0
3	1	1	100.0
4	22	22	100.0
5	36	34	94.4
6	50	47	94.0
7	36	30	83.3
8	23	20	87.0
9	18	18	100.0
10	13	9	69.2

Now if we consider all the injected faults of this campaign, which are 424 in total if we allow multiplicities even above 10 concurrent faults, another interesting result is the multiplicity of the affected bytes. In Table 4.6 we can see that 49.3% of the injected faults lead to only one affected byte. The remaining percentage managed to affect more than one byte of the AES, with a maximum of 11 bytes. This result shows that we cannot safely assume that a protection at the byte level will be able to protect against the majority of the attacks as there exist errors which affect at the same time more than one bytes.

The errors obtained in the scan chains after a single laser shot have also been analyzed to determine the error model to be preferred when doing early emulation-based fault injections. Considering all flip-flops in the AES crypto-processor, a total of 1883 erroneous bits was recorded during the campaign (by considering for example one 5-bit error as 5 separate bits), in 292 flip-flops out of 743. 190 errors can be classified as bit-set only (i.e. the considered cells were never changed from 1 to 0 by another shot). 170 errors can be classified as bit-reset only (i.e. the considered cells were never changed from 0 to 1 by another shot). 1523 errors can be classified as bit-flips (i.e. the considered cells were changed in either direction depending on the shot).

Table 4.6: RTL fault coverage per multiplicity – mismatch of one allowed

Multiplicity of affected bytes	Number of faults	% of each multiplicity of affected bytes
1	209	49.3
2	4	0.9
3	6	1.4
4	64	15.1
5	91	21.5
6	27	6.4
7	12	2.8
8	1	0.2
9	2	0.5
10	5	1.2
11	3	0.7

This confirms that an attacker may find laser positions so that only bit-set or only bit-reset errors occur in a given flip-flop; but at the same time, during automated scanning without a precise positioning with respect to each flip-flop cell, the bit-flip fault model is the most representative.

4.2.3.2 Second Laser Fault Injection Campaign

This fault injection campaign involved a full scan of the circuit with a spot size of 5 μm and a step size of 4 μm, the same as the first campaign. This time the injection took place only at the clock cycle at the beginning of the penultimate round of the AES. Furthermore multiple steps were used for the power of the laser source:

Power = [0.7, 0.725, 0.75, 0.775, 0.8, 0.85, 0.9, 0.95, 1] W.

Additionally for each set of the above parameters we performed five identical attacks.

This campaign was performed in order to further investigate the results of the previous campaign were at the highest power setting used we had always multiple errors in the data unit and only one bit error in the key unit at the same time (or vice versa). Therefore in this campaign we performed a gradual increase of the power of the laser source and scanned the entire circuit each time with a five micrometer laser spot and four micrometers step.

The results in this campaign were different as the RTL coverage this time was **52.6%** (without allowing any mismatch). The reason for this was the better coverages for the multiplicities of 2 ~ 5, with the exception of the multiplicity of 3 concurrent faults. When we allowed the RTL fault model this time to miss only one bit of the experimental fault then we noticed **100%** coverage. We did not notice a big dependence of the coverage in either case (when we allowed one mismatch or when we did not) on the gradual increase of the power.

This campaign therefore shows that the main reason for affecting at the same time the data and the key unit is that there was no placement constraint to separate cells belonging to the data-unit and the key-unit. As the laser spot was able to attack at the same time both units it is expected that the RTL fault model will fail in predicting this kind of errors. On the other hand the RTL fault model manages to cover 100% of injected experimental faults if there are no

errors involving both data and the key unit at the same time (by removing only a single error bit, in this manner).

In Table 4.7 we see the amount of covered and non-covered errors, classified by multiplicity. Table 4.8 and Table 4.9 show that 55.4% of the attacks lead to single bit errors and the remaining to multiple bit errors. The majority of covered multiple bit errors are the ones with multiplicities of 2 up to 5 concurrent erroneous bits. In combination with the total amount of errors per multiplicity we can see that the majority of the attacks lead to errors with multiplicities up to 5 concurrent erroneous bits.

Table 4.7: RTL fault coverage per multiplicity – no mismatch allowed

Multiplicity	1	2	3	4	5	6	7	8	9	10
Covered Errors	227	16	3	30	32	0	10	0	0	0
Non-covered Errors	0	1	10	20	13	16	4	8	12	8
Total errors per Multiplicity	227	17	13	50	45	16	14	8	12	8
Total Errors	410	410	410	410	410	410	410	410	410	410

Table 4.8: Coverage percentages over the total number of faults

Multiplicity	1	2	3	4	5	6	7	8	9	10
Covered Multiplicities (%)	55.4	3.9	0.7	7.3	7.8	0	2.4	0	0	0
Non-covered Multiplicities (%)	0	0.2	2.4	4.9	3.2	3.9	1	2	2.9	2
Total per Multiplicity (%)	55.4	4.1	3.2	12.2	11	3.9	3.4	2	2.9	2

Table 4.9: Coverage percentages over each fault multiplicity

Multiplicity	1	2	3	4	5	6	7	8	9	10
Covered Multiplicities	100	94.1	23.1	60	71.1	0	71.4	0	0	0
Non-covered Multiplicities	0	5.9	76.9	40	28.9	100	28.6	100	100	100

We present no tables for the case where we allow a mismatch of one bit between the experimental results and the RTL fault model since in this case and for this campaign the coverage was 100%.

4.2.4 Conclusions

Even though the fault injections are by no means exhaustive as exhaustive analyses would take huge amount of experimentation time, we have seen that the RTL fault model covers:

- 9.3% of all multi-bit errors during the first campaign
- 52.6% of all multi-bit errors during the second campaign

The results show, especially for the results of the second campaign, a large coverage. At the same time, at RTL the data-unit and the key-unit of the AES are completely separated (no logic cone intersection between them), while at the layout they are allowed to be placed near-by (sea of gates layout). This inconsistency can lead to experimental fault scenarios which are a priori not possible at RTL, according to the cone fault model.

On the other hand if we rule out the phenomenon when a multiple error combination in the data-unit is not covered due to one erroneous bit from the key-unit we achieve far better coverages:

- 91.2% of all multi-bit errors during the first campaign
- 100% of multi-bit errors during the second campaign

These results show that when the layout is completed by taking into account the results of the RTL tool and fault model then the RTL evaluation will have the opportunity to evaluate the design efficiently. Also these results show that the RTL approach managed to categorize the FFs of the design in a way compatible with localized laser fault injections.

4.2.5 Contributions

- Experimental validation
  - Definition of the experimental validation methodology
  - Performance of experimental laser fault injection campaigns on a protected AES, implemented on 28nm STmicroelectronics technology
  - Determination of the percentage of the experimental attacks which are covered by the proposed fault model

5 Countermeasures development against laser fault attacks

As we saw in the previous chapters, laser fault attacks can be used to inject into secure ICs either single or multiple faults. Such attacks may prove as effective means to produce exploitable faults on the functionality of secure integrated circuits. The capabilities of laser attacks are mainly attributed to the high accuracy they possess of precise spatial focus and accurate timing [5]. In order to avoid their exploitation, modern secure ICs should be capable to detect such attacks.

Although countermeasures can be developed at different levels of abstraction, the RTL abstraction level proves very useful since it takes place early in the design flow. One inherent disadvantage of the design of countermeasures at RTL is the missing information regarding synthesis, placement and routing. To overcome these disadvantages, in the current chapter we will use the RTL laser fault model which was defined and validated in the previous chapters. Our goal is to show how a relevant fault model can assist in the design flow by reducing the feedback loops from lower to higher levels of abstraction. In fact in the current chapter we want to show that the disposal of an RTL laser fault model can restrict the feedback loops between the design and the validation of the countermeasure within the RTL abstraction.

Countermeasures will inevitably increase the cost of the design and evaluation of an IC mainly because of the hardware overhead needed for its protection. Furthermore, countermeasures often employ: hardware redundancy, temporal redundancy, and information redundancy [49]. In the current chapter, we will describe a hybrid approach to protect secure ICs from laser attacks. This approach is based on both hardware and information redundancy so as to achieve high levels of protection with reasonable hardware overheads.

The advantage of the countermeasure presented in this chapter is that it can be applied to different secure circuits as it does not make use of the properties of a specific architecture. The goal of the countermeasure is to achieve 100% detection rates according to the laser attack RTL fault model, presented in this thesis. Furthermore, the goal is to provide a generic, effective and simple countermeasure at the RTL level that uses a hybrid approach of redundancy with a low additional area overhead.

Section 5.1 presents two fault models this work bases its analysis on. Section 5.2 is dedicated to the description of the implementation of our countermeasure. Section 5.3 describes a case study for an AES cryptographic circuit. Section 5.4 comprehensively presents the results about area overhead, fault injection error rates and detection capabilities, with respect to the layout, achieved by the protected AES circuit.

The work presented in this chapter has been published in [P7].

5.1 Information/hardware redundancy based countermeasures

Mostly, countermeasures that can be developed at RTL are redundancy based countermeasures. Such countermeasures can be categorized in: hardware redundancy, temporal redundancy and information redundancy. In the literature there exist countermeasures that are implemented at RTL for the protection against fault attacks but not specialized for laser fault attacks.

The hardware redundancy countermeasures are mainly implemented by replicating functional blocks of the circuit [50]. The results these separate blocks are producing are compared and if a difference is found that means that an error occurred in at least one of them. The drawback of this method is the large area overhead impact due to the replication.

The temporal redundancy countermeasures actually use the same hardware repetitively in order to compute the same process at least twice [51]. They can detect errors with minimum area penalty but this method entails large time overheads, and thus a big reduction in performance.

The information redundancy countermeasures are based on error detecting codes. An error detecting code is made of check bits generated from the information under protection, propagated through computation, and validated at the end. One technique, for block ciphers such as AES, is parity prediction [45]. Although the parity code is simple and cheap, the main disadvantage is that it cannot detect any even number of injected faults. Apart from parity, other solutions have been proposed that are more effective but entail bigger area overhead [52], [53]. According to [54], nonlinear robust codes provide theoretical guaranteed detection probability for each error. In [55] the authors provide results, which show that these (theoretical) detection probabilities also depend on structural properties of the protected circuits. Therefore, the detection probabilities are not always valid. Especially in [55], the authors have used gate hardening to strengthen nonlinear robust codes against laser fault attacks.

5.2 Countermeasure against Laser Fault Attacks

The implementation architecture of the countermeasure is based on a hybrid approach mixing spatial hardware redundancy and information redundancy. More specifically, the data which must be protected are separated in groups and one parity bit is used for each group. To maximize the effectiveness of such a countermeasure, at most one fault must be concurrently injected in each group. In the same sense a concurrent error detection scheme presented in [56] for the detection of single stuck-at faults. This countermeasure is not specialized for laser attacks and it proposes parity groups that will contain only one fault. In our work in order to achieve at most one fault on each parity group we use the results of the analysis of the design according to the types of concurrent faults predicted by the laser attack RTL fault model.

In Figure 5.1 we can see the principle of group parity generation with prediction [57] as a concurrent error detection scheme on which our countermeasure is based. A simple parity code is used as information redundancy, and the duplication of functional blocks for the prediction of parity is used as hardware redundancy. This countermeasure aims at protecting both the combinational and sequential instances of the design, which can be achieved due to the protection capabilities of group parity generation with prediction. The proposed countermeasure is based on three steps, the first step is the calculation of parity from the registered bits of the original design, the second step is the prediction of calculated parity with the duplicated components, and the final step is the comparison of these parities. A mismatch in calculated and predicted parity means that an error occurred in either the combinational or sequential instances. The comparator is indicating if an error occurred by outputting an error signal.

The proposed countermeasure is based on parity code. Parity bits are calculated from fixed predetermined vectors of FF bits Yi. Every vector of FF bits forms a parity group. Depending on the size of the vector, the protection that parity provides against laser attacks is characterized either robust or less robust. For example, one parity bit calculated for an 8-bit vector is less capable to detect an attack than one parity bit for a 4-bit vector as it will be explained later.

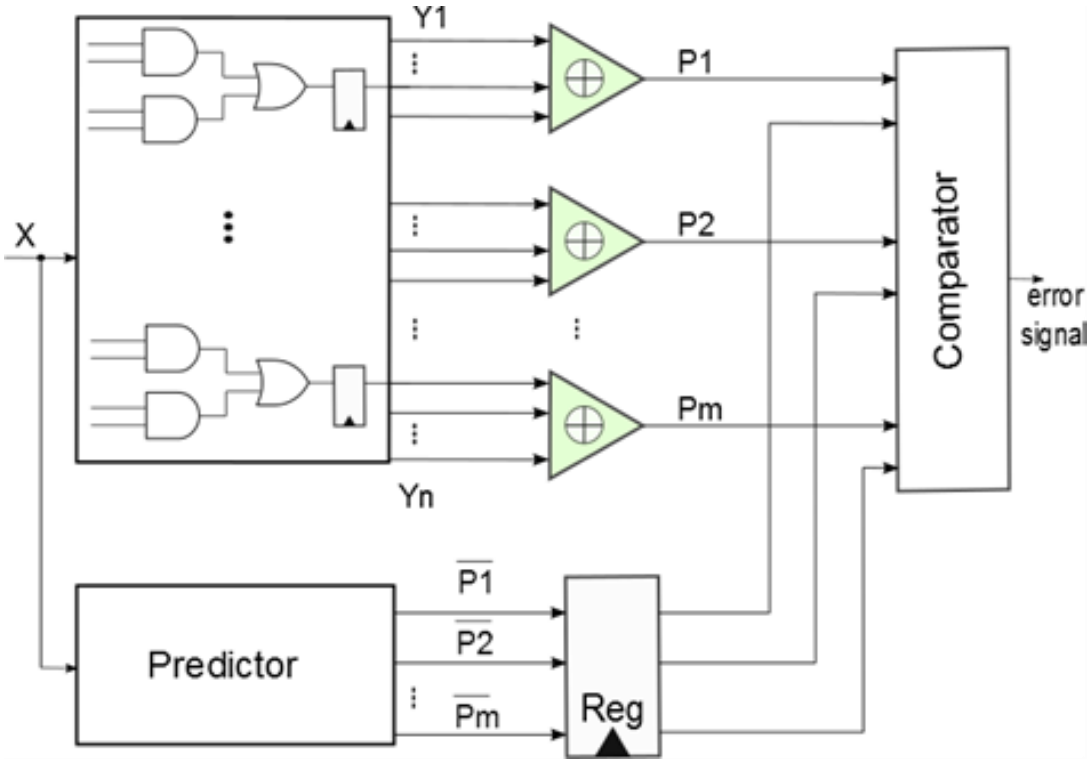


Figure 5.1: General countermeasure architecture: (1) Pi parity computed from the FFs of the original design (2) Pi-dashed parity predicted from inputs x (3) comparison of each parity Pi and Pi -dashed

The functional blocks that we want to protect are partially duplicated. These duplicated blocks constitute the predictor in our countermeasure (Fig. 1). During synthesis, the predictor and the original components are optimized by the EDA tools as much as possible so as to output only the parity bits. The gain in the overheads presented later is mainly due to this strategy. Secondly, in order to further reduce the area overhead imposed by duplication, a technique similar to [58] is used. This technique proposes the reduction of duplicated registers. In the duplicated components the n-bit output registers of the original design that are used to save information are replaced with 1-bit registers to save only the predicted parities.

Every calculated and predicted parity bit ends up in the comparator. The comparator consists of checkers that implement the comparison between the parity bits. For each parity pair, either predicted or calculated, there exists a checker that outputs an error bit signal which indicates if an error occurred or not in this specific combination. Consequently, the final sequence of error bits forms the output error signal vector.

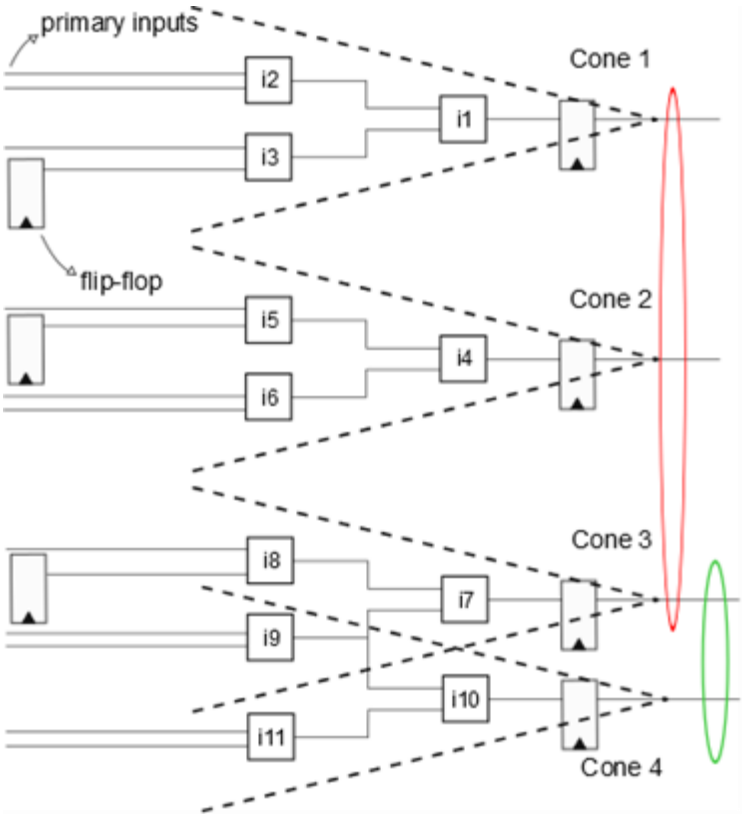


Figure 5.1: Cone partitioning and parity grouping example

In order to protect a circuit against laser attacks, the number and size of parity groups is critical and it is dependent on three factors. These three factors answer to three specific questions. The first question is how are the bits selected to be combined together in parity groups efficiently? The second question is how many bits are going to be in each group? The third question is which bits are possible to be included in the same parity group? These three factors are respectively answered by the following:

1. laser attack RTL fault model
2. robustness vs area tradeoff
3. architectural characteristics of the circuit in conjunction with the laser attack RTL fault model

The first answer is based on the laser attack RTL fault model analysis. The RTL tool which was described in the previous chapters provides the necessary information about the possible independent combinations of bits in parity groups. Furthermore it identifies the structural dependencies among all the logic cones. Independent cones are those that do not share any common elements; on the other hand, dependent cones are those that share at least one common element or primary input. Therefore, the possible candidates for the same parity group are independent cones. The laser attack RTL fault model also dictates that any laser attack will be confined inside a single logic cone: then, if each parity group of the countermeasure contains the outputs of independent logic cones, this will lead to a theoretical detection rate of 100%.

As can be seen in Figure 5.1, cones 1, 2 and 3 are independent and thus the output bits of these cones can be combined in a parity group. On the other hand, cones 3 and 4 are dependent and thus they cannot be part of the same parity group. The logic behind this reasoning is that cones that are independent in RTL design are more likely those that will be further away from each other after place-and-route compared to dependent cones. Thus, in case of a laser attack it is expected that it will be more unlikely to attack simultaneously independent cones than dependent ones, and a single laser shot will not succeed in injecting more than one fault into each group. In the end, the actual possible candidates for grouping are the output bits of FFs whose input logic belongs to independent cones. Although logic cones in RTL may differ from cones after synthesis, as we saw in the previous chapters the methodology provides good results, when it is validated versus the layout and actual experimental laser attacks.

The second factor that influences the implementation of the countermeasure is the effectiveness of the protection scheme, which depends on the size of the parity group. The bigger the group is, the more bits are included and concurrently protected in the same parity group. The smaller the group is, the fewer bits are included in the same parity group. Moreover, we know that the parity code cannot detect an even number of faults, occurring in the same protected vector. Hence, the bigger the parity group is, the bigger the possibility of an even number of faults to be injected inside the same parity group. This analysis, finally, ends up as a tradeoff between protection and area overhead. A representative example is going to be presented in the next section.

The third factor that affects the implementation of the countermeasure is the architectural characteristics of the design. The combination of bits in each parity group is determined by the structure of the circuit. The goal is to include all the possible candidate bits in parity groups in a way that will minimize the alteration of the original RTL design.

The proposed countermeasure is based on a simple and well known architecture, and it can protect both the combinational and sequential logic of the design. Finally, different versions of the countermeasure may be designed, depending on the tradeoffs between protection and area overhead. These aspects are going to be further analyzed in the following sections by applying the methodology to a practical AES description.

5.3 Case Study

In this section, we describe the procedure to implement our countermeasure on a secure circuit. The basic AES implementation, which is described in [59], is used without its countermeasures. In particular, the design has the advantage of a symmetric implementation by dividing the default 128-bit block in 16 blocks of 8-bits. The AES design consists of three basic components, which are: Data Unit (DU) (the encryption data path), Key Unit (KU) (round key generation) and Control Unit (CU). In our case study, we applied the proposed countermeasure to the DU and the KU components. The DU component consists of 16 blocks and each block operates on an 8-bit vector. The KU component is responsible for the secret key and works on a 128-bit vector which is the size of the key (also keys of 192-bits and 256-bits are allowed).

A cone analysis with the RTL tool that implements the laser attack RTL fault model indicates the possible combinations of bits which should be included in parity groups. As can be seen in Figure 5.2, the DU component consists of four rows and each row represents a 32-bit block. On each row there exist four identical 8-bit blocks that implement the main operations of AES. Each ellipsis in the diagonals is made of two cells and indicates different parity groups at the bit level. These are functionally independent groups which are extracted from the analysis of the RTL tool. As an example: a parity group consists of the first bit of the first cell of the first row (block A0) that is combined with the first bit of the second cell of the second row (block B1) and so on, until the eighth bit which is the last parity group of this ellipsis. Thus, each ellipsis also shows the combination of bits that form the parity groups.

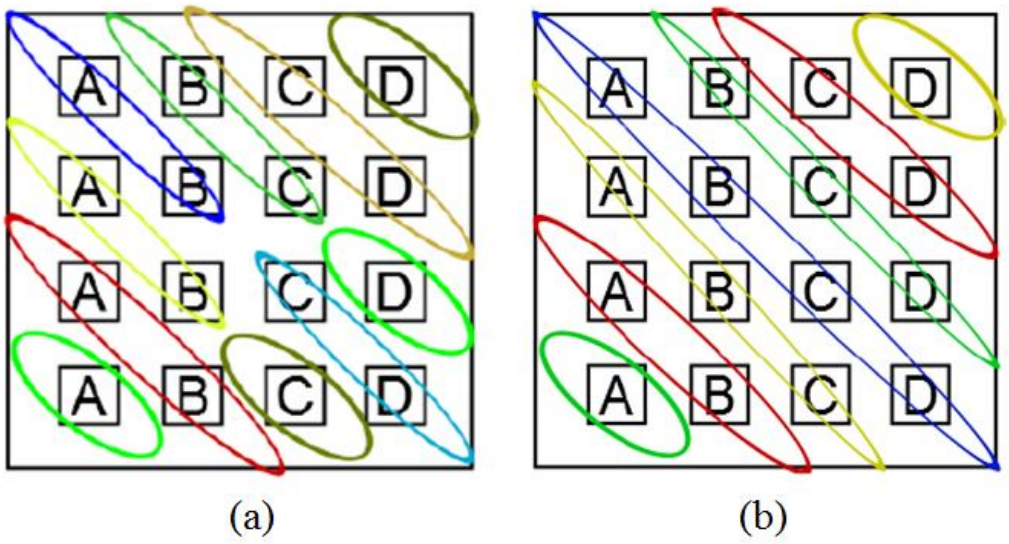


Figure 5.2: Parity group combinations based on the functional independencies. (a) categorization of the parity groups protecting the Data-Cell registers and the Key register, (b) categorization of the parity groups protecting the registers of the S-Boxes

The main countermeasure developed includes the full protection of DU and KU So in order to evaluate the efficiency of our countermeasure and to emphasize the tradeoff opportunities a second type of countermeasure has been developed. This second type of countermeasure is considered as a classical approach and aims at protecting groups of 8-bits from the same register without taking into account the dependencies of the logic cones. The implemented countermeasures hence are:

- Protection of DU and KU based on the laser attack RTL fault model with parity groups made from independent cones (FM)
- Protection of DU and KU, with parity groups made from dependent cones not using the laser attack RTL fault model (NFM)

The Data-Unit (DU) block is divided in a 4 by 4 array of cells equivalently to the state matrix. Each cell contains: two 8 bit registers, plus three 4 bit registers which belong to the S-box. On the other hand, the Key-Unit (KU) contains the 128 bit key register, one 32 bit regis-

ter which is used to hold the output of the S-boxes, plus three 4 bit register which belong to the S-boxes. The implemented countermeasure partitions these flip flops in parity groups. Initially, for each group their parity bit is computed. Then the parity of each group is predicted and compared with the computed parity. In total 176 parity groups were used to protect the design.

The RTL tool is used to analyze the design and extract which logic cones and FFs are independent with each other and which are not. The analysis has shown that logic cones which belong to the same cell are dependent. Furthermore, cones which belong to the same column are also dependent. Independent cones can be found for instance at the diagonals of the array (each cell is independent from its diagonal). This is the main property we have used to form the parity groups. There are two different categories of groups.

The first category concerns the protection of the Data-Cell registers (the two 8 bit registers of each cell giving 16 Data-Cell bits to be protected in each cell) and the 128 bit key register, in the following manner:

- A parity group contains:
  - The first bit is selected from one of the 16 Data-Cell bits.
  - One bit selected from one of the 16 Data-Cell bits of a cell located in the diagonal of the first bit, as in Figure 5.2 – (a).
  - One of the bits of the 128 bit key register.

This procedure is repeated until all the bits are categorized in parity groups. Therefore each group contains three bits (in addition to the parity bit). Also since there are 16 bits and 8 possible combinations (Figure 5.2 – (a)), we have 16 x 8 = 128 groups.

The second group category protects the registers which belong to the S-boxes. Thus we have three 4 bit registers of the Data-Unit S-Box for each cell (12 bits in total), three 4 bit registers for the KU S-Box , either in the DU or KU blocks in the following way:

- A parity group contains:
  - The first bit is one of the 12 bits of one DU S-Box
  - Three bits corresponding to the first, selected from the three cells which remain in its diagonal, as in Figure 5.2 – (b).
  - One of the 12 bits of one KU S-Box

Therefore there are 5 bits in each parity group and a total of 48 groups.

In total we have 128 + 48 = 176 parity groups to protect the DU and KU blocks.

5.4 Analysis of Countermeasures

Firstly, we evaluate the countermeasures in terms of area overhead, with respect to the original AES design. The countermeasures have been synthesized with the NANGATES 45nm open technology using Synopsys Design Compiler. The first synthesis strategy uses the simple-compilation options (SC), while the second one uses the ultra-compilation options (UC). Synthesis results can be seen in Table 5.1. This table shows that with the simple compilation option the FM countermeasure has approximately the same area overhead as the NFM countermeasure. On the other hand, with the ultra-compilation option, a difference in the area

overhead is observed. In fact, the NFM countermeasure is better optimized than the FM by approximately 12%.

Table 5.1: Coverage percentages over each fault multiplicity

Design	Synthesis	
	Area (μm2)	Overhead (%)
FM (SC)	28133	79.9
NFM (SC)	27870	78.3
FM (UC)	21191	63
NFM (UC)	19635	51

Secondly, we analyze the countermeasures with respect to two evaluation criteria. The first is based on RTL injection campaigns using the laser attack RTL fault model. These campaigns validate the expected theoretical 100% detection rate to be achieved by the proposed FM countermeasure. Also, it shows that the two countermeasures achieve different error rates and thus they provide different detection capabilities. The second evaluation criterion aims at

Table 5.2: Fault injection error rates for laser attack RTL fault model for a worst case margin of error of 10%.

Fault multiplicity		M2	M3	M4	M5	M6	M7	M8	M9	M10
FM	Detected	18.4	24	28.3	31.6	34.1	36.7	38.2	39.9	40.6
	Undetected	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	Silent	1.5	0.5	0.3	0.4	0.3	0.3	0.1	0	0
	False positive	79.9	75.4	71.3	67.9	65.5	62.9	61.6	60	59.3
	Final detection rate	**98.3**	**99.4**	**99.6**	**99.5**	**99.6**	**99.6**	**99.8**	**99.9**	**99.9**
NFM	Detected	19.9	28.2	30	35.1	35.2	39.1	38.3	41.8	40.8
	Undetected	2.9	0.2	2	0.1	1.7	0.1	1.3	0.1	1.4
	Silent	9.2	1.1	4.9	0.4	4.4	0.3	4.5	0.8	4.2
	False positive	68	70.5	63.1	64.4	58.7	60.5	55.9	57.3	53.7
	Final detection rate	**87.9**	**98.7**	**93.1**	**99.5**	**93.9**	**99.6**	**94.2**	**99.1**	**94.5**

practically evaluating the efficiency of the two countermeasures using layout information. Fault injection campaigns at RTL aim at simulating the effects of laser shots on the design according to the laser attack RTL fault model. The fault samples used for these campaigns are randomly generated with respect to a given multiplicity and margin of error, within each extracted set of FFs [22]. The final error rates are derived by comparing the outputs of the design with the golden values which are the fault free results. Table 5.2 includes the error rates achieved by the countermeasures for the laser attack RTL fault model. For the injection campaigns a fault multiplicity of 2 up to 10 has been chosen, per fault sample (M2-M10). For each multiplicity, the number of samples has been chosen, so as to achieve a worst case margin of error of 10%. In this case, the margin of error depends on the number of samples generated per set of FFs [22]. The more samples generated per set of FFs the smaller the margin of error is and vice versa. The categories of faults are the following:

- Detected: the faulty result is identified
- Undetected: the result is faulty, but no error detected
- Silent: the result is unchanged, and no error is shown
- False-positive: detection occurs, but no error in the result

Table 5.2 shows the final detection rates achieved and represents the detection capability of each countermeasure at RTL. The final detection rate is calculated by adding the detected and the false positive rates. The results show that the detection capability of the proposed countermeasure FM is very close to the theoretical detection rate of 100%. However, there are undetected faults in FM due to the faults injected in the Control Unit, which is unprotected. Also, we observe that the NFM countermeasure has a low detection capability for the even multiplicities. In particular, the small differences in the final detection rates between the countermeasures highlight the benefits of FM over NFM. We observe that the FM countermeasure has better detection rates in the even multiplicities, while in the odd multiplicities the NFM countermeasure presents also high detection rates. So a more representative analysis, taking into account the circuit layout, will help us evaluate further the countermeasures, and thus the physical locality of laser attacks.

The evaluation of the countermeasures in the layout, in terms of fault detection percentages, is based on the layout fault model. The analysis of the countermeasure's robustness with the layout fault model requires first to build a model of each countermeasure. Each countermeasure model consists of the names of FFs that form the parity groups and the FF that saves the predicted parity, for each group. Every parity group is paired with the corresponding FF that saves the predicted parity. After that, the layout fault model, obtained by partitioning the design in spots, provides two cases of affected instances: first case is when the FFs are directly affected by a laser attack, and the second case is when both combinational instances and FFs are affected by a laser attack. The second case is further analyzed in order to find potential propagated faults in the FFs that are driven from the combinational instances affected. These two cases are:

- Direct FF attacks (Multiple Event Upset, MEU)
- Direct FF & Combinational attacks (Multiple Event Upset & Multiple Event Transient, MEU & MET)

According to the layout validation approach, the layout of the design is partitioned in spots. Then for each spot the FFs which may capture faults of either combinational or sequential origin are extracted. These FFs are all potential candidates for multiple fault injections. At first we check if the FFs of a spot are all located in different parity groups. In this case any combination of faults originating from this spot will be detected since we will have at most one fault in each corresponding parity group. The spots which do not fulfill these criteria are further examined per multiplicity. Since all possible combinations for these cases generate a huge fault list, we apply a statistical approach [22]. For each spot we fix the maximum multiplicity of concurrent errors to ten. For every spot and multiplicity from 2 up to 10 (since we care for multiple faults, we ignore multiplicities of 1) we choose a number of samples which will lead to a margin of error of 5%.

Here we have to distinguish when a sample generated from a spot is considered as undetected and when as detected. The first undetected case is if an even number of FFs obtained from the examined sample is found in the same parity group. This means there is an even number of faults in the same parity group, which is undetectable with the parity code. The second undetected case is if an odd number of FFs is found in the same parity group but at the same time the corresponding FF that saves the predicted parity is affected. Any other case is considered as detected, and these are: if there are only an odd number of FFs affected in the same parity group or if only the predictor's FF is affected. We must stress here that for this analysis if at least one parity group leads to the detection of a fault, then this attack sample is considered as detected.

Table III presents the detection rates achieved for each countermeasure, i.e., the percentage of detected spots on each design, for the layout fault model and for two different spot sizes (size of the spot is the diameter of the laser beam). The table shows the detection capability of the countermeasures after layout. Firstly, the theoretical expectation of 100% detection rate for the FM countermeasure is closely confirmed. In fact, the proposed countermeasure is achieving very high detection rates, close to 100%, for the simple compilation option, and high detection rates for the ultra-compilation option. However, while the ultra-compilation option reduces significantly the area overhead, it does not reduce the detection rates proportionally. Although the countermeasures have approximately the same area overhead, the FM countermeasure achieves higher detection rates. In particular, the FM countermeasure preserves the high detection rates for both the affected cases (MEU, MEU & MET) while the NFM countermeasure has a big reduction in the detection rate between the two affected cases. This is mainly because the parity grouping strategy of our countermeasure takes into consideration the combinational logic of the circuit with the assistance of the laser attack RTL fault model.

Table 5.3: Fault detection percentages for layout fault model.

Design	Spot 1µm		Spot 5µm	
	MEU	MEU & MET	MEU	MEU & MET
FM (SC)	**99.6**	**99.1**	**98.8**	**98**
NFM (SC)	86.6	66.6	68.6	72.2
FM (UC)	**98.3**	**88**	**94.7**	**93.4**
NFM (UC)	84.4	57.3	68.8	73

5.5 Comparison of injections with the RTL cone fault model and the random fault model for early RTL evaluations of the implemented countermeasures

In this section we will present the fault injection results which were presented in the previous section in comparison with injection results by means of the totally random fault model. This fault injection campaign allows us to compare the cone fault model with the random fault model. Even though both fault injections have an associated margin of error of 5%, there are multiple points where they can be compared. In Table 5.4 we include the results for the FM countermeasure. For a multiplicity of two injected faults we notice that the random fault model achieves more than double the detection rate of the cone fault model. This result is

mainly due to the fact that the random fault model disperses the faults inside the circuit and therefore it is much easier to manage to inject an error inside an active part of the design. Furthermore, the parity detection is better if the faults are not in the same group, since there is a higher probability to inject an odd number of faults inside at least one parity group. At the same time false positive rates are 79.9% and 56.3% for the cone and the random fault model respectively. This difference is once again for the same reason, as the cone fault model injects localized faults and thus it manages to avoid the activation of the countermeasures. These patterns are consistently present for all the injected multiplicities up to ten injected faults. For both fault models, as the multiplicities increase we notice that the detection rates increase while the false positive rates decrease.

Table 5.4: Fault injection error rates for the cone and the random fault models achieved by DU & KU FM. Multiplicity of samples = 2 to 10. Margin of error = 5 %.

Fault injection error Rates – FM Countermeasure		**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**	**M10**
Cone fault model	**Detected**	18.4	24	28.3	31.6	34.1	36.7	38.2	39.9	40.6
	Undetected	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	Silent	1.5	0.5	0.3	0.4	0.3	0.3	0.1	0	0
	False positive	79.9	75.4	71.3	67.9	65.5	62.9	61.6	60	59.3
	Final detection rate	98.3	99.4	99.6	99.5	99.6	99.6	99.8	99.9	99.9
Random fault model	**Detected**	38	43.8	59.6	69.8	73.7	76.6	79.9	83.1	83.6
	Undetected	1	0	0.1	0	0	0	0	0	0.3
	Silent	4.7	1	0.5	0.3	0.3	0.3	0.3	0.8	0.3
	False positive	56.3	55.2	39.8	29.9	26	23.1	19.8	16.1	15.9
	Final detection rate	94.3	99.0	99.4	99.7	99.7	99.7	99.7	99.2	99.5

Exactly the same behavior appears for the fault injection results of the NFM countermeasure, presented in Table 5.5. Furthermore, as we can see in the two tables, the undetected faults for both countermeasures according to the random fault model are almost equal. The results show that if an RTL evaluation is performed during design time only by using the random fault model, then the two countermeasures seem to the designers as equally capable to detect fault attacks. On the other hand the cone fault model shows that there exists a difference and that the FM countermeasure is capable to protect the design against odd and even faults, equally well. This is not true in the case of the NFM countermeasure.

Table 5.5: Fault injection error rates for the cone and the random fault models achieved by DU & KU NFM. Multiplicity of samples = 2 to 10. Margin of error = 5 %.

Fault injection error Rates – NFM Countermeasure		M2	M3	M4	M5	M6	M7	M8	M9	M10
Cone fault model	Detected	19.9	28.2	30	35.1	35.2	39.1	38.3	41.8	40.8
	Undetected	2.9	0.2	2	0.1	1.7	0.1	1.3	0.1	1.4
	Silent	9.2	1.1	4.9	0.4	4.4	0.3	4.5	0.8	4.2
	False positive	68	70.5	63.1	64.4	58.7	60.5	55.9	57.3	53.7
	Final detection rate	87.9	98.7	93.1	99.5	93.9	99.6	94.2	99.1	94.5
Random fault model	Detected	43	54.2	63.5	66.9	73.7	79.9	85.7	86.6	89.1
	Undetected	0.3	0.5	0	0.3	0	0	0	0	0
	Silent	3.9	1.3	0.3	0	0	0	0	0	0
	False positive	52.8	44	36.2	32.8	26.3	20.1	14.3	10.4	10.9
	Final detection rate	95.8	98.2	99.7	99.7	100	100	100	97	100

We must also point out once again that in the case of the FM countermeasure all the undetected faults are due to faults injected strictly inside the unprotected control logic. Therefore the FM countermeasure has an undetected fault rate of 0%, a fact which was already known before performing the fault injections. On the contrary the NFM countermeasure's undetected rates contain fault scenarios injected inside the protected parts of the circuit.

5.6 Conclusion

This chapter presents a generic countermeasure for the protection of integrated circuits, especially developed to protect both the combinational and sequential instances of the circuit. This countermeasure was based on the laser attack RTL fault model, which has assisted in the implementation process and early evaluation of the countermeasure. The novelty of the countermeasure consists in providing an approach on how parity groups should be constructed in order to increase the detectability rates against laser attacks considering a dedicated high level laser attack RTL fault model. The evaluation includes RTL fault injection by means of the cone fault model, the random fault model. Furthermore, a layout based evaluation is used to further strengthen the results. The results show that the presented countermeasure can achieve higher fault detection rates than a similar countermeasure with approximately the same area overhead.

5.7 Contributions

- Definition and implementation of a new countermeasure against laser attacks based on the developed RTL laser fault model
  - Calculation of the area overheads, introduced by the countermeasure
- Validation of the countermeasure
  - Comparison of fault injection results based on the proposed and the random fault models
  - Definition of the layout-based validation flow
  - Comparison of the countermeasure with other existing countermeasures

6 General Conclusions

In the current thesis we have presented a new fault model (cone fault model), defined at RTL, in order to predict the effects of laser attacks on integrated circuits, early in the design flow. Furthermore, the fault model and methodology was applied to state of the art AES RTL architectures involving different countermeasures. The fault model generated the fault lists to be injected to the RTL descriptions and the fault injection campaigns have been performed by means of the partial reconfiguration FPGA-based emulator at TIMA laboratory. Separate fault injections have been completed on the same circuits also by means of the totally random fault injection approach (random fault model).

The results of these two evaluations were compared in order to evaluate the circuits, but also so as to compare the two fault models. The results have shown that the cone fault model was able to inject faults which were qualitatively different than the random fault model. Error characterization has shown that the cone fault model is able to inject faults which have better locality characteristics. For the same order of multiplicity, as for the random approach, it has managed to limit the dispersion of the faults inside the circuits. Therefore, it has also managed to determine fault combinations which lead to undetected faults even for larger multiplicities (of 2 up to 8 concurrently injected faults) for the AES Parity design. On the other hand, the random fault model was unable to point out the design's inability to detect odd and even multiplicities equally well. These results point out that the selection of the fault model can impact the evaluation and that if the utilized fault model is not accurate enough, it may lead to a false assurance of security. The second evaluated fault model which is has countermeasures based on hardware redundancy, highlighted another difference between the two fault models. The random fault model was a bigger impact on the circuit. With the same multiplicity of faults it was easier for the random approach to lead to detected faults as their dispersion allowed to affect more easily parts of the circuit which were active and performing computations. This point shows that the cone fault model is mode capable to inject localized faults.

Even though there are clear indications about the capability of the cone fault model to predict localized attacks better than the random approach during the fault injection campaigns it is important to also use the other levels of abstraction to validate the fault model. For this reason we also presented a layout based validation methodology which strengthens our confidence on the capabilities of the cone fault model. The methodology characterizes geometrical spots on a layout as covered or not by the cone fault model and the random fault model. It is shown that for spots of 1 and 5 μm, the majority of the layout spots are covered by the cone fault model. This analysis also takes into account the synthesis steps of the design flow and it shows that the RTL locality characteristics the cone fault model predicts are valid even after synthesis, placement and routing optimizations. Furthermore, an analysis of the relevant fault spaces shows that the fault space of the cone fault model and the layout validation method overlap in a great extent even for large multiplicities of injected faults. On the other hand, the random fault space and the layout fault space do not overlap equally well. At the same time, the RTL cone fault space and the layout fault space are many orders of magnitude smaller than the random fault space (of any possible fault combination). Due to this fact the majority of the fault combinations of the random fault space are faults which are not possible accord-

ing to the layout, for localized attacks. Therefore a possible outcome of relying on the random fault model during an evaluation is to decide to over-constrain the countermeasures which protect the design. This may in turn lead to larger design time and larger area and/or time countermeasure overheads which increase the cost and time-to-market of the implementation.

Thanks to the LIESSE project and in collaboration with CMP (Centre Microelectronique de Provence) we were able to further validate the capability of the cone fault model to predict fault which are actually possible during an experimental laser fault injection campaign. The fault injection campaign was performed on a 28nm STMictoelectronics technology implementation of the AES Parity design. After post processing, the results have shown that the cone fault model is able to predict a large percentage of the actual experimentally injected faults. Furthermore, there were strong indications that if the design hierarchy at RTL is used as a placement constraint on the layout, then the majority of the experimental faults (close to 100%) will be predicted by the cone fault model.

The validated cone fault model leads to the development of a new countermeasure against localized laser attacks. Besides laser attacks, it can also be used to protect integrated circuits against any kind of fault attacks with a highly local impact on the circuit. The countermeasure has a theoretical detection capability of 100% in detecting faults which are predicted by the cone fault model. Fault injection campaigns and analysis with layout information show that the true detection capabilities of the countermeasure are very close to the theoretical expectations. The advantages of this countermeasure are the demonstrated relatively low area overheads, as well as the fact that it can be applied easily to any RTL implementation.

Perspectives of this work include modifications of the cone fault model in order to include more than one logic cones as the attack origin. Additionally, it would be interesting to use the fault model from the beginning of the design procedure of a secure IC on each of its submodules and the comparison of this analysis with the application of the model on the completed design.

Concerning the new countermeasure some interesting perspectives include the automation of the countermeasure addition as an EDA tool without the need for the designer to manually implement it. Also further works can include the implementation of the countermeasure, on a variety of different designs, with FPGAs but also as an integrated circuit. Then laser fault injection on the protected designs can show the experimental capability of the countermeasure to protect against laser attacks.

7 Résumé en français

1 Introduction générale et état de l'art

1.1 Introduction

1.1.1 Vulnérabilités des circuits intégrés sécurisés aux attaques en fautes

Les circuits intégrés sont souvent considérés comme à l'origine de la confiance des systèmes embarqués pour lesquels la sécurité est nécessaire. Les exemples de systèmes embarqués sécurisés incluent, mais ne sont pas limités aux cartes de crédit, set top box, contrôle d'accès, ainsi que, dans des domaines plus récents, l'internet des objets (IoT). Les mécanismes de sécurité qui sont intégrés dans ces applications utilisent des implémentations cryptographiques. Néanmoins, les algorithmes cryptographiques, qui sont mises en œuvre sur des circuits intégrés, ne peuvent pas être tous considérés comme robustes aux attaques. Plusieurs techniques ont été développées par des pirates informatiques afin de révéler des informations importantes des algorithmes de chiffrement. Ces techniques comprennent différents types d'attaques à canaux cachés ainsi que les attaques par injection de fautes. Les résultats de ces attaques peuvent, ensuite, être utilisés par des cryptanalystes afin de casser la sécurité d'une application [1]. Ces attaques permettent, par exemple, de déduire la clé secrète d'un algorithme de cryptage ou de provoquer un déni de service. Les attaques par canaux cachés sont des attaques qui utilisent des fuites d'informations fournies par le matériel qui met en œuvre un algorithme cryptographique. Par exemple, les sources d'informations qui peuvent être utilisées pour effectuer ces attaques comprennent la consommation énergétique ou les émissions électromagnétiques d'un circuit intégré. En outre, ces attaques peuvent être utilisées afin d'aider l'attaquant à effectuer des attaques en fautes. Un des premiers types d'attaques en fautes appliquées aux cartes à puce a été l'attaque par glitchs. Un glitch fort est appliqué soit à l'alimentation, soit à la terre électrique ou à l'horloge de la carte. De cette façon, le circuit peut dysfonctionner et des fautes peuvent être générées au cours de son fonctionnement. Alors que l'attaque par glitch peut être réalisée facilement, elle affecte l'ensemble du circuit. Par conséquent, il est très difficile d'injecter des fautes à certaines parties spécifiques du circuit [2].

Les attaques en fautes basées sur des attaques par radiation ont été introduites dans [4]. Des sources intenses de lumière, comme un éclair, ou des lasers peuvent être utilisés pour perturber le fonctionnement d'un circuit intégré en induisant un courant dû à l'effet photoélectrique.

Entre toutes les différentes approches ci-dessus pour effectuer des attaques en fautes, l'injection de fautes par laser est une forme très efficace d'attaque [5], [6], [7]. Son efficacité se base principalement sur son excellente précision et le contrôle qu'il fournit à l'attaquant pendant une campagne d'injection de fautes. Plus précisément, les attaques au laser peuvent être utilisées pour injecter des fautes caractérisées par une très grande localité et une grande précision de la synchronisation, soit sur l'entrée d'une seule porte combinatoire (Single Transient Event - SET) ou dans une seule bascule (Single Event Upset - SEU). Des paramètres tels que la taille du spot, la durée d'impulsion, l'énergie et la répétitivité des impulsions peuvent être contrôlés avec précision pour atteindre un niveau de précision élevé des fautes injectés

dans un circuit. Ils peuvent également être utilisés pour induire dans les circuits plusieurs fautes. Ces fautes peuvent être soit des transitoires, soit des événements multiples [8], [9].

Différents types de lasers peuvent être utilisés pour effectuer une attaque par injection de fautes sur un circuit intégré. Parmi ces types de laser les plus chers permettent de contrôler la localité et la synchronisation. Un tel laser est capable de concentrer fortement le faisceau de laser et d'affecter des parties spécifiques d'un circuit intégré et, en plus, d'avoir des durées très courtes d'impulsion et de répétitivité rapide (ordre de la picoseconde). D'autre part, des équipements moins chers ont des capacités plus faibles, mais leur coût faible peut permettre des attaques sur des circuits intégrés pour lesquels l'investissement d'un attaquant dans l'équipement pour l'injection de fautes est faible. Les attaques peuvent être réalisées soit par la face avant, à travers les couches d'interconnexion, ou par la face arrière, à travers le substrat du circuit intégré.

Le coût qui est impliqué au cours de la conception et de la fabrication d'un prototype de circuit intégré impose la nécessité d'évaluer des circuits intégrés sécurisés sur différents niveaux de conception avant la fabrication. Une telle évaluation doit commencer tôt dans le flot de conception afin de réduire les coûts de la conception. Bien que les campagnes d'injection de fautes, au moment de la conception, puissent être effectuées à différents niveaux d'abstraction pour évaluer un circuit intégré sécurisé, l'évaluation ne sera utile que si les fautes injectées sont représentatives d'une attaque réelle après la fabrication. Les évaluations de l'état de l'art, au moment de la conception, sont effectuées sur les descriptions RTL et elles supposent généralement un modèle spécifique des fautes. Alors que les fautes simples ont une localité inhérente au cours des injections, au contraire, pour les fautes multiples, la localité des fautes simultanément injectées n'est pas généralement prise en compte. La raison est que si le placement et le routage ne sont pas terminés, il est difficile d'avoir des informations disponibles sur les combinaisons des fautes multiples qui peuvent se produire en même temps lors d'une attaque. L'information de layout est importante car un spot de laser peut affecter plus d'un des portes logiques et / ou des bascules et conduire aux fautes injectées, en particulier pour des technologies récentes submicroniques.

1.2 Évaluation des implémentations sécurisées à l'intérieur du flot de conception

L'aspect le plus important d'une injection par laser est sa capacité de perturber seulement une partie très spécifique d'un circuit intégré. Cela peut fournir à l'attaquant la possibilité d'injecter des fautes avec une contrôlabilité très bonne de la localité. Cette propriété est un aspect difficile de la modélisation des fautes du laser. La raison principale de cette difficulté est le manque d'informations de bas niveau (la synthèse, le placement et le routage). Un autre problème inhérent aux plus bas niveaux d'abstraction est que leur complexité croissante force la durée d'évaluation à augmenter très rapidement. De plus, les faiblesses de conception découvertes tard dans le flot de conception exigent des retours en arrière dans le flot très coûteux.

L'objectif de cette thèse est de mettre en œuvre et de valider un modèle de fautes laser au niveau d'abstraction RTL, qui peut être utilisé pendant les premières évaluations du flot. Un tel modèle de fautes peut contribuer à la réduction, voire l'élimination des re-spins de la conception entre les différents niveaux d'abstraction afin de mettre en œuvre des circuits intégrés résistants contre les attaques au laser.

Les sections suivantes de cette thèse sont organisées comme il suit. Dans la deuxième partie de ce chapitre, nous présentons des modèles de fautes et des outils qui sont utilisés dans la littérature pour effectuer des évaluations. De plus, nous soulignons pourquoi les outils RTL existants ne suffisent pas à être utilisés pour effectuer des évaluations des fautes par injection compte tenu des caractéristiques de la localité des attaques au laser. Dans le chapitre 2, un modèle RTL des fautes par laser qui tient compte des caractéristiques de la localité des attaques au laser est présenté. Chapitre 3 applique le modèle RTL des fautes développé à l'état de l'art du standard de l' AES (Advanced Encryption Standard - AES) et examine les résultats. Le chapitre 4 inclut la validation du modèle RTL des fautes à travers un flux de validation basée sur le layout et les résultats expérimentaux d'injection de fautes par laser. Dans le chapitre 5, nous présentons et évaluons une contremesure au niveau RTL pour la protection des circuits intégrés contre les attaques au laser. Chapitre 6 comprend les conclusions générales de cette thèse, ainsi que quelques perspectives.

1.3 État de l'art

Dans ce chapitre, nous commençons par résumer les principaux effets physiques par lesquels un laser peut induire des fautes à l'intérieur d'un circuit intégré. Ensuite, nous présentons les effets qu'une attaque laser peut avoir sur circuits et leur classification. Ensuite, nous présentons les modèles de fautes qui existent à ce jour dans la bibliographie. Nous commençons par résumer les modèles de fautes de niveau plus bas et nous nous concentrons au niveau porte et principalement des modèles au niveau RTL des fautes existants. La troisième partie du chapitre en cours décrit les différentes plateformes qui peuvent être utilisées pour effectuer des campagnes d'injection de fautes. De plus, nous présentons les méthodologies et modèles des fautes principaux qui sont jusqu'à présent utilisés pour modéliser des attaques laser. Dans la dernière partie, nous présentons le projet dans lequel cette thèse est réalisée et nous expliquons ses objectifs, ainsi que les différents partenaires du projet.

1.3.1 Injection de fautes par laser

1.3.1.1 Résumé des effets physiques de laser

L'illumination d'un semi-conducteur au moyen d'une source de laser peut provoquer des niveaux élevés d'ionisation [11]. Les photons qui influencent le semi-conducteur cause l'ionisation et générer des paires libres « électron-trou ». Cela se traduit par une charge électrique qui peut être collecté par des jonctions du circuit.

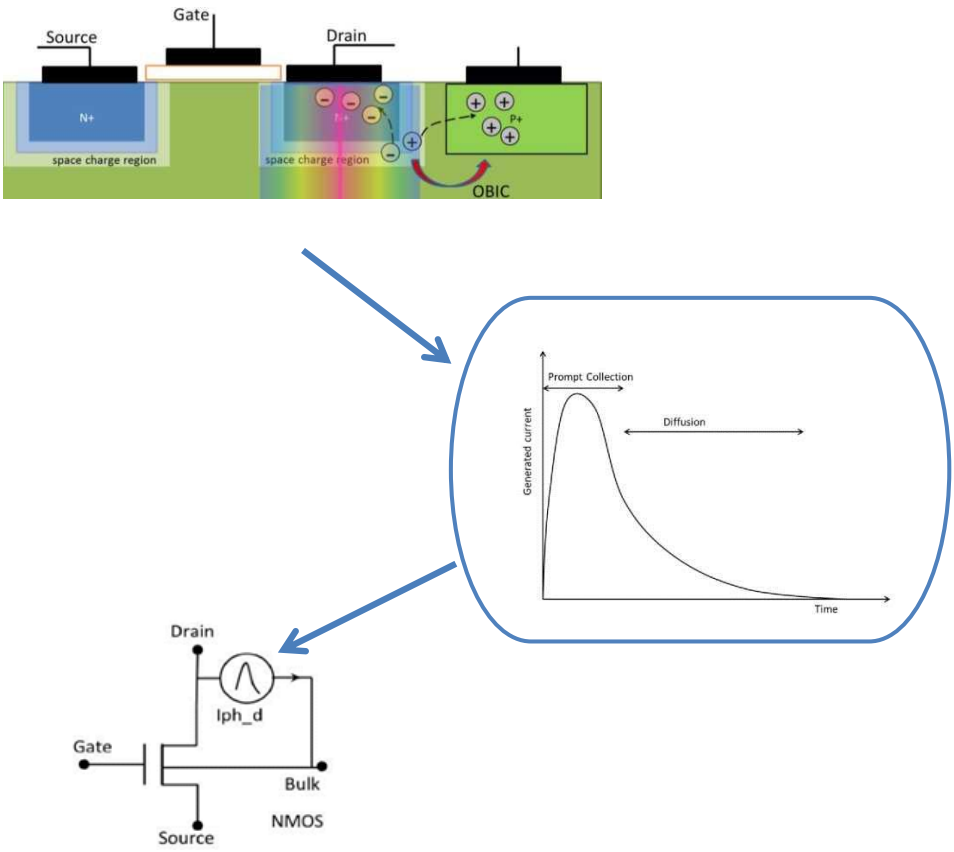


Figure 7.1: Photocourant induit au laser dans des circuits microélectroniques – [Feng Lu – PhD, Simulation de fautes par laser dans les circuits cryptographiques]

Les jonctions p/n polarisées en inverse sont capables de collecter cette charge, induite par le laser principalement par leurs régions d'appauvrissement et les champs électriques élevés qui existent là-bas [8]. Ces diodes polarisées en inverse forment par exemple entre le drain/source d'un transistor NMOS et le substrat. Cette collection de charge résulte finalement dans un photocourant transitoire au niveau des jonctions qui sont illuminés par le laser. Ce photocourant peut affecter les cellules logiques soit combinés ou séquentiels et les amener à fonctionner faussement.

Dans la figure 1.1, nous pouvons voir un incident du faisceau du laser à l'arrière d'un transistor NMOS. Le faisceau est focalisé sur le drain de NMOS et il induit des paires « électron-trou » dans la région. Cette charge est collectée par la région de charge autour du drain. Cela peut se traduire par une perturbation du courant au niveau du drain du transistor, comme dans la partie au milieu de la figure. Le courant transitoire peut être considéré comme une perturbation au niveau du nœud de drain du transistor NMOS et modélisé comme une source de courant en tant que dans la troisième partie de la figure.

1.3.1.2 Effets de laser sur la logique combinatoire et séquentielle

Quand une attaque au laser affecte la logique combinatoire, il peut induire des perturbations transitoires qui à son tour peuvent conduire aux erreurs au niveau des sorties des portes :

- Les fautes appelées Single Event Transient Faults (SET) sont des perturbations transitoires qui apparaissent à la sortie d'une porte logique affectée.
- Les fautes appelées Multiple Event Transient Faults (MET) apparaissent lorsque plusieurs portes logiques sont affectées simultanément.

Les éléments séquentiels à part le fait qu'ils peuvent être affectés et capturer des erreurs en raison de fautes SET ou MET, ils peuvent également être affectés directement si le faisceau du laser est directement concentré sur eux.

- Les effets appelés Single Event Upset (SEU): il se produit quand un tir du laser affecte un élément de la mémoire (par exemple la bascule) et cela conduit à une valeur erronée.
- Les effets appelés Multiple Event Upsets (MEU): ils se produisent lorsque plusieurs éléments de la mémoire sont affectés en même temps.

1.3.2 Modèles de fautes

Des modèles de fautes ont été utilisés pendant beaucoup d'années afin de prédire le comportement des circuits affectés par les fautes. Il existe plusieurs modèles dans tous les niveaux d'abstraction de manière à fournir la capacité aux concepteurs d'évaluer le comportement du circuit avant la fabrication.

Ces différentes modélisations sont la modélisation au niveau TCAD, au niveau SPICE, au niveau porte, au niveau du layout et au niveau RTL. Au niveau TCAD, nous rencontrons des modèles de fautes les plus élaborés à proximité de la réalité. Cela est principalement dû au fait que TCAD est le niveau d'abstraction qui est le plus proche de la physique des semi-conducteurs et peut donc directement décrire le phénomène de la charge induite dans le semi-conducteur. Les modèles de fautes au niveau SPICE peuvent être utilisés de manière à fournir des résultats plus précis et augmenter la vitesse de la simulation en comparaison avec la simulation au niveau TCAD.

En outre, la modélisation au niveau SPICE est plus proche à la procédure de conception en particulier pour les circuits analogiques. Dans les circuits numériques, les simulations au niveau SPICE peuvent caractériser le comportement des portes logiques affectées par des fautes. Malgré que la simulation au niveau SPICE soit plus rapide que la simulation TCAD, elle n'est pas pratique pour la validation de la conception numérique. Pour l'analyse des fautes, les approches classiques incluent l'utilisation des modèles de « bit-flipping » simple ou multiple et de « stuck-at » au niveau porte [16].

D'autres modèles de fautes utilisés au niveau porte comprennent : les fautes de court-circuit, les fautes transitoires ainsi que les fautes de délai. Le désavantage principal de ces modèles de fautes au niveau porte est que, concernant les injections multiples des fautes, l'espace des fautes explose même pour des multiplicités faibles. En plus, au niveau porte, s'il n'y a pas d'informations de placement, il est difficile de prédire quelle porte logique peut être affectée simultanément par un impact laser sur le circuit intégré.

Ensuite, le niveau layout est une description efficace pour identifier des cellules adjacentes et, donc, la localisation d'une attaque. Par conséquent, le layout est une source précieuse d'information pour les modèles de fautes qui essaient de décrire les fautes qui sont induites par les radiations [15]. Un défaut majeur des modèles au niveau layout est qu'ils sont complétés uniquement au cours de la phase finale du processus du flot de conception. Par conséquent, tous les problèmes critiques qui sont découverts par un tel modèle doivent impliquer des répétitions qui coûtent afin d'améliorer la robustesse d'une conception.

En plus, afin d'évaluer la fonctionnalité d'un circuit intégré sous des conditions erronées, beaucoup de modèles ont été proposés. Ceux-ci comprennent les fautes « stuck-at » ainsi que les fautes transitoires uniques de différents types, comme bit-flip, bit-set, bit-reset, transitoires dans des portes combinatoires. Initialement, la taille des nœuds technologiques ont autorisé la fausse fonctionnalité d'un circuit intégré, en raison soit de problèmes de fabrication ou d'un rayonnement dur, à modéliser par des types des fautes simples, même au niveau RTL.

1.3.3 Plateformes d'évaluation de l'injection de fautes

Afin d'évaluer une conception par injection de fautes pour les attaques au laser il y a les options soit de simulation ou d'émulation. Des campagnes d'injection de fautes basées sur la simulation peuvent avoir lieu à tous les niveaux d'abstraction. Une plateforme de simulation pour les attaques au laser est tLIFTING, qui est un simulateur des fautes basé sur le simulateur open source appelé LIFTING [24]. Avec les informations de géométrie du laser et le layout du circuit en tant que point de départ, il simule l'effet des fautes avec un modèle de fautes au niveau électrique. Les autres campagnes basées sur l'émulation ont généralement lieu au niveau RTL en utilisant des FPGAs. L'accélération qui est obtenue par une émulation sur FPGA est importante.

Afin de minimiser la durée des évaluations au niveau RTL, différentes techniques et plateformes d'émulation des fautes ont été développées. Même si les modèles de fautes définis au niveau RTL permettent une réduction de la durée des campagnes d'injection de fautes par simulation, le passage à l'injection de plusieurs fautes en même temps génère le besoin des techniques encore plus rapides. L'émulation des fautes en utilisant des plateformes FPGA permet une accélération drastique des campagnes d'injection de fautes. Certaines de ces techniques, utilisées pour réaliser l'instrumentation du dispositif sous test (device under test - DUT), sont appelés des techniques de modification de code HDL [25]. Les techniques d'émulation des fautes ont l'avantage d'être très rapides et injecter des fautes dans tous les nœud souhaités de la conception qui est bien défini au niveau RTL. Leur désavantage principal est qu'ils augmentent la complexité.

1.3.3.1 Injections statistiques aléatoires des fautes

Lorsque les modèles de fautes, qui ont été décrits ci-dessus, sont utilisées pour injecter des fautes multiples, la question d'avoir une collection immense des fautes arrive toujours. Cela conduit aux campagnes d'injection de fautes de très longue durée, même pour les plateformes d'émulation de fautes. Une méthode pour circonvenir ce problème est d'effectuer une injection de fautes statistique. Les auteurs présentent une méthode d'injection statistique aléatoire

des fautes, et des résultats des injections aléatoires des fautes sur des implémentations sécurisées [22].

L'hypothèse principale de cette approche aléatoire est que les caractéristiques des populations (toutes les fautes à tous les cycles d'horloge) suivent une distribution uniforme.

1.3.4 Projet LIESSE

Ce travail est une partie du projet ANR "LIESSE". Les objectifs principaux de ce projet sont l'étude et la modélisation de l'effet des attaques laser sur les circuits submicroniques et de fournir des outils de CAO efficaces pour les concepteurs de circuits pour prédire les effets de ces attaques laser lors de la conception de circuits. D'autres partenaires travaillent à des niveaux inférieurs au niveau RTL, y compris pour les essais expérimentaux et la caractérisation des blocs principaux de circuits, ainsi que la mise en œuvre des modèles de CAO concernant des attaques au laser sur des circuits intégrés sécurisés. L'interaction entre les partenaires et les résultats dans tous ces niveaux d'abstraction sont utiles en particulier pour la validation des outils et des modèles de fautes développés. Par exemple, des injections expérimentales de fautes laser sont très important afin de valider l'approche au niveau RTL présenté dans cette thèse. En outre, l'évaluation d'une conception avec les modèles de fautes qui ont été développés dans le cadre du projet "LIESSE" peut renforcer les conclusions sur la résilience de la conception contre les attaques au laser.

Un autre objectif du projet est de proposer des contre-mesures afin de renforcer les implémentations contre ces attaques laser.

1.3.5 Conclusions

Dans le cas des fautes provoquées par un laser et en particulier dans le contexte de la sécurité, l'analyse des fautes doit traiter efficacement la complexité supplémentaire imposée par les caractéristiques du laser. La complexité provient du fait qu'une attaque au laser, surtout dans les dernières technologies de fabrication, permet à l'attaquant une excellente maniabilité de la localité et du timing. Même une taille de spot minimale de l'ordre de 1 µm affecte plusieurs éléments. Un seul bit flipping dans les registres ne décrit pas le phénomène avec précision et des modèles de fautes multiples doivent être utilisés [6].

Dans la littérature, il n'existe pas un modèle complet de fautes laser au niveau RTL. Dans plusieurs approches différentes, des plateformes génériques d'injection de fautes sont utilisées [26], [28], avec la possibilité d'introduire des fautes multiples, soit par simulation ou émulation, mais sans aucune corrélation avec les capacités d'un laser.

2 Modélisation des attaques laser au niveau RTL

La modélisation des attaques en fautes localisées au début du flot de conception est un atout précieux pour le développement du matériel sécurisé. Un modèle de fautes capable de modéliser l'injection de fautes localisée peut être utilisé pour prédire l'effet des attaques en fautes du laser. Le défi principal, afin de développer un modèle de fautes au niveau RTL est que beaucoup d'informations manquent au cours des premières étapes de la conception.

2.1 Propriétés d'attaque en fautes par laser

Comme déjà expliqué dans l'état de l'art, les lasers permettent des attaques en fautes précises. Certaines propriétés essentielles d'un laser qui définissent également les capacités d'injection d'un attaquant sont les suivantes:

- Localisation
- Temps
- Durée d'impulsion
- Taux de répétitions

Dans les propriétés ci-dessus, la localité d'injection est l'une des plus importante.

2.2 Netlist élaborée au niveau RTL

La netlist élaborée au niveau RTL, à partir de maintenant appelé simplement « RTL netlist » [32], est constitué de cellules génériques du niveau RTL non rattachées à une technologie spécifique du circuit intégré. Ces cellules peuvent être des opérateurs du niveau RTL fonctionnels ou des portes de base génériques et d'autres RTL éléments (par exemple, les multiplexeurs). L'un des avantages de la netlist au niveau RLT est qu'elle contient exactement les mêmes fonctionnalités que la description au niveau RTL. En même temps, elle a une structure « netlist » à partir de laquelle les propriétés fonctionnelles utiles de la conception peuvent être facilement extraites de manière automatique. La netlist du niveau RTL élaborée est un choix très approprié pour modéliser des attaques locales au niveau RTL.

2.3 Abstraction logique du cône et dépendances fonctionnelles

Pour modéliser une attaque, une option est de sélectionner des cellules spécifiques de la netlist élaborée du niveau RTL comme étant son origine. Afin de modéliser la localité des attaques au laser au niveau RTL et de fournir une bonne approximation de l'origine de l'attaque, nous effectuons le partitionnement logique de la description. Ce partitionnement au niveau porte a été utilisé dans le passé pour effectuer automatiquement le diagnostic des fautes et de localiser l'origine d'un ou plusieurs fautes multiples, étant donné certains effets des fautes [34], [35].

Pour notre analyse, nous considérons les cônes qui sont délimités par une bascule de départ (père) et vers l'arrière, jusqu'à ce que nous rencontrons soit les bascules ou les entrées primaires du circuit. La Figure 2.4 montre un cône logique et ses limites.

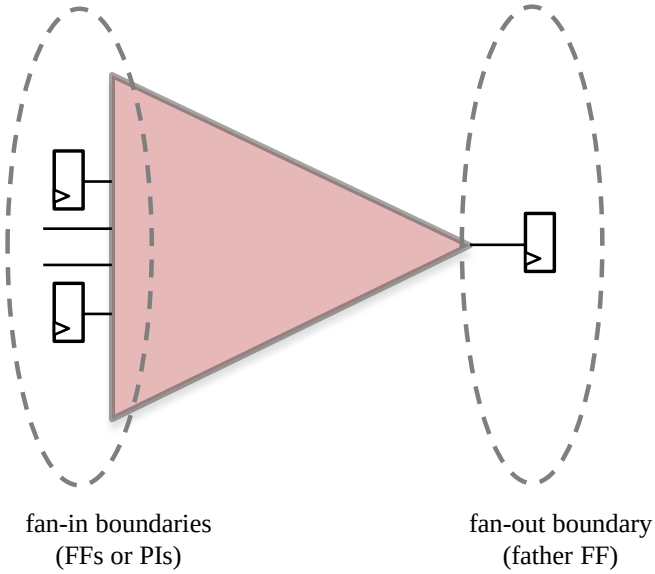


Figure 2.4: Limites de fan-in et fan-out du cône logique

Les cônes logiques et les bascules qui dépendent l'un de l'autre sont les suivants :

- Deux cônes logiques seront définis comme fonctionnellement dépendants s'ils contiennent au moins un élément ou une câble commun (ils se croisent).
- De même façon, deux ou plusieurs bascules dépendent fonctionnellement si leurs cônes logiques sont également fonctionnellement dépendants (ils se croisent).

2.4 Description du modèle de fautes au niveau RTL

2.4.1 Hypothèse du cône singulier affecté

- Hypothèse 1 - Origine de l'attaque :

L'origine d'une attaque laser est contenue à l'intérieur d'un seul cône logique. Par conséquent, l'origine de chaque attaque affecte le seul cône choisi et, en même temps, tous les cônes avec lesquels ce cône se croise.

En termes de dépendances des bascules, cela équivaut à supposer qu'un attaquant peut cibler simultanément une bascule et, en même temps, toutes les bascules qui sont fonctionnellement dépendantes d'elle.

2.4.2 Capture de l'attaque

- Hypothèse 2 – capture de l'attaque

Une attaque combinatoire est capturée, dans un cycle d'horloge, à travers les cônes logiques (dépendances fonctionnelles) qui ont des bascules situés dans le fan-out de chaque cône logique.
Par conséquent, pour une origine donnée (cône), nous pouvons obtenir les destinataires de l'attaque, si nous croisons l'origine (cône) avec tous les cônes logiques restants du circuit.

L'exemple de la figure 2.5 (a) illustre la situation dans laquelle le cône 1 est considéré comme étant l'origine d'attaque (cône rouge). Comme nous pouvons voir toute faute induite peut se propager aux bascules fan-out des cônes 1 et 2. Par conséquent, afin de déterminer les destinataires d'attaque pour cette attaque, nous devons extraire tous les cônes logiques que l'origine de l'attaque recoupe. La Figure 2.5 (b) illustre la situation où une attaque localisée affecte seulement le cône 4.

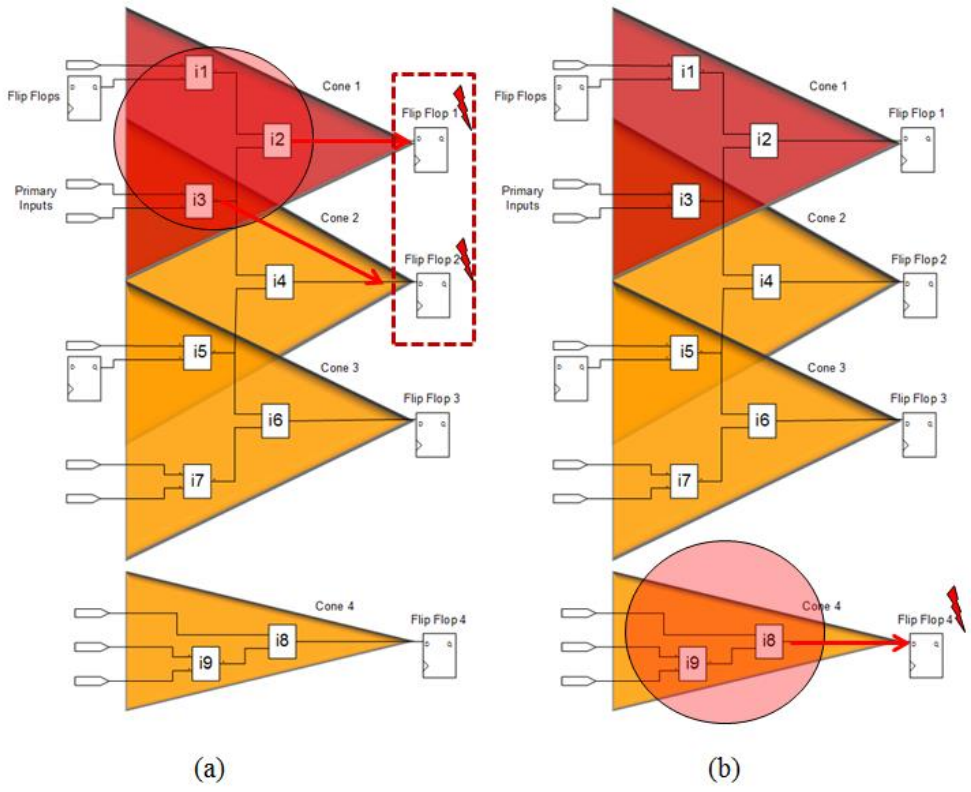


Figure 2.5: Extraction des destinataires de l'attaque étant donné les hypothèses pour l'origine et la capture de l'attaque, (a) Les destinataires lorsque le cône 1 est affecté, (b) Destinataires lorsque le cône 4 est affecté. Les deux scénarios couvrent les cas où une attaque (spot du laser) affecte seulement le cône 1 ou seulement le cône 2.

Un inconvénient est qu'il est possible d'avoir des scénarios de l'attaque où un spot laser sur le niveau physique est en mesure d'affecter en même temps deux cônes qui ne se croisent pas.

2.4.3 Les attaques directes des bascules

- Hypothèse 3 - Les attaques directes sur les bascules

  Les attaques directes causant des fautes multiples sur les bascules sont possibles sur des combinaisons de bascules qui sont fonctionnellement dépendantes (leurs cônes logiques se croisent).

2.5 Génération de la liste des fautes

Après la génération des ensembles de destinataires pour toutes les attaques possibles, l'étape suivante consiste à générer la liste des fautes qui doivent être injectées dans le circuit de manière à l'évaluer par injection de fautes.

2.6 La mise en œuvre d'un outil EDA

Afin d'effectuer le partitionnement, un front-end VHDL et Verilog, fourni par Verific Design Automation Inc., a été utilisé [36]. Une description VHDL ou Verilog est d'abord analysée, suivie par l'élaboration au niveau RTL. Le résultat est une netlist de base de données contenant des primitives principales de "Verific". Le C++ API du front-end est ensuite utilisé pour mettre en œuvre les algorithmes de l'analyse.

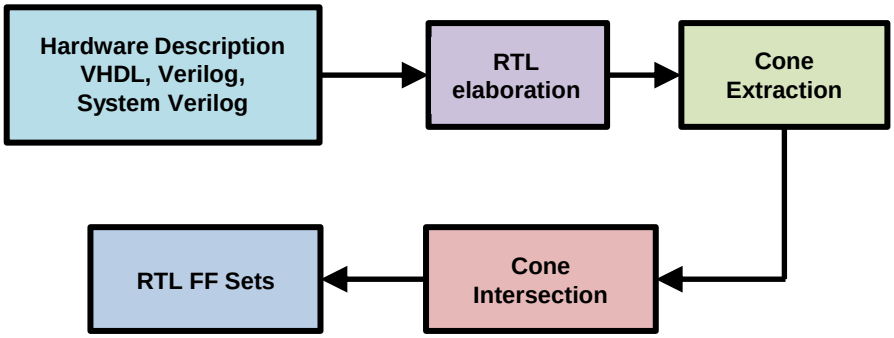


Figure 2.6: Processus d'outil EDA pour l'extraction des ensembles de destinataires de l'attaque (changement des bascules au niveau RTL avec les ensembles de destinataires de l'attaque)

3 Evaluation de descriptions RTL contre les attaques laser

Dans ce chapitre, nous présentons les résultats des campagnes d'injection de fautes émulées par rapport au modèle de fautes du cône et au modèle de fautes aléatoires. L'objectif principal est de comparer les deux modèles de fautes basés sur les résultats de l'évaluation de deux architectures d'AES avec des contremesures différentes.

Nous allons décrire brièvement l'émulateur développé au laboratoire TIMA pour effectuer les injections des fautes. Ensuite, nous allons fournir des informations sur la génération des échantillons de fautes selon les modèles de fautes appliqués. Puis, nous allons utiliser les résultats expérimentaux afin de déterminer le type des fautes qui est plus appropriée pour l'évaluation.

3.1 L'échantillonnage statistique des fautes

Pour les conceptions complexes, il est impossible d'effectuer des campagnes d'injection de fautes exhaustives, même en utilisant l'émulation. Par conséquent, l'injection statistique de fautes (Statistical Fault Injection - SFI) est principalement utilisée pour faire face aux espaces énormes de fautes. La référence [22] présente une approche rigoureuse de façon à appliquer le SFI avec une marge d'erreur quantifiée et un intervalle de confiance.

3.2 Evaluation des résultats de la conception et de la contremesure au niveau RTL

3.2.2 Comparaison de deux approches

Dans le tableau 3.5, nous présentons les résultats obtenus avec le modèle aléatoire et le modèle de fautes du cône pour l'AES à base de parité. En outre, dans la figure 3.1, nous avons inclus les graphiques des résultats en fonction des mêmes.

Tableau 3.5 Injections de bit-flips avec multiplicité jusqu'à 8 sur l'AES parité avec les bits erronés uniformément étalés dans toutes les bascules ou limitées par le modèle du cône

Multiplicité	Campagne de l'injection	Silencieux	Erreurs non-détectées	Erreurs détectées	Faux positive	AES Crash
2	Touts les bascules	36.88%	3.38%	25.45%	31.95%	2.34%
	Groupes locales	52.24%	8.39%	12.38%	25.55%	1.44%
3	Touts les bascules	30.65%	2.34%	34.28%	29.09%	3.64%
	Groupes locales	42.23%	1.29%	20.07%	34.64%	1.77%
4	Touts les bascules	22.08%	0.52%	44.41%	29.87%	3.12%
	Groupes locales	50.93%	8.21%	14.32%	24.44%	2.1%
5	Touts les bascules	16.88%	0.78%	44.68%	30.13%	7.53%
	Groupes locales	41.63%	0.95%	21.67%	33.53%	2.22%
6	Touts les bascules	10.91%	2.08%	47.01%	31.17%	8.83%
	Groupes locales	49.61%	7.96%	17.82%	22.21%	2.4%
7	Touts les bascules	9.09%	1.3%	52.47%	30.13%	7.01%
	Groupes locales	37.92%	0.69%	23.68%	34.98%	2.73%
8	Touts les bascules	5.97%	0%	54.81%	27.79%	11.43%
	Groupes locales	27.62%	5.92%	29.33%	33.31%	3.82%

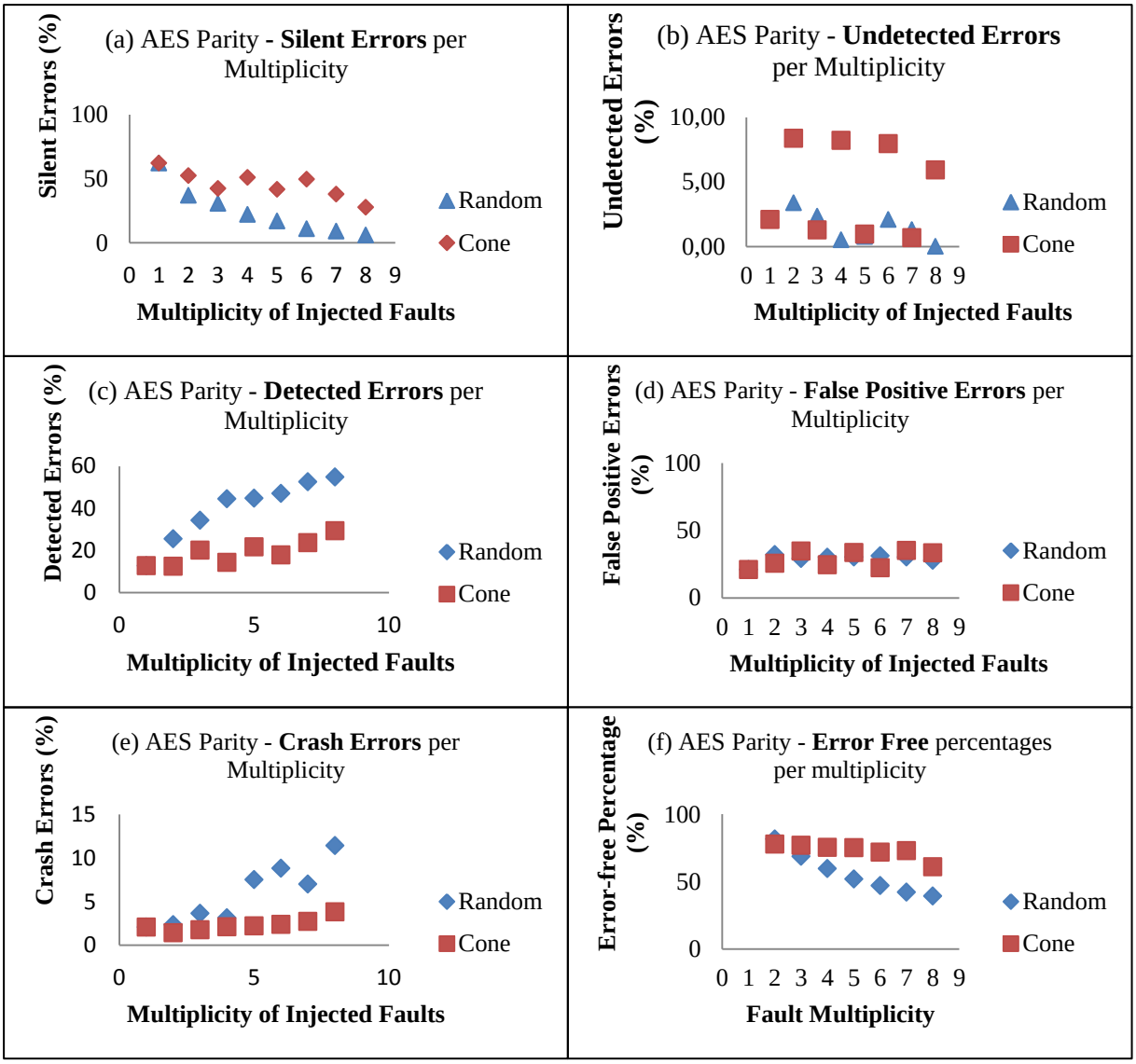


Figure 3.1 AES à base de parité, les pourcentages de fautes silencieuses par la multiplicité des modèles de fautes aléatoires et du cône

La figure 3.1 (a) contient les résultats des fautes silencieuses, où nous pouvons voir une nette différence entre les deux approches. Le modèle de fautes aléatoire a des taux croissants de fautes silencieuses en fonction de la multiplicité des fautes injectés. Au contraire lorsque nous avons utilisé le modèle de fautes du cône la conception a des taux de fautes silencieuses plus stables. En ce qui concerne les erreurs faux positifs (Figure 3.1 (d)), nous pouvons remarquer que pour les deux modèles de fautes le taux est constant (dans la marge d'erreur) et indépendants de la multiplicité des fautes injectées simultanément. Plus précisément, les résultats montrent que le pourcentage de fautes silencieuses en ce qui concerne la multiplicité de fautes est beaucoup plus proche au modèle de fautes du cône que le modèle de fautes aléatoires, même dans le cas d'un assez grand spot du laser par rapport à la taille du transistor.

La figure 3.1 (f), montre le taux des injections ne produisant pas d'erreurs. Le taux pour l'injection de fautes aléatoires suit le comportement de la catégorie des fautes silencieuses tandis que le modèle de fautes du cône produit des taux sans erreur presque constants. Comme

déjà discuté dans les sections précédentes, cela montre que la méthode du cône est plus capable de limiter les fautes injectées à une région fonctionnelle spécifique. Ceci peut être déduit par le fait que le taux sans faute comprend l'injection de fautes dans les régions inactives du circuit. Par conséquent, le taux constant montre que les fautes générées par le modèle de fautes du cône n'augmentent pas la probabilité d'affecter une cible active quand les multiplicités augmentent. Le taux de fautes non détectées de la figure 3.1 (b), montre que le modèle de fautes du cône est en mesure d'indiquer clairement aux concepteurs que les contremesures en cours d'évaluation n'ont pas les mêmes capacités de détection de fautes injectés paires et impaires. En outre, l'approche aléatoire, même si elle montre aussi des différences entre les multiplicités paires et impaires (mais pas uniforme), le taux de fautes pour des multiplicités nombreuses est très proche à zéro. La méthode du cône indique clairement que les fautes non détectées paires et impaires ne sont pas très proches à zéro. Cette analyse montre que le modèle de fautes aléatoire peut dans ce cas conduire à une fausse assurance de la sécurité.

Figure 3.1 (c), montre qu'une nette différence entre les deux modèles de fautes existe aussi dans le cas des taux de détection des fautes. L'approche aléatoire indique des capacités de détection plus élevées quand les multiplicités augmentent alors que la méthode du cône produit des résultats plus stables.

3.3 Résultats

Les résultats de l'injection de fautes présentées dans ce chapitre montrent que le modèle de fautes considéré a un grand intérêt pour l'évaluation tôt dans le flot de conception. Les deux modèles de fautes aléatoires et du cône sont définis par des hypothèses différentes. D'une part, le modèle de fautes aléatoires suppose qu'il est probable pour une attaque d'injecter une faute à toutes les bascules de la conception. D'autre part, le modèle de fautes du cône suppose que seulement un ensemble du cône logique peut être affecté en même temps (et tous les cônes avec lesquels le cône affecté se croise). Par conséquent, il suppose que seulement les bascules dépendantes de la fonctionnalité peuvent être destinataires des plusieurs fautes simultanées. Les résultats de taux de fautes obtenus par les deux modèles de fautes sont qualitativement et quantitativement différents. Par ailleurs, l'analyse des résultats indique que le modèle de fautes du cône est plus capable de suivre les propriétés de chacune des contremesures évaluées. Dans le cas de l'AES à base de parité, le modèle de fautes du cône était capable de mesurer la capacité de la contremesure à détecter les fautes paires et impaires. En outre, concernant le modèle de fautes du cône, le taux sans erreur est plus stable et cela montre qu'il peut mieux contraindre l'injection de fautes aux régions fonctionnelles spécifiques de la conception.

4 Validation du modèle de fautes proposée

4.1 Validation par rapport au layout

L'objectif de l'approche de la validation présentée dans cette section est de valider la propriété de localité que le modèle de fautes du cône prédit au niveau RTL.

4.1.1 Approche de la validation

Les hypothèses du modèle de fautes du cône doivent être. Par conséquent, nous avons besoin de valider si l'hypothèse, qu'un seul cône logique peut être attaquée en même temps, est valable (hypothèse 1 du chapitre 2) et dans quelle mesure c'est bien le cas. La deuxième hypothèse à évaluer est si les fautes, qui ont leur origine à l'intérieur d'un cône logique, se propagent uniquement aux bascules père (hypothèse 2 du chapitre 2). Cela peut ne pas être valable dans le cas où la synthèse et les optimisations, qui se produisent après le niveau netlist élaboré, modifient radicalement les intersections des cônes logiques (modifient les dépendances fonctionnelles). En outre, nous avons également besoin de vérifier l'hypothèse selon laquelle les bascules avec des dépendances fonctionnelles sont plus susceptibles d'être placés à proximité et, donc, de vérifier dans quelle mesure le modèle est capable de prédire les attaques directes des bascules (hypothèse 3 du chapitre 2).

La procédure de validation, qui est décrite en détail dans [P6], commence avec le partitionnement des cônes logique des descriptions au niveau porte. Ceci permet la mise en œuvre de la même méthodologie, qui a été présenté dans le chapitre 2, mais cette fois au niveau porte de façon à comparer les résultats obtenus avant et après la synthèse. En outre, de cette manière on peut aussi obtenir l'espace des fautes au niveau porte en fonction du modèle de fautes du cône. Ainsi, nous sommes en mesure d'obtenir l'ensemble potentiels des destinataires de l'attaque qui peut capturer des fautes au niveau porte (niveau porte du modèle de fautes du cône). En outre, le partitionnement des cônes au niveau porte permet la connexion entre le niveau RTL et le niveau layout comme il existe une correspondance un à un entre le niveau RTL et les cônes logiques du niveau porte.

Dans la figure 4.1, nous indiquons une attaque locale limitée dans un carré sur le layout. Pour l'application de l'approche de validation, ce carré est défini comme l'origine de l'attaque au niveau du layout. La forme carrée a été choisie principalement pour des raisons techniques (en raison de l'interface OpenAccess). En outre, il conduit à des résultats plus pessimistes de la validation, comme il est utilisé pour décrire le (petit) cercle inclusif, qui représente la forme actuelle du spot du laser. En extrayant son contenu, on obtient l'ensemble des cônes au niveau porte qui sont touchés par cette attaque. Le partitionnement des cônes au niveau porte fournit, alors, la capture de l'attaque aux destinataires de l'attaque considérée. Encore une fois, les destinataires de l'attaque sont les bascules qui peuvent potentiellement capturer une attaque. Ainsi, nous sommes en mesure de déterminer l'ensemble des bascules (destinataires de l'attaque) qui peuvent contenir des valeurs erronées, après une attaque à l'intérieur du carré. La capture de l'attaque considère tous les chemins fonctionnels possibles pour déterminer les destinataires de l'attaque, l'espace de fautes généré ne dépend pas d'entrées spécifiques du

circuit ou de son état pendant le temps que l'attaque se produit. En outre, nous ne considérons pas les effets physiques ou structurels qui nécessitent des informations non disponibles au niveau RTL. Par conséquent, nous ne prenons pas en compte les effets sur l'horloge et sur la réinitialisation des arbres, ainsi que sur la distribution de la puissance et de la terre électrique.

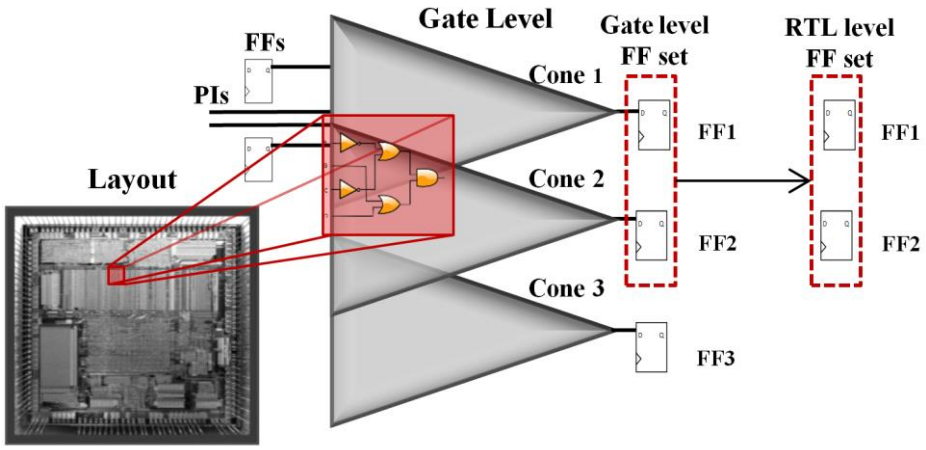


Figure 4.2: Extraction de l'ensemble du destinataire de l'attaque (bascules) affectée par une attaque locale

La procédure ci-dessus est répétée et une analyse de la puce est réalisée, selon la taille du carré (spot) et la taille du pas. Après avoir scanné la puce, nous vérifions si chaque attaque est couverte par le niveau RTL et les modèles de fautes au niveau porte. Pour déterminer si une attaque est couverte, nous comparons les destinataires des attaques possibles du niveau RTL et les modèles de fautes au niveau porte avec ceux que l'outil de validation du layout indique. Ainsi, nous utilisons les ensembles de bascules des bénéficiaires de l'attaque liés à chaque carré et nous vérifions si cette combinaison est entièrement contenu (sous-ensemble), dans l'un des ensembles des destinataires de l'attaque au niveau RTL. Ensuite, nous effectuons la même vérification pour les destinataires de l'attaque au niveau porte. Après cette analyse, chaque case est classé en fonction de si elle était couverte (ou non) par le niveau RTL et les ensembles de fautes au niveau porte. On finit par déterminer le pourcentage de la région du layout qui a été couvert par le niveau RTL et l'analyse du niveau porte.

Un exemple de la procédure de validation est également illustré dans la figure 4.3. À gauche, on peut voir une partie d'un circuit abstrait et son partitionnement aux cônes logiques. Dans cet exemple, le cône logique B est l'origine de l'attaque considérée qui génère un ensemble de destinataires de l'attaque contenant la bascule A, la bascule B et des autres bascules. De même façon, dans le milieu de la figure, nous pouvons voir que la même origine de l'attaque au niveau porte génère le même ensemble de destinataires de l'attaque. Le côté droit de la figure montre le layout finale et son partitionnement aux origines de l'attaque au niveau du layout (spots rouges). Comme c'est montré, l'outil de validation lit le contenu des spots du layout et, un par un et pour chacun d'eux, il extrait les portes et / ou les bascules logiques qu'il contient (par exemple, le spot 1).

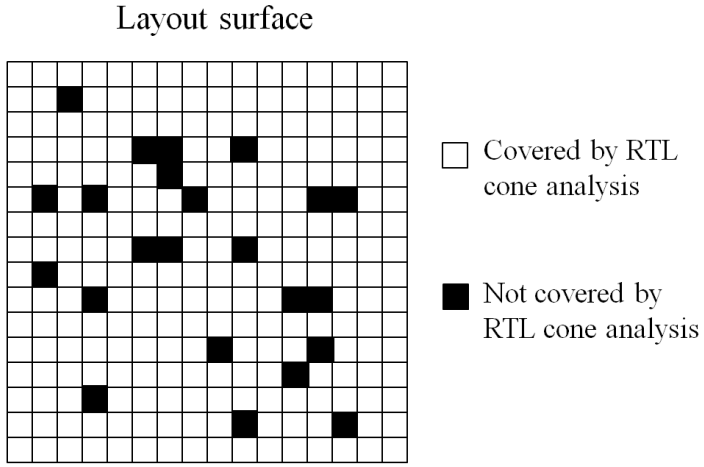
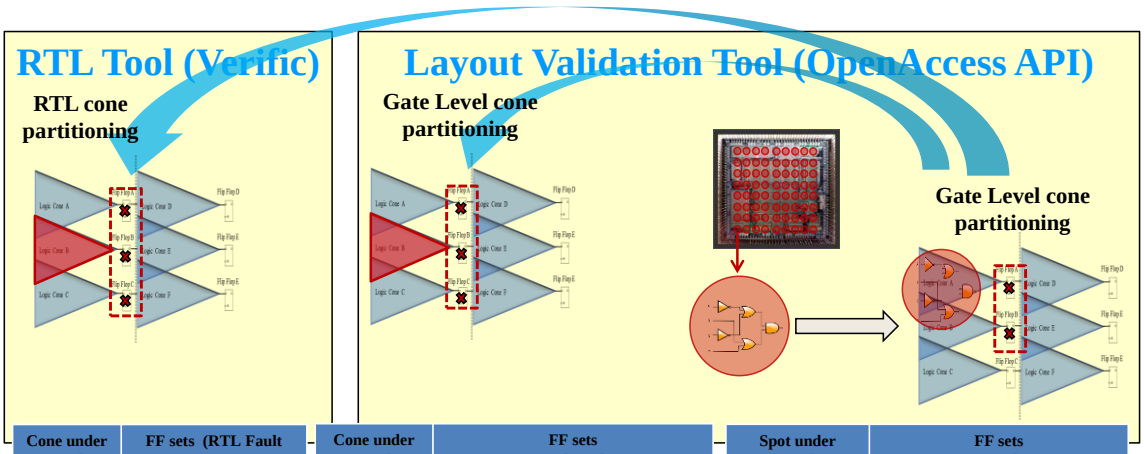


Figure 4.3: Illustration des pourcentages de couverture; dans cet exemple, pour le spot = étape, nous avons le pourcentage de couverture ~= 90%)

Ensuite, l'outil extrait tous les cônes logiques du niveau porte où ces éléments du niveau porte appartiennent. Dans notre exemple, ils sont situés à l'intérieur des cônes logiques A et B. Par conséquent, une attaque au spot 1 dicte, en tant que destinataire de l'attaque, cônes 1 et 2. Comme c'est déjà expliqué précédemment, l'étape prochaine de la validation est de vérifier si l'ensemble de destinataires de l'attaque du spot 1 est un sous-ensemble d'au moins un ensemble de destinataires de l'attaque au niveau porte. Pour cet exemple, l'ensemble contenant les bascules A et B est un sous-ensemble d'au moins deux ensembles de destinataires au niveau porte, c'est à dire les ensembles générés par les cônes A et B. Par conséquent, le spot 1 est couvert par le modèle de fautes du cône du niveau porte. En outre, le spot 1 est, de façon similaire, un sous-ensemble des ensembles de destinataires des cônes A et B au niveau RTL. Ainsi, le spot 1 est couvert par le modèle de fautes du cône soit au niveau RTL ou au niveau porte. Dans cet exemple, un spot du layout est couvert et, en outre, nous pouvons voir que pour cette partie du circuit il n'y a pas de différence entre le niveau RTL et le partitionnement du cônes au niveau porte. Naturellement, il est également possible pour un spot d'être couvert par l'un des modèles de fautes du cône, soit au niveau RTL ou au niveau porte, ou il ne peut pas être couvert par aucun d'eux.



Cone under attack	FF sets (RTL Fault space)		
Cone B	FFA	FFB	FFC
Cone A	FFA	FFB	
Cone C	FFB	FFC	
…	…	…	…

Cone under attack	FF sets (Gate Level Fault space)		
Cone B	FFA	FFB	FFC
Cone A	FFA	FFB	
Cone C	FFB	FFC	
…	…	…	…

Spot under attack	FF sets (Layout Fault space)		
Spot 1	FFA	FFB	
Spot 2	FFA	FFC	
…	…	…	…

Figure 4.4: Procedure de validation

4.1.2 La mise en œuvre et les résultats du processus de validation

Pour la mise en œuvre du processus de validation, le OpenAccess (OA) API C ++ a été utilisé [47]. Initialement, une conception au niveau RTL est synthétisé en utilisant Synopsys Design Compiler, en utilisant la bibliothèque de la technologie de cellules ouvertes Nangate 45nm. La netlist Verilog synthétisé est, ensuite, transmise à Cadence SOC Encounter, pour le placement et le routage. Pour l'évaluation, seulement les contraintes temporelles a été utilisé et aucun floorplan ou placement des contraintes était prévus à l'outil du layout. Les cellules standards combinatoires, qui sont placés sous la région souligné dans la figure 4.1, peut être extraite en utilisant l'OA API. En détail, une fonction C ++ a été mis en place: il instancie une case de sélection, en prenant comme arguments les coordonnées de ses coins, l'inférieur gauche et le supérieur droite, et effectue une question sur le layout pour les cellules standards fermés dans cette case.

Tableau 4.1 contient les résultats de validation pour les implémentations choisies. Pour chaque conception que nous avons choisi, trois tailles différentes de carré, 1 µm, 5 µm et 20 µm, (deuxième colonne) et une étape toujours égale à la moitié de la taille du spot (colonne trois).

Tableau 4.1 Résultats de validation par rapport au layout – les spots couverts – les attaques des bascules couvertes directes – les multiplicités d'espace des fautes

Circuit	Spot (µm)	Pas (µm)	RTL spots couverts (%)	Spots couverts au niveau du layout (%)	Attaques directes couvertes au niveau RTL (%)	Attaques directes couvertes au niveau porte (%)	Multiplicité maximum de l'attaque directe	Multiplicité maximum au niveau RTL	Muliplicité maximum au niveau porte	Multiplicité maximum au niveau du layout
AES HR (2369 FFs)	1	0.5	91.1	95.6	90.1	93.5	4	1152	2113	2065
	5	2.5	79.1	91.7	79.1	89.0	13			2066
	20	10	55.5	82.1	67.4	79.8	81			2087
AES Parity (936 FFs)	1	0.5	76.4	88.9	85.9	88.2	4	202	685	652
	5	2.5	35.7	68.8	60.4	68.6	11			655
	20	10	7.3	34.3	27.8	41.2	54			711
ITC99-B18 (3320 FFs)	1	0.5	86.9	90.3	96.5	99.1	4	561	447	478
	5	2.5	78.1	86.6	89.2	96.2	10			496
	20	10	61.5	71.3	66.9	82.6	55			633

4.1.3 Analyse d'espace des fautes

Nous considérons une combinaison aléatoire des bascules de la conception (de toute multiplicité). Cette combinaison appartient à l'espace des fautes au niveau RTL si elle est un sous-ensemble d'au moins un ensemble de bascules de l'espace des fautes du cône au niveau RTL. D'autre part, la même combinaison des bascules appartiennent à l'espace des fautes au niveau le layout, si elle est un sous-ensemble d'au moins un ensemble de destinataires de l'attaque de l'espace des fautes au niveau du layout. Par conséquent, cette combinaison appartient à l'intersection des deux espaces des fautes si elle est en même temps un sous-ensemble d'au moins un ensemble de destinataire de l'attaque pour tous les deux espaces des fautes. La taille de ces espaces des fautes, même pour les circuits de taille moyenne, ne permet pas de quantifier analytiquement leur intersection. Afin d'obtenir une approximation de cette intersection, nous appliquons la méthodologie statistique [22]. D'abord nous choisissons une multiplicité spécifique (nombre de bascules) et, de manière aléatoire, nous choisissons, aussi, des échantillons provenant d'un ensemble de bascules au niveau RTL, jusqu'à ce que nous atteignions une marge d'erreur spécifique. Ensuite, nous vérifions si chaque échantillon est en même temps un sous-ensemble d'au moins une bascule au niveau du layout. De cette façon, on calcule le pourcentage pour chaque ensemble au niveau RTL et on trouve les moyennes pour chaque multiplicité.

Dans le tableau 3.3, nous incluons cette analyse pour la conception de l'AES à base de parité. La première colonne contient les échantillons de multiplicités. Les colonnes deux et trois contiennent les résultats des analyses statistiques de l'espace de fautes du cône au niveau RTL. Les deux colonnes restantes concernent la même analyse pour le modèle de fautes multi-bits aléatoires. Pour cette analyse, nous avons utilisé une marge d'erreur de 10% pour les deux modèles de fautes et toutes les multiplicités. Pour l'espace des fautes basé sur le modèle du cône au niveau RTL dans le tableau III, nous présentons le chevauchement moyen pour

chaque multiplicité. De l'autre côté, l'espace des fautes multi-bits aléatoires est constitué par un seul ensemble contenant toutes les bascules de la conception.

Table 7.2: Analyse statistique du niveau RTL et le chevauchement de l'espace des fautes au niveau de la mise en page

Parité d'AES				
Modèle de fautes du cône au niveau RTL			Modèle de fautes multi-bits aléatoires	
Multiplicité des fautes	Chevaucement moyen du niveau RTL et du niveau de la mise en page (%)	Erreur maximum (%)	Chevaucement du niveau RTL et du niveau de la mise en page (%)	Erreur maximum (%)
2	95.7	10	57.4	10
3	92.3	10	28.7	10
4	89.3	10	25.8	10
5	87.2	10	17.1	10
10	80.7	10	1.9	10
20	77.4	10	0	10

4.1.4 Conclusion

Dans cette section, nous avons présenté les résultats validant les caractéristiques de localité du modèle de fautes du cône avec le layout final de deux implémentations d'AES utilisant des contremesures différentes. Les résultats montrent que la synthèse, le placement et le routage (en raison des optimisations du placement et du routage) n'influencent pas les hypothèses initiales du modèle de fautes du cône, comme ils ont été présentés au chapitre 2.

4.2 Validation par rapport aux attaques laser expérimentales

La validation est basée sur la comparaison des destinataires de l'attaque entre l'injection de fautes expérimentale par laser et le modèle de fautes au niveau RTL. Les destinataires de l'attaque du modèle de fautes au niveau RTL sont déjà connus au niveau RTL, comme expliqué dans les chapitres précédents. D'autre part, les destinataires expérimentaux de l'attaque en fautes ont été obtenus en utilisant la chaîne de scan de l'AES à base de parité.

Au cours des expériences, un cycle d'horloge est sélectionné (que nous allons aussi appeler «instant d'injection») pour effectuer une attaque au laser, avec une durée d'impulsion d'un cycle d'horloge. Le laser a été paramétré de façon à cibler le cycle d'horloge sélectionné avec une incertitude (jitter) de plus ou moins un cycle d'horloge en raison des capacités de l'équipement utilisé. Puis des attaques multiples ont été réalisées avec les mêmes paramètres d'injection. On a tiré une impulsion chaque fois, en ciblant le même cycle d'horloge X, et on a effectué la lecture du contenu de la chaîne chaque fois au prochain front montant de l'horloge, comme représenté dans le figure 4.4. Après l'achèvement de la campagne d'injection de fautes, nous traitons les résultats en les comparant avec le contenu de la chaîne de scan.

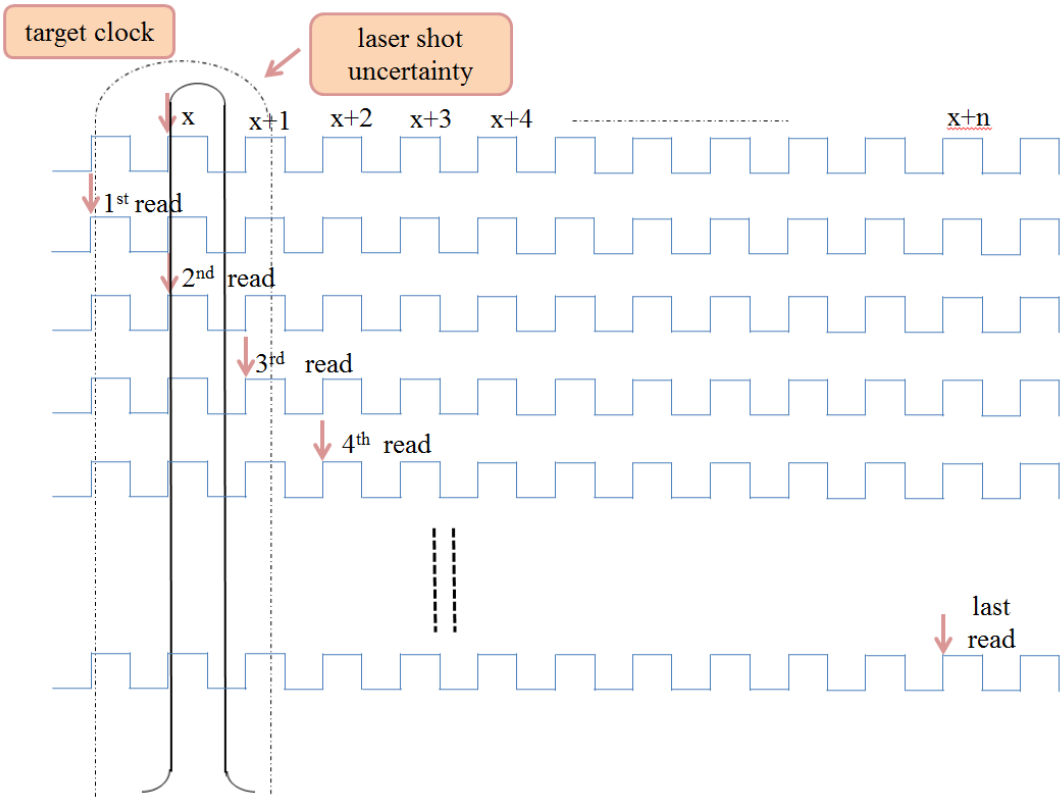


Figure 7.4: Approche de l'injection de fautes par laser pour un ensemble des paramètres du laser

4.2.1 Plateforme expérimentale

Le dispositif expérimental comprend deux circuits imprimés (Printed Circuit Boards-PCB) de manière à aligner la puce avec la source de laser et à l'interfacer avec une carte de développement FPGA. Dans la figure 4.5, nous pouvons voir les deux PCB, en haut la carte mère, responsable de l'alimentation et de l'interfaçage de la prise avec la carte FPGA. Ci-dessous les cartes mères, nous pouvons voir la carte de la prise qui a été utilisé pour maintenir l'ASIC. Les prises ont été nécessaires afin de faciliter le test de multiples puces différentes. Enfin, au fond, nous pouvons voir l'ASIC et son ouverture afin d'être capable de focaliser le faisceau du laser sur sa face arrière.

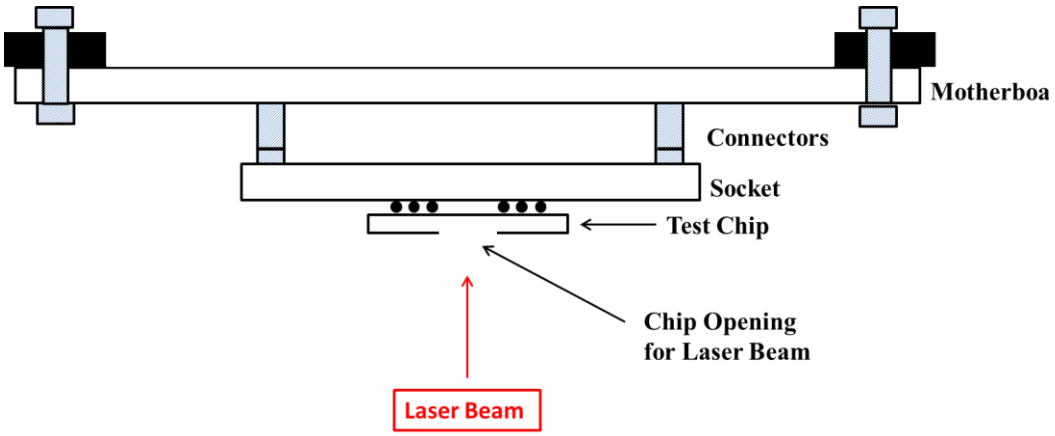


Figure 7.5: Représentation schématique de la pile de la carte mère, de la carte de prise et de la puce pour la campagne expérimentale de laser

4.2.2 Paramètres et résultats expérimentaux

4.2.2.1 Première campagne d'injection de fautes par laser

Pour la première campagne expérimentale d'attaque au laser, les paramètres comprennent un scan de la région complète de la conception de l'AES à base de parité avec une taille de spot du laser de 5 µm et un pas de 4 µm. Ce scan nous a permis d'avoir une analyse complète du circuit, y compris un certain chevauchement entre les spots. Les paramètres utilisés étaient la puissance et la durée d'injection.

Après le traitement des résultats, on obtient la couverture au niveau RTL, compte tenu de toutes les multiplicités des fautes obtenues qui vont de un jusqu'à dix. La couverture de cette campagne expérimentale est égale à 9,3%. Cela signifie que 9,3% de toutes les fautes expérimentales est un sous-ensemble d'au moins un ensemble de destinataires de l'attaque au niveau RTL. Après l'analyse des résultats, nous avons découvert une raison systématique pour cette couverture faible. Dans chaque faute expérimentale avec une multiplicité supérieure à 3, il y a toujours une seule bascule qui ne figure pas dans le même ensemble des destinataires de l'attaque au niveau RTL. En fait, cette bascule appartient toujours à un byte différent et, aussi, à un sous-module différent. Par exemple, quand une attaque influence plusieurs bascules du chemin de données, il y avait une bascule du « key unit » impliquée et vice versa.

Plus tôt dans la section actuelle nous avons mentionné que lors de la conception du layout de l'AES à base de parité, aucune contrainte de placement a été suivi. Cela signifie que les cellules (soit combinatoires ou séquentielles) du « data unit » et du « key unit » de l'AES ont été autorisées à être placées à proximité. D'autre part, l'analyse au niveau RTL a considéré ces deux sous-circuits de la conception en tant que complètement découplé. Par conséquent, il existait des spots du laser pendant les campagnes expérimentales de l'injection de fautes qui étaient en mesure de cibler des cônes logiques non-croisés.

4.2.2.2 Deuxième campagne d'injection de fautes par laser

Cette campagne de l''injection de fautes a impliqué une analyse complète du circuit avec une taille de spot de 5 µm et un taille du pas de 4 µm, la même que la première campagne. Cette fois, l'injection a eu lieu seulement au cycle d'horloge au début de l'avant-dernier tour de l'AES. De plus, plusieurs étapes ont été utilisées pour la puissance de la source du laser.

Les résultats de cette campagne étaient différents lorsque la couverture au niveau RTL était de 52,6% (sans permettre une quelconque incompatibilité). La raison de ceci était le meilleur des couvertures pour les multiplicités de 2 à 5, avec l'exception de la multiplicité de 3 fautes simultanées. Quand nous avons permis au modèle de fautes au niveau RTL cette fois de perdre un seul bit de la faute expérimentale puis nous avons remarqué une couverture de 100%. On n'a pas remarqué une grande dépendance à l'égard de la couverture dans les deux cas (lorsque nous avons permis une incompatibilité ou lorsque nous n'avons pas) sur l'augmentation progressive de la puissance.

Cette campagne montre, donc, que la raison principale pour affecter à la fois les données et le key unit est qu'il n'y a aucune contrainte de placement pour séparer les cellules appartenant au data unit et au key unit. Comme le spot du laser est capable d'attaquer en même temps les deux unités, il est prévu que le modèle de fautes au niveau RTL va échouer à prédire de ce type de fautes. D'autre part, le modèle de fautes au niveau RTL parvient à couvrir le 100% des fautes expérimentaux injectées s'il n'y a pas de fautes impliquant data et key unit en même temps (en supprimant seulement un seul bit de faute, de cette manière).

4.2.3 Conclusions

L'analyse exhaustive des injections des fautes prend énormément de temps d'expérimentation et nous avons vu que le modèle de fautes au niveau RTL couvre :

- 9,3% de toutes les fautes multi-bits au cours de la première campagne
- 52,6% de toutes les fautes multi-bits au cours de la deuxième campagne

Les résultats montrent, en particulier, pour les résultats de la deuxième campagne, une couverture grande. En même temps, au niveau RTL le data unit et le key unit de l'AES sont complètement séparés (pas d'intersection logique des cônes entre les deux), tandis qu'au niveau du layout ils sont autorisés à être placé à proximité. Cette incohérence peut conduire aux scénarios des fautes expérimentales qui ne sont pas, a priori, possibles au niveau RTL, selon le modèle de fautes du cône.

D'autre part, si nous excluons le phénomène quand une combinaison de fautes multiples dans le data unit n'est pas couverte en raison d'un bit erroné de key unit, nous obtenons des couvertures meilleures:

- 91,2% de toutes les fautes multi-bits au cours de la première campagne
- 100% des fautes multi-bits au cours de la deuxième campagne

5. Développement des contremesures contre les attaques au laser

Comme nous l'avons vu dans les chapitres précédents, les attaques en fautes au laser peuvent être utilisées pour injecter dans des circuits intégrés sécurisés, soit des fautes simples ou multiples. Ces attaques peuvent être des moyens efficaces pour produire des fautes exploitables sur le fonctionnement des circuits intégrés sécurisés. Les capacités des attaques au laser sont principalement attribuées à la grande précision qu'ils possèdent et à leur timing précis [5]. Afin d'éviter leur exploitation, les circuits intégrés sécurisés modernes doivent être capables de détecter de telles attaques.

Bien que les contremesures puissent être développées à différents niveaux d'abstraction, le niveau d'abstraction au niveau RTL est très utile car il intervient tôt dans le flot de conception. Un inconvénient inhérent de la conception des contremesures au niveau RTL est l'information manquante en ce qui concerne la synthèse, le placement et le routage. Pour dépasser ces désavantages, dans ce chapitre, nous allons utiliser le modèle de fautes par laser au niveau RTL qui a été défini et validé dans les chapitres précédents. Notre objectif est de montrer comment le modèle de fautes concerné peut aider dans le processus de conception en réduisant les re-spins à partir des niveaux d'abstraction inférieurs aux niveaux plus élevés. En fait, dans ce chapitre, nous voulons montrer que la disposition d'un modèle de fautes par laser au niveau RTL peut limiter les re-spins entre la conception et la validation de la contremesure au sein de l'abstraction au niveau RTL.

Les contremesures vont inévitablement augmenter le coût de la conception et de l'évaluation d'un circuit intégré principalement en raison de coût du matériel nécessaire pour sa protection. En outre, les contremesures utilisent souvent de la redondance du matériel, de la redondance temporelle et de la redondance de l'information [49]. Dans le chapitre, nous allons décrire une approche hybride pour protéger des circuits intégrés sécurisés contre les attaques au laser. Cette approche est basée sur le matériel et la redondance des informations de manière à atteindre des niveaux élevés de protection avec des frais du matériel raisonnables.

L'avantage de la contremesure présentée dans ce chapitre est qu'il peut être appliqué aux circuits sécurisés différents parce qu'il n'utilise pas de propriétés d'une architecture spécifique. Le but de la contremesure est d'atteindre 100% de détection selon le modèle de fautes par attaques au laser au niveau RTL, présenté dans cette thèse. En outre, l'objectif est de fournir une contremesure générique, simple et efficace au niveau RTL qui utilise une approche hybride de redondance avec un coût supplémentaire bas.

Ce chapitre présent deux modèles de fautes sur lesquels l'analyse de ce travail est basé. La mise en œuvre de notre contremesure est décrite aussi. Ensuite, nous décrivons l'étude pour un circuit cryptographique d'AES. De plus, les résultats sont présentés concernant le coût, les taux d'erreur de l'injection de fautes et les capacités de détection, par rapport au layout, réalisé par le circuit d'AES protégé.

5.1 Contremesures basé sur la redondance de l'information et du matériel

Les contremesures qui peuvent être développées au niveau RTL sont des contremesures basées sur la redondance. Ces contremesures peuvent être classées en plusieurs types : redondance matérielle, redondance temporelle et redondance de l'information. Dans la littérature, il existe des contremesures qui sont mises en œuvre au niveau RTL pour la protection contre les attaques en fautes, mais elles ne sont pas spécialisés pour les attaques en fautes par laser.

5.2 Contremesures contre des attaques en fautes par laser

L'architecture de la mise en œuvre de la contremesure est basée sur une approche hybride mélangeant redondance matérielle et redondance de l'information. Plus précisément, les données qui doivent être protégées sont séparées en groupes et un bit de parité est utilisée pour chaque groupe. Afin de maximiser l'efficacité d'une telle contremesure, au plus une faute doit être simultanément injectée dans chaque groupe. Afin d'avoir au plus une faute sur chaque groupe de parité, nous utilisons des résultats de l'analyse de la conception selon les types de fautes simultanées prévus par le modèle de fautes de l'attaque au laser au niveau RTL.

Dans la figure 5.1, nous pouvons voir le principe de la génération de la parité avec la prédiction [57]. Un code de parité simple est utilisé comme redondance de l'information, et la reproduction des blocs fonctionnels pour la prédiction de la parité est utilisée comme redondance du matériel. Cette contremesure vise à protéger à la fois les instances combinatoires et séquentielles de la conception. La contremesure proposée est basée sur trois étapes, la première étape est le calcul de la parité, la deuxième étape est la prédiction de la parité calculée avec les composants dupliqués, et la dernière étape est la comparaison de ces parités. Une incompatibilité dans la parité calculée et prédite signifie qu'une erreur est survenue dans les instances soient combinatoires ou séquentielles. Le comparateur indique si une erreur est survenue en délivrant en sortie un signal d'erreur.

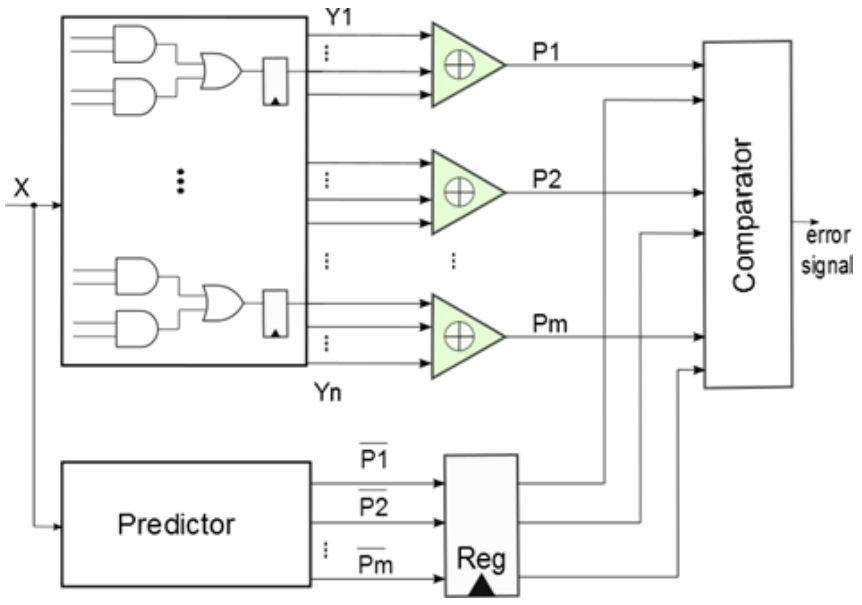


Figure 5.1: Architecture génerale de la contremesure: (1) Pi parité calculée par les bascules de la conception originale (2) Pi-dashed parité predite par les données à l'entrée x (3) comparaison entre la parité Pi et la parité Pi -dashed

5.3 Etude de cas

Dans cette section, nous décrivons la procédure pour mettre en œuvre notre contremesure sur un circuit sécurisé. La mise en œuvre est réalisée sur l'AES, qui est décrit dans [59], utilisée sans ses contremesures. En particulier, ce circuit présente l'avantage d'une implémentation régulière en divisant le bloc de 128 bits par défaut en 16 blocs de 8 bits. LE circuit AES se compose de trois éléments principaux, qui sont: Data Unit (DU) (le chemin de données de chiffrement), Key Unit (KU) (cycle de génération de clés) et Control Unit (CU). Dans notre étude de cas, nous avons appliqué la contremesure proposée à le DU et les composants de KU. Le composant DU composé de 16 blocs et chaque bloc opère sur un vecteur de 8 bits. Le composant KU est responsable de la clé secrète et travaille sur un vecteur de 128 bits qui est la taille de la clé (également des clés de 192 bits et 256 bits sont autorisés).

Une analyse du cône avec l'outil au niveau RTL, qui implémente les modèles de fautes de l'attaque au laser au niveau RTL, indique les combinaisons possibles de bits qui doivent être inclus dans les groupes de la parité.

La contremesure principale développée comprend la protection totale du DU et KU. Ainsi, afin d'évaluer l'efficacité de notre contremesure et de souligner les opportunités, un second type de contremesure a été mis au point. Ce second type de contremesure est considéré comme une approche classique et vise à protéger les groupes de 8 bits provenant du même registre sans prendre en compte les dépendances des cônes logiques. Les contremesures mises en œuvre sont donc :

- Protections du DU et du KU basées sur le modèle de fautes de l'attaque au laser au niveau RTL avec des groupes de la parité faits par des cônes indépendants (FM)
- Protections du DU et du KU, avec des groupes de la parité faits par des cônes dépendants pas en utilisant le modèle de fautes de l'attaque au laser au niveau RTL (NFM)

5.4 Analyse des contremesures

Tout d'abord, nous évaluons les contremesures en termes de coût, par rapport à la conception originale de l'AES. Les contremesures ont été synthétisées avec la technologie ouverte « NANGATES 45nm » en utilisant Synopsys Design Compiler. La première stratégie de la synthèse utilise les options « simple-compilation » (SC), tandis que la deuxième utilise les options « ultra-compilation » (UC). Les résultats de la synthèse sont montrés dans le tableau 5.1. Ce tableau montre que l'option « simple-compilation » de la contremesure FM a approximativement le même coût que la contremesure NFM. De l'autre côté, avec l'option – « ultra-compilation », une différence dans le coût est observée. En fait, la contremesure NFM est mieux optimisé que la contremesure FM d'environ 12%.

Tableau 7.3: Pourcentages de couverture sur chaque multiplicité des fautes

Conception	Synthèse	
	Espace (µm2)	Coût (%)
FM (SC)	28133	79.9
NFM (SC)	27870	78.3
FM (UC)	21191	63
NFM (UC)	19635	51

Table 7.4: Taux d'erreur d'injection de fautes pour le modèle de fautes de l'attaque laser au niveau RTL pour une marge d'erreur au plus de 10%.

Multiplicité des fautes		M2	M3	M4	M5	M6	M7	M8	M9	M10
FM	Detectées	18.4	24	28.3	31.6	34.1	36.7	38.2	39.9	40.6
	Non-détectées	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	Silencieux	1.5	0.5	0.3	0.4	0.3	0.3	0.1	0	0
	Faux positive	79.9	75.4	71.3	67.9	65.5	62.9	61.6	60	59.3
	Taux final de detection	98.3	99.4	99.6	99.5	99.6	99.6	99.8	99.9	99.9
NFM	Detectées	19.9	28.2	30	35.1	35.2	39.1	38.3	41.8	40.8
	Non-détectées	2.9	0.2	2	0.1	1.7	0.1	1.3	0.1	1.4
	Silencieux	9.2	1.1	4.9	0.4	4.4	0.3	4.5	0.8	4.2
	Faux positive	68	70.5	63.1	64.4	58.7	60.5	55.9	57.3	53.7
	Taux final de detection	87.9	98.7	93.1	99.5	93.9	99.6	94.2	99.1	94.5

Tableau 7.5: Pourcentages de fautes de détection pour le modèle de fautes au niveau layout

Conception	Spot 1µm		Spot 5µm	
	MEU	MEU & MET	MEU	MEU & MET
FM (SC)	99.6	99.1	98.8	98
NFM (SC)	86.6	66.6	68.6	72.2
FM (UC)	98.3	88	94.7	93.4
NFM (UC)	84.4	57.3	68.8	73

5.5 Conclusion

Ce chapitre présente une contremesure générique pour la protection des circuits intégrés, spécialement développée pour protéger à la fois les instances combinatoires et séquentielles. Cette contremesure est basée sur le modèle de fautes de l'attaque laser au niveau RTL. La nouveauté de la contremesure est de fournir une approche sur la façon de créer les groupes de parité pour augmenter les taux de détectabilité contre les attaques laser. L'évaluation comprend l'injection de fautes au niveau RTL à travers le modèle de fautes du cône et le modèle de fautes aléatoires. En outre, une évaluation basée sur le layout est utilisée pour renforcer les

résultats. Les résultats montrent que la contremesure présentée peut obtenir de meilleurs taux de détection de fautes qu'une contremesure similaire avec approximativement le même espace du coût.

6. Conclusion

Dans cette thèse, nous avons présenté un nouveau modèle de fautes (modèle de fautes du cône), défini au niveau RTL, afin de prédire les effets des attaques laser sur les circuits intégrés, tôt dans le flot de conception. En outre, le modèle de fautes et la méthodologie a été appliqué(e)s à différentes architectures AES au niveau RTL impliquant des contremesures différentes. Le modèle de fautes a permis de générer les listes des fautes à injecter et les campagnes d'injection de fautes ont été effectuées à travers l'émulateur basé sur la reconfiguration partielle de FPGA du laboratoire TIMA. Des injections des fautes ont été réalisées sur les mêmes circuits en utilisant l'approche par injection de fautes aléatoire (modèle de fautes aléatoires).

Les résultats de ces deux évaluations ont été comparés afin d'évaluer les circuits, mais aussi de manière à comparer les deux modèles de fautes. Les résultats ont montré que le modèle de fautes du cône est capable d'injecter des fautes qui étaient qualitativement différentes de celles du modèle de fautes aléatoires. La caractérisation de ces modèles a montré que le modèle de fautes du cône est capable d'injecter des fautes qui ont des meilleures caractéristiques de localité. Pour le même ordre de multiplicité, par rapport à l'approche aléatoire, le modèle du cône limite la dispersion des fautes à l'intérieur des circuits. Par conséquent, il a également réussi à déterminer les combinaisons des fautes qui conduisent aux fautes non-détectées même pour des multiplicités grandes (de 2 à 8 fautes simultanément injectés) pour la conception de l'AES à base de parité. D'autre part, le modèle de fautes aléatoires a été incapable de souligner l'incapacité de la conception à détecter aussi bien les multiplicités paires et impaires. Ces résultats soulignent que le choix du modèle de fautes peut avoir un impact sur l'évaluation et que si le modèle de fautes utilisé n'est pas assez précis, il peut conduire à une fausse assurance de la sécurité.

Même si il est clair que le modèle de fautes du cône pour mieux prédire les attaques que l'approche aléatoire au cours des campagnes d'injection de fautes, il est important d'utiliser aussi les autres niveaux d'abstraction bas pour valider le modèle de fautes. Pour cette raison, nous avons également présenté une méthode de validation qui renforce notre confiance sur les capacités du modèle de fautes du cône. La méthodologie caractérise des spots géométriques au niveau du layout comme couvert ou non par le modèle de fautes du cône et le modèle de fautes aléatoires. Il est montré que, pour les spots de 1 et 5 µm, la plupart des spots au niveau du layout sont couverts par le modèle de fautes du cône. Cette analyse prend également en compte les étapes de la synthèse du flot de conception et il montre que les caractéristiques de la localité au niveau RTL, que le modèle de fautes du cône prédit, sont valables même après les optimisations de la synthèse, du placement et du routage. De plus, une analyse des espaces des fautes pertinents montre que l'espace des fautes du modèle de fautes du cône et la validation de la méthode au niveau du layout se chevauchent beaucoup, même pour des multiplicités grandes des fautes injectées. De l'autre côté, l'espace des fautes aléatoires et l'espace des fautes au niveau du layout ne se chevauchent pas aussi bien. En même temps, l'espace des fautes du cône au niveau RTL et l'espace de fautes au niveau du layout sont beaucoup plus petits que l'espace des fautes aléatoires (de toutes les combinaisons possibles des fautes). En effet, la majorité des combinaisons des fautes de l'espace des fautes aléatoires sont des fautes

qui ne sont pas possibles par rapport au layout, pour les attaques localisées. Par conséquent, un résultat possible à compter sur le modèle de fautes aléatoires lors d'une évaluation est de décider de contraindre trop les contremesures qui protègent la conception. Cela peut à son tour conduire à une durée plus grande de la conception et à un espace plus grand qui augmentent le coût et le temps d'accès au marché de la mise en œuvre.

Grâce au projet LIESSE et en collaboration avec CMP (Centre Microélectronique de Provence), nous avons pu valider la capacité du modèle de fautes du cône à prédire des fautes qui sont possibles au cours d'une campagne expérimentale d'injection de fautes par laser. La campagne d'injection de fautes a été réalisée en utilisant la mise en œuvre de la technologie de STMictoelectronics de 28nm pour la conception de l'AES à base de parité. Les résultats ont montré que le modèle de fautes du cône peut prévoir un grand pourcentage des fautes expérimentalement injectées.

Le modèle de fautes du cône validé a conduit au développement d'une nouvelle contremesure contre les attaques laser. A part les attaques laser, il peut également être utilisé pour protéger les circuits intégrés contre toute forme d'attaques en fautes avec un impact local sur le circuit. La contremesure a une capacité théorique de détection de 100% dans la détection des fautes qui sont prédites par le modèle de fautes du cône. Les campagnes d'injection de fautes et l'analyse avec des informations au niveau du layout montrent que les capacités réelles de détection de la contremesure sont très proches des attentes théoriques. L'avantage de cette contremesure est le coût bas, ainsi que le fait qu'elle peut être appliquée facilement à toute application au niveau RTL.

Les perspectives de ce travail comprennent des modifications du modèle de fautes du cône afin d'inclure plus d'un des cônes logiques comme l'origine de l'attaque.

En ce qui concerne la nouvelle contremesure, l'automatisation de l'addition de la contremesure est une perspective intéressante. De plus, d'autres travaux peuvent comprendre l'application de la contremesure sur une variété de conceptions, avec FPGAs, mais aussi comme un circuit intégré. Ensuite, l'injection de fautes par laser sur des conceptions protégées peut montrer la capacité expérimentale de la contremesure de protéger contre les attaques au laser.

8 Publication List

Journals

[P1] Athanasios Papadimitriou, David Hély, Vincent Beroulle, Paolo Maistri, Regis Leveugle, "Analysis of laser-induced errors: RTL fault models versus layout locality characteristics", ELSEVIER Microprocessors and Microsystems (MICPRO) Journal, Special Issue on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE), DOI: 10.1016/j.micpro.2016.01.018, January 18 2016.

[P2] X. Guo, C. Jin, C. Zhang, A. Papadimitriou, D. Hély, R. Karri, "Can Algorithm Diversity in Stream Cipher Implementation Thwart (Natural and) Malicious Faults?", IEEE Transactions on Emerging Topics in Computing (Volume: PP , Issue: 99), DOI: 10.1109/TETC.2015.2434103, May 20 2015.

International Conferences & Workshops

[P3] A. Papadimitriou, D. Hély, V. Beroulle, P. Maistri, R. Leveugle, "A multiple fault injection methodology based on cone partitioning towards RTL modeling of laser attacks", Design, Automation and Test in Europe Conference and Exhibition (DATE), IEEE, DOI: 10.7873/DATE.2014.219, 24-28 March 2014.

[P4] P. Vanhauwaert, P. Maistri, R. Leveugle, A. Papadimitriou, D. Hély, V. Beroulle, "On error models for RTL security evaluations" , Design & Technology of Integrated Systems In Nanoscale Era (DTIS), 2014 9th IEEE International Conference On, IEEE, DOI: 10.1109/DTIS.2014.6850666, 6-8 May 2014.

[P5] R. Leveugle, P. Maistri, P. Vanhauwaert, F. Lu, G. Di Natale, M. L. Flottes, B. Rouzeyre, A. Papadimitriou, D. Hély, V. Beroulle, G. Hubert, S. De Castro, J. M. Dutertre, A. Sarafianos, N. Boher, M. Lisart, J. Damiens, P. Candelier, C. Tavernier, "Laser-induced fault effects in security-dedicated circuits", Very Large Scale Integration (VLSI-SoC), 2014 22nd International Conference on, IEEE, DOI: 10.1109/VLSI-SoC.2014.7004184, 6-8 Oct. 2014.

[P6] A. Papadimitriou, M. Tampas, D. Hély, V. Beroulle, P. Maistri, R. Leveugle, "Validation of RTL laser fault injection model with respect to layout information", Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on, IEEE, DOI: 10.1109/HST.2015.7140241, 5-7 May 2015.

[P7] Charalampos Ananiadis, Athanasios Papadimitriou, David Hély, Vincent Beroulle, Regis Leveugle, Paolo Maistri, "On The Development of a new countermeasure based on a Laser Attack RTL Fault Model", Design, Automation and Test in Europe Conference and Exhibition (DATE), IEEE, DOI: not yet assigned , 14-18 March 2016.

Book Chapters

[P8] Vincent Beroulle, Philippe Candelier, Stephan De Castro, Giorgio Di Natale, Jean-Max Dutertre, Marie-Lise Flottes , David Hély, Guillaume Hubert, Regis Leveugle, Feng Lu, Paolo Maistri, Athanasios Papadimitriou, Bruno Rouzeyre, Clement Taver-

nier, Pierre Vanhauwaert, " Laser-Induced Fault Effects in Security-Dedicated Circuits", VLSI-SoC: Internet of Things Foundations, Volume 464 of the series IFIP Advances in Information and Communication Technology pp 220-240, DOI: 10.1007/978-3-319-25279-7_12, November 25 2015.

Conferences and workshops without formal proceedings

[P9] Athanasios Papadimitriou, David Hély, Vincent Beroulle, Paolo Maistri, Regis Leveugle, "Analysis of laser-induced errors: RTL fault model versus layout locality characteristics", TRUDEVICE 2015: Workshop on Trustworthy Manufacturing and Utilization of Secure Devices, Grenoble, March 13 2015.

[P10] Athanasios Papadimitriou, David Hély, Vincent Beroulle, Paolo Maistri, Regis Leveugle, "FPGA Emulation of Laser Attacks Against Secure Deep Submicron Integrated Circuits", TRUDEVICE Workshop on Test and Security, Paderborn, 29-30 May, 2014.

[P11] Athanasios Papadimitriou, David Hély, Vincent Beroulle, Paolo Maistri, Regis Leveugle, "FPGA Emulation of Laser Attacks Against Secure Deep Submicron Integrated Circuits", GDR-SOC-SIP, Lyon, France 2013.

9 References

[1] Boneh, D., DeMillo, R. A., & Lipton, R. J. (2001). On the importance of eliminating errors in cryptographic computations. Journal of cryptology, 14(2), 101-119.

[2] Giraud, C., & Thiebeauld, H. (2004). A survey on fault attacks. In Smart Card Research and Advanced Applications VI (pp. 159-176). Springer US.

[3] Schmidt, J. M., & Hutter, M. (2007). Optical and EM fault-attacks on crt-based rsa: Concrete results.

[4] Skorobogatov, S. P., & Anderson, R. J. (2002). Optical fault induction attacks. In Cryptographic Hardware and Embedded Systems-CHES 2002 (pp. 2-12). Springer, Berlin, Heidelberg.

[5] J. G. van Woudenberg, M. F. Witteman, and F. Menarini, "Practical optical fault injection on secure microcontrollers," in Fault Diagnosis and Tolerance in Cryptography (FDTC), 2011 Workshop on, 2011, pp. 91–99.

[6] C. Roscian, A. Sarafianos, J.-M. Dutertre, A. Tria, "Fault Model Analysis of Laser-Induced Faults in SRAM Memory Cells", Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013, pp. 89-98.

[7] C. Roscian, J.-M. Dutertre, A. Tria, "Frontside laser fault injection on cryptosystems - Application to the AES' last round", International Symposium on Hardware-Oriented Security and Trust (HOST), 2013, pp. 119-124.

[8] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," IEEE Transactions on Nuclear Science, vol. 50, no. 3, pp. 583–602, Jun. 2003.

[9] Miskov-Zivanov, Natasa, and Diana Marculescu. "Multiple transient faults in combinational and sequential circuits: a systematic approach." Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 29.10 (2010): 1614-1627.

[10] Pagliarini, Samuel N., and Dhiraj Pradhan. "A placement strategy for reducing the effects of multiple faults in digital circuits." On-Line Testing Symposium (IOLTS), 2014 IEEE 20th International. IEEE, 2014.

[11] Habing, D. H. (1965). "The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits", Nuclear Science, IEEE Transactions on, 12(5), 91-100.

[12] Buchner, S., Baze, M., Brown, D., McMorrow, D., & Melinge, J. (1997). Comparison of error rates in combinational and sequential logic. Nuclear Science, IEEE Transactions on, 44(6), 2209-2216.

[13] Mahatme, N. N., Jagannathan, S., Loveless, T. D., Massengill, L. W., Bhuva, B. L., Wen, S. J., & Wong, R. (2011). Comparison of combinational and sequential error rates for a deep submicron process. Nuclear Science, IEEE Transactions on, 58(6), 2719-2725.

[14] Artola, L., Hubert, G., & Rousselin, T. (2014). Single-event latchup modeling based on coupled physical and electrical transient simulations in CMOS technology. Nuclear Science, IEEE Transactions on, 61(6), 3543-3549.

[15] Lu, F., Di Natale, G., Flottes, M. L., Rouzeyre, B., & Hubert, G. (2014, May). Layout-aware laser fault injection simulation and modeling: From physical level to gate level. In Design & Technology of Integrated Systems In Nanoscale Era (DTIS), 2014 9th IEEE International Conference On (pp. 1-6). IEEE.

[16] Fibich, C., Rössler, P., Tauner, S., Taucher, H., & Matschnig, M. (2015). A netlist-level fault-injection tool for FPGAs. e & i Elektrotechnik und Informationstechnik, 132(6), 274-281.

[17] Maxwell, P. C., & Aitken, R. C. (1993, October). Biased voting: a method for simulating CMOS bridging faults in the presence of variable gate logic thresholds. In Test Conference, 1993. Proceedings., International (pp. 63-72). IEEE.

[18] R. Leveugle,"Early Analysis of Fault-based Attack Effects in Secure Circuits," IEEE Transactions on Computers, vol. 56, no. 10, pp. 1431–1434, Oct. 2007.

[19] Ebrahimi, Mojtaba, Hossein Asadi, and Mehdi B. Tahoori. "A layout-based approach for multiple event transient analysis." Proceedings of the 50th Annual Design Automation Conference. ACM, 2013.

[20] Favalli, M. (2004, October). Annotated bit flip fault model. In Defect and Fault Tolerance in VLSI Systems, 2004. DFT 2004. Proceedings. 19th IEEE International Symposium on (pp. 366-376). IEEE.

[21] Riesgo, T., & Uceda, J. (1996, September). A fault model for VHDL descriptions at the register transfer level. In Design Automation Conference, 1996, with EURO-VHDL'96 and Exhibition, Proceedings EURO-DAC'96, European (pp. 462-467). IEEE.

[22] Leveugle, R., Calvez, A., Maistri, P., & Vanhauwaert, P. (2009, April). Statistical fault injection: quantified error and confidence. In Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09. (pp. 502-506). IEEE.

[23] Lu, F., Di Natale, G., Flottes, M. L., & Rouzeyre, B. (2013, September). Laser-induced fault simulation. In Digital System Design (DSD), 2013 Euromicro Conference on (pp. 609-614). IEEE.

[24] Bosio, A., & Natale, G. D. (2008, November). LIFTING: A flexible open-source fault simulator. In Asian Test Symposium, 2008. ATS'08. 17th (pp. 35-40). IEEE.

[25] Baraza, J. C., Gracia, J., Gil, D., & Gil, P. J. (2005, December). Improvement of fault injection techniques based on VHDL code modification. In High-Level Design Validation and Test Workshop, 2005. Tenth IEEE International (pp. 19-26). IEEE.

[26] J. Grinschgl, A. Krieg, C. Steger, R. Weiss, H. Bock, and J. Haid, "Automatic saboteur placement for emulation-based multi-bit fault injection," in Reconfigurable

Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on, 2011, pp. 1–8.

[27] Antoni, L., Leveugle, R., & Fehér, B. (2003). Using run-time reconfiguration for fault injection applications. Instrumentation and Measurement, IEEE Transactions on, 52(5), 1468-1473.

[28] A. Janning, J. Heyszl, F. Stumpf, and G. Sigl, "A Cost-Effective FPGA-based Fault Simulation Environment," Workshop on Fault Di-agnosis and Tolerance in Cryptography 2011, pp. 21–31.

[29] R. Leveugle,"Early Analysis of Fault-based Attack Effects in Secure Circuits," IEEE Transactions on Computers, vol. 56, no. 10, pp. 1431–1434, Oct. 2007.

[30] Otto, M. (2005). Fault attacks and countermeasures (Doctoral dissertation, University of Paderborn).

[31] Jenn, E., Arlat, J., Rimen, M., Ohlsson, J., & Karlsson, J. (1994, June). Fault injection into VHDL models: the MEFISTO tool. In Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on (pp. 66-75). IEEE

[32] IEEE Standard 1076.6 for VHDL Register Transfer Level (RTL) Synthesis

[33] Hachtel, G. D., & Somenzi, F. (2006). Logic synthesis and verification algorithms. Springer Science & Business Media

[34] S. Venkataraman and S. B. Drummonds, "Poirot: Applications of a logic fault diagnosis tool," Design & Test of Computers, IEEE, vol. 18, no. 1, pp. 19–30, 2001.

[35] Wang, L. T., Wu, C. W., & Wen, X. (2006). VLSI test principles and architectures: design for testability. Academic Press. ISBN-13: 978-0123705976

[36] www.verific.com

[37] Even, S. (2011). Graph algorithms. Cambridge University Press, 2nd edition. ISBN: 978-0-521-73653-4

[38] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error analysis and detection procedures for a hardware implementation of the advanced encryption standard," Computers, IEEE Transactions on, vol. 52, no. 4, pp. 492–505, 2003.

[39] http://www.cad.polito.it/downloads/tools/itc99.html

[40] J. Gracia, J. C. Baraza, D. Gil, P. J. Gil, "Comparison and application of different VHDL-based fault injection techniques", IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, 2001, pp. 233-241.

[41] L. Berrojo, I. Gonzalez, F. Corno, M. Sonza-Reorda, G. Squillero, L. Entrena, C. Lopez, "New techniques for speeding up fault-injection campaigns", Design, Automation and Test in Europe Conference (DATE), March 4-8, 2002, pp. 847-852

[42] R. Leveugle, "Fault injection in VHDL descriptions and emulation", IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, 2000, pp. 414-419

[43] P. Vanhauwaert, R. Leveugle, P. Roche, "A flexible SoPC-based fault injection environment", 9th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS), Prague, Czech Republic, April 18-21, 2006, pp. 192-197

[44] Bertoni, G., Breveglieri, L., Koren, I., Maistri, P., & Piuri, V. (2002). On the propagation of faults and their detection in a hardware implementation of the advanced encryption standard. In Application-Specific Systems, Architectures and Processors, 2002. Proceedings. The IEEE International Conference on (pp. 303-312). IEEE.

[45] Bertoni, G., Breveglieri, L., Koren, I., Maistri, P., & Piuri, V. (2003). Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. Computers, IEEE Transactions on, 52(4), 492-505.

[46] Maistri, P., Tiran, S., Maurine, P., Koren, I., & Leveugle, R. (2013, December). Countermeasures against EM analysis for a secured FPGA-based AES implementation. In Reconfigurable Computing and FPGAs (ReConFig), 2013 International Conference on (pp. 1-6). IEEE.

[47] www.si2.org

[48] http://www.cad.polito.it/downloads/tools/itc99.html

[49] Barenghi, A., Breveglieri, L., Koren, I., & Naccache, D. (2012). Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. Proceedings of the IEEE, 100(11), 3056-3076.

[50] Di Natale, G., Doulcier, M., Flottes, M. L., & Rouzeyre, B. (2009). A reliable architecture for parallel implementations of the advanced encryption standard. Journal of Electronic Testing, 25(4-5), 269-278.

[51] Rajendran, J., Borad, H., Mantravadi, S., & Karri, R. (2010, June). SLICED: Slide-based concurrent error detection technique for symmetric block ciphers. In Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on (pp. 70-75). IEEE.

[52] Yen, C. H., & Wu, B. F. (2006). Simple error detection methods for hardware implementation of advanced encryption standard. Computers, IEEE Transactions on, 55(6), 720-731.

[53] Karpovsky, M., Kulikowski, K. J., & Taubin, A. (2004, June). Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In Dependable Systems and Networks, 2004 International Conference on (pp. 93-101). IEEE.

[54] Karpovsky, M. G., Kulikowski, K. J., & Wang, Z. (2007). Robust error detection in communication and computational channels. In Spectral Methods and Multirate Signal Processing. SMMSP'2007. 2007 International Workshop on.

[55] Tomashevich, V., Srinivasan, S., Foerg, F., & Polian, I. (2012, June). Cross-level protection of circuits against faults and malicious attacks. In On-Line Testing Symposium (IOLTS), 2012 IEEE 18th International (pp. 150-155). IEEE.

[56] Touba, N., & McCluskey, E. J. (1997). Logic synthesis of multilevel circuits with concurrent error detection. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 16(7), 783-789.

[57] Sogomonyan, E. S., & Goessel, M. (1993). Design of self-testing and on-line fault detection combinational circuits with weakly independent outputs. Journal of Electronic Testing, 4(3), 267-281.

[58] Pistoulet, P. (2008). U.S. Patent No. 7,428,694. Washington, DC: U.S. Patent and Trademark Office.

[59] Bertoni, G., Breveglieri, L., Koren, I., & Maistri, P. (2004, October). An efficient hardware-based fault diagnosis scheme for AES: performances and cost. InDefect and Fault Tolerance in VLSI Systems, 2004. DFT 2004. Proceedings. 19th IEEE International Symposium on (pp. 130-138). IEEE.

Summary in English:

Many aspects of our current life rely on the exchange of data through electronic media. Powerful encryption algorithms guarantee the security, privacy and authentication of these exchanges. Nevertheless, those algorithms are implemented in electronic devices that may be the target of attacks despite their proven robustness. Several means of attacking integrated circuits are reported in the literature (for instance analysis of the correlation between the processed data and power consumption). Among them, laser illumination of the device has been reported to be one important and effective mean to perform attacks. The principle is to illuminate the circuit by mean of a laser and then to induce an erroneous behavior.

For instance, in so-called Differential Fault Analysis (DFA), an attacker can deduce the secret key used in the crypto-algorithms by comparing the faulty result and the correct one. Other types of attacks exist, also based on fault injection but not requiring a differential analysis; the safe error attacks or clocks attacks are such examples.

The main goal of the PhD thesis was to provide efficient CAD tools to secure circuit designers in order to evaluate counter-measures against such laser attacks early in the design process. This thesis has been driven by two Grenoble INP laboratories: LCIS and TIMA. The work has been carried out in the frame of the collaborative ANR project LIESSE involving several other partners, including STMicroelectronics.

A RT level model of laser effects has been developed, capable of emulating laser attacks. The fault model was used in order to evaluate several different secure cryptographic implementations through FPGA emulated fault injection campaigns. The injection campaigns were performed in collaboration with TIMA laboratory and they allowed comparing the results with other state of the art fault models. Furthermore, the approach was validated versus the layout of several circuits. The layout based validation allowed to quantify the effectiveness of the fault model to predict localized faults. Additionally, in collaboration with CMP (Centre Microélectronique de Provence) experimental laser fault injections has been performed on a state of the art STMicroelectronics IC and the results has been used for further validation of the fault model. Finally the validated fault model leads to the development of an RTL (Register Transfer Level) countermeasure against laser attacks. The countermeasure was implemented and evaluated by fault injection campaigns, according to the developed fault model, other state of the art fault models and versus layout information.

Résumé en Français:

De nombreux aspects de notre vie courante reposent sur l'échange de données grâce à des systèmes de communication électroniques. Des algorithmes de chiffrement puissants garantissent alors la sécurité, la confidentialité et l'authentification de ces échanges. Néanmoins, ces algorithmes sont implémentés dans des équipements qui peuvent être la cible d'attaques. Plusieurs attaques visant les circuits intégrés sont rapportées dans la littérature. Parmi celles-ci, les attaques laser ont été rapportées comme étant très efficace. Le principe consiste alors à illuminer le circuit au moyen d'un faisceau laser afin d'induire un comportement erroné et par analyse différentielle (DFA) afin de déduire des informations secrètes.

L'objectif principal de cette thèse est de fournir des outils de CAO efficaces permettant de sécuriser les circuits en évaluant les contre-mesures proposées contre les attaques laser et cela très tôt dans le flot de conception.

Cette thèse est effectuée dans le cadre d'une collaboration étroite entre deux laboratoires de Grenoble INP : le LCIS et le TIMA. Ce travail est également réalisé dans le cadre du projet ANR LIESSE impliquant plusieurs autres partenaires, dont notamment STMicroelectronics.

Un modèle de faute au niveau RTL a été développé afin d'émuler des attaques laser. Ce modèle de faute a été utilisé pour évaluer différentes architectures cryptographiques sécurisées grâce à des campagnes d'injection de faute émulées sur FPGA.

Ces campagnes d'injection ont été réalisées en collaboration avec le laboratoire TIMA et elles ont permis de comparer les résultats obtenus avec d'autres modèles de faute. De plus, l'approche a été validée en utilisant une description au niveau layout de plusieurs circuits. Cette validation a permis de quantifier l'efficacité du modèle de faute pour prévoir des fautes localisées. De plus, en collaboration avec le CMP (Centre de Microélectronique de Provence) des injections de faute laser expérimentales ont été réalisées sur des circuits intégrés récents de STMICROELECTRONICS et les résultats ont été utilisés pour valider le modèle de faute RTL.

Finalement, ce modèle de faute RTL mène au développement d'une contremesure RTL contre les attaques laser. Cette contre-mesure a été mise en œuvre et évaluée par des campagnes de simulation de fautes avec le modèle de faute RTL et d'autres modèles de faute classiques.