National Technical University of Athens

School of Electrical and Computer Engineering

DIVISION OF INFORMATICS AND COMPUTER TECHNOLOGY

Development of a Layout-based Tool for the evaluation of an RTL Laser Fault Model

DIPLOMA THESIS

MARIOS TAMPAS

Supervisor: Kiamal Pekmestzi

Professor of NTUA

Athens, October 2014

The thesis took place at the LCIS laboratory of Grenoble INP

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Development of a Layout-based Tool for the evaluation of an RTL Laser Fault Model

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Μάριου Τάμπα

Επιβλέπων : Κιαμάλ Πεκμεστζή

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Οκτωβρίου 2014.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Κιαμάλ Πεκμεστζή	Δημήτριος Σούντρης	Πέτρος-Παύλος Σωτηριάδης
Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2014

(Signature)

.....

ΤΑΜΠΑΣ ΜΑΡΙΟΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

The thesis took place at the LCIS laboratory (Laboratoire de Conception et d'Intégration des Systèmes)

Supervisors at the LCIS:

Athanasios Papadimitriou: athanasios.papadimitriou@lcis.grenoble-inp.fr

Vincent Beroulle: Vincent.beroulle@esisar.grenoble-inp.fr

David Hély: david.hely@lcis.grenoble-inp.fr

University Grenoble Alpes, LCIS 26000 Valence FRANCE

Abstract

Nowadays, digital integrated circuits (ICs) are found in all electronic devices and computer systems. Security and resilience of ICs is a modern subject that concerns scientists and hardware engineers. IC designers have recognized the importance of incorporating fault tolerance into microelectronic devices. In order to develop proper countermeasures for the security and the normal functionality of ICs, it is imperative to study the impact of attacks against circuits. Even further, study at the early stages of IC manufacture is mainly taken into consideration. There are different kinds of malicious attacks against ICs, cryptographic or not, such as hardware or physical attacks, as well as cryptanalytic attacks against the cryptographic algorithms. IC piracy using laser beam is the latest and most commonly applied by hardware hackers, who aim to extract information from secure ICs. Most of the times the aim of the hackers is to turn against chips that contain cryptographic algorithms, because those are the ones that hold the valuable data in encrypted form.

The object of the internship is divided in two parts. The first part includes the development of a layout extraction platform in order to model localized attacks on the layout of integrated circuits. Using OpenAccess™, an EDA tool provided by Si2 (Silicon Integration Initiative) that allows the interface with IC designs, we were able to design the extraction tool, study the logic components that lie underneath specific areas of the layouts (potential laser spots) and check their significance on the functionality of the ICs. Next and principal task of the internship is the validation of an RTL laser fault methodology, already proposed in the article [1]. In this thesis, this methodology is explained in details. During the internship, long time and effort was dedicated for the validation of this methodology, not only on layouts of AES designs, but also on some benchmark designs with technology node of 45nm. Several localized attacks on the layout were studied thoroughly in order to examine if such kind of attacks could be predicted from the RTL, the early and abstract stage of IC manufacture which models digital systems. In this way, we get a concept of the most critical logic components to be hit during a realistic laser attack.

The internship took place in the Laboratory of Design and System Integration, LCIS (Laboratoire de Conception et d'Intégration des Systèmes), which is located in the city Valence of France. LCIS is one of the 21 research labs of the Grenoble Institute of Technology and its research activities are oriented towards the specification, modeling, design, communication, validation, diagnosis and security of integrated circuits, embedded and communication systems. This work is considered a part of the global project LIESSE (Laser-Induced fault Effects in Security-dedicated circuitS). Among others, the goal of this project is to study and model the effects of laser shots onto submicronic circuits and provide efficient tools to prevent such laser attacks.

Keywords

Hardware Security, Cryptography, AES, Fault Injection, OpenAccess, Integrated Circuits, Layout, Laser attacks, Fault Model, Validation

Περίληψη

Στις μέρες μας, ψηφιακά μικροηλεκτρονικά κυκλώματα συναντώνται σε όλες τις ηλεκτρονικές συσκευές και σε συστήματα υπολογιστών. Η ασφάλεια και ανθεκτικότητα των ψηφιακών ολοκληρωμένων κυκλωμάτων (ΟΚ) συνιστά σύγχρονο θέμα που απασχολεί επιστήμονές και μηχανικούς hardware. Σχεδιαστές των ΟΚ έχουν αναγνωρίσει τη σημασία ενσωμάτωσης μηχανισμών ανοχής σφαλμάτων στις μικροηλεκτρονικές διατάξεις. Προκειμένου να σχεδιαστούν κατάλληλα αντίμμετρα που θα συμβάλλουν στην ασφάλεια και ομαλή λειτουργία των ΟΚ, θεωρείται επιτακτική ανάγκη η μελέτη της επίδρασης κακόβουλων επιθέσεων εις βάρος τέτοιων κυκλωμάτων. Ακόμη περισσότερο, μελέτη στα πρώιμα στάδια κατασκευής ΟΚ πρέπει να ληφθεί υπ'όψιν. Υπάρχουν διάφοροι τύποι κακόβουλων επιθέσεων ενάντια ΟΚ, είτε κρυπτογραφικών είτε όχι, όπως physical επιθέσεις ή επιθέσεις υλικού, καθώς επίσης και κρυπτογραφικές (μαθηματικές) επιθέσεις εις βάρος των κρυπτογραφικών αλγορίθμων. Ειδικά, η «πειρατεία» των ΟΚ χρησιμοποιώντας λέιζερ ακτινοβολία συνιστά την πιο σύγχρονη και εφαρμόσιμη τεχνική επίθεσης, αποσκοπώντας στην υποκλοπή απόρρητης πληροφορίας από τα ασφαλή ΟΚ. Στις περισσότερες των περιπτώσεων ο στόχος των hardware hackers είναι να «σπάσουν» το κλειδί των κρυπτογραφικών υλοποιήσεων στα ΟΚ, καθώς τέτοιου είδους κυκλώματα προστατεύουν πολύτιμη πληροφορία σε κρυπτογραφημένη μορφή.

Το αντικείμενο που πραγματεύτηκε η Πρακτική, και κατ' επέκταση το κείμενο της Διπλωματικής αυτής, μπορεί να χωριστεί σε δύο σκέλη. Το πρώτο σκέλος περιελάμβανε την ανάπτυξη μιας Πλατφόρμας Εξόρυξης Στοιχείων του Layout ΟΚ με σκοπό τη μοντελοποίηση της τοπικής επίδρασης επιθέσεων στο Layout των ΟΚ. Χρησιμοποιώντας το OpenAccess™, ένα EDA λογισμικό που παρέχεται από την Si2 (Silicon Integration Initiative) και επιτρέπει τη διεπαφή με την επιφάνεια ΟΚ, είχαμε τη δυνατότητα να σχεδιάσουμε την πλατφόρμα αυτή και να μελετήσουμε τα εξαρτήματα – συστατικά των Layout σε καθορισμένα σημεία (πιθανές περιοχές του Layout υπό λείζερ επίθεση), καθώς επίσης να μελετήσουμε τη συμβολή των στοιχείων αυτών στη λειτουργικότητα των ΟΚ. Επόμενο και κρίσιμο έργο της πρακτικής εργασίας ήταν η επικύρωση ενός RTL Λέιζερ Μοντέλου Σφαλμάτων. Το μοντέλο έχει ήδη δημοσιευθεί στο άρθρο [1]. Στα πλαίσια αυτής της Διπλωματικής, το μοντέλο περιγράφεται με λεπτομέρεια. Συνοπτικά, πραγματεύεται την μοντελοποίηση επιθέσεων λέιζερ στο RTL επίπεδο των ΟΚ. Στη διάρκεια της Πρακτικής, αρκετός χρόνος και ενασχόληση αφιερώθηκαν στην επικύρωση της μεθοδολογίας στα Layout, όχι μόνο κρυπτογραφικών AES υποκυκλωμάτων αλλά και κυκλωμάτων αναφοράς (benchmarks) με τεχνολογία υλοποίησης 45nm. Πολλαπλά σενάρια τοπικών επιθέσεων στο Layout (χρησιμοποιώντας προσέγγιση λέιζερ επίθεσης) μελετήθηκαν διεξοδικά προκειμένου να εξετάσουμε αν τέτοιες τοπικές επιθέσεις θα μπορούσαν να έχουν εκτιμηθεί και προβλεφθεί ήδη από το RTL, το πρώιμο και αφηρημένο στάδιο υλοποίησης ΟΚ που περιγράφει τα ψηφιακά συστήματα. Με αυτό τον τρόπο, μπορέσαμε να λάβουμε επίγνωση των πιο κρίσιμων στοιχείων στο Layout που κινδυνεύουν να χτυπηθούν από μια ρεαλιστική επίθεση με λέιζερ.

Η πρακτική έλαβε μέρος στο Εργαστήριο Σχεδιασμού και Υλοποίησης Συστημάτων (Laboratoire de Conception et d'Intégration des Systèmes LCIS), το οποίο βρίσκεται στην πόλη Valence της Γαλλίας. Το LCIS είναι ένα από τα 21 σύγχρονα τεχνολογικά ερευνητικά εργαστήρια του Πανεπιστημίου της Grenoble και οι ερευνητικές του δραστηριότητες προσανατολίζονται στη μοντελοποίηση, ανάπτυξη προδιαγραφών, σχεδιασμό, επικοινωνία, επικύρωση, διάγνωση και ασφάλεια ΟΚ, ενσωματωμένων συστημάτων και συστημάτων επικοινωνίας. Η πραγματοποιηθείσα εργασία θεωρείται μέρος του παγκόσμιου project LIESSE (Laser-Induced fault Effects in Security-dedicated circuitS). Μεταξύ άλλων, σκοπός του project αυτού είναι η μελέτη και μοντελοποίηση της επίδρασης επιθέσεων λέιζερ σε μικροηλεκτρονικά κυκλώματα, καθώς επίσης η ανάπτυξη αποτελεσματικών εργαλείων για την αντιμετώπιση τέτοιων επιθέσεων.

Λέξεις – Κλειδιά

Ασφάλεια Υλικού, Κρυπτογραφία, AES , Εισαγωγή σφαλμάτων, OpenAccess, Ολοκληρωμένο Κύκλωμα, Layout, Επίθεση με λέιζερ, Επικύρωση μοντέλου σφαλμάτων

Content

Acknowledgments

First, I would like to thank my professor from NTUA, Mr. Kiamal Pekmestzi, who supported my decision to do the internship in Valence. Moreover, I want to express my thanks to my supervisor in LCIS and PhD student of University of Grenoble, Mr. Athanasios Papadimitriou, for his advice, comments, practical indications and his patience during my internship in the Laboratory. In addition, I would like to express my gratitude towards the professors of University of Grenoble, Mr. David Hély and Mr. Vincent Beroulle for their guidance and their willingness to assist in my work and provide with corrections and advices during the whole period of my internship. Finally, I would like to thank my family for their unwavering support on me.

Chapter 1: Introduction

1.1 Motivation

Nowadays, security on digital integrated circuits (IC) is an extremely important subject, since ICs are involved in many critical aspects of our lives. Digital ICs can be found in many common electronic devices, such as cell phones, computer systems or smart-cards, credit cards, pay-per-view TV, etc. Security algorithms are implemented in order to ensure appropriate functionality of the circuits, however those algorithms often turn out to be inefficient. Hostile intrusions against secure ICs have been developed alongside the growth of silicon technology, enabling adversaries to unfold their malicious intentions. Especially, laser proves to be one of the most efficient and controllable means of attack. Scientists have concluded that innovative, proper kinds of countermeasures must be introduced, as current state-of-the-art countermeasure tools prove to be incapable of preventing several attacks.

Designing a secure integrated circuit requires implementing protection against malicious threats. The design and integration of efficient countermeasures depend on the methods available for an early validation in the design process. Study and analysis at an early level of abstraction, such as the Register Transfer Level (RTL), can provide the means to efficiently expose any vulnerabilities of security oriented circuit designs, and contribute to the implementation of both defensive and preventive mechanisms. At the same time, RTL analysis can lead to the enhancement of the design flow with the capability to avoid costly feedback runs [1].

This work focuses on the development of a tool for the validation of a fault methodology applied at the RTL. The methodology is based on the partitioning of the elaborated RTL net-list (RTL logic circuit) and attempts to model the locality of laser attacks at RTL of digital ICs, either cryptographic or not. Several assumptions concerning the manifestation of faults are implied within the method. The main objective is to prove that the study of early stage of abstraction in circuit design flow, Register Transfer Level, can prove an effective source of information concerning the prediction of localized IC attacks. This information can ultimately contribute to the evaluation of new countermeasures against laser attacks.

1.2 Scope of the Internship

1.2.1 Development of Layout Extraction Tool

The scope of the internship and of this thesis, in extension, is twofold. The first part of the scope is the development of a layout extraction platform that is used in order to validate the RTL laser fault injection methodology, already published in the literature [1]. The platform is designed based on OpenAccess™, a well-known C++ API, that targets to enable and facilitate the interface with IC design database. Using the layout extraction platform, we have the capability to inspect potential localized attacks on the layout and extract information from the layout. The nature of these components is either combinational logic (logic gates) or sequential logic (registers, flip flops). The ICs that we have chosen to validate include certain cryptographic implementations of the AES with different countermeasures, as well as certain benchmarks, the layout of which has been implemented with the Nangate Open-Cell process technology of 45nm.

1.2.2 Validation of the Proposed Methodology

Concerning the second and most important part of the scope of internship, it is the part that deals with the validation of a fault methodology already proposed in the literature [1], concerning the modeling of laser attacks at the RTL of digital circuits. The methodology is based on Cone Partitioning, which constitutes the partitioning of the elaborated (non-optimized) net-list of an RTL description into logic cones. Briefly explained, a logic cone starts at a flip flop and ends to other flip flops or primary inputs. The last elements constitute the boundary of the expanding cone. Each cone corresponds to a single flip flop. In other words, the cone is the fan-in network of each single flip flop. For the validation, the cone of each flip flop in the RTL net-list is identified and extracted, with respect to the connectivity with its fan-in network.

Moreover, a single cone at the RT level may contain numerous elements of combinational logic. There is this scenario where many logic elements appear in more than one RTL cones, making these cones intersect with each other. By thoroughly examining, for each element of a single cone, where else it belongs to, the methodology extracts the set of RTL cones that intersect with this particular cone. In this thesis, the groups of RTL cones that intersect are referred as “RTL Intersection Sets”. Each RTL Intersection Set corresponds to a single RTL cone.

Validation flow proceeds with the extraction of multiple localized spots (bounding boxes) on the layout and their detailed examination with respect to the logic components found underneath. These spots are considered as areas on the layout affected by a potential localized attack (laser beam), an attack that injects faults to the affected components included in the area. Such attack affects either sequential logic, i.e. registers (group of flip flops) or combinational logic, i.e. gates, or both. According to the proposed methodology of [1], for a certain localized spot, faults can be modeled by their injection in flip flops. In case the spot covers combinational logic, then the faults are not stored in the combinational elements, but they are modeled at the flip flops that include the affected combinational elements in their

fan-in network. The fan-in network of each flip flop on the layout is technically another cone partitioning technique, this time not at RTL, but at the Gate Level of the design. Gate cone partitioning endorses the same functional formality as the RTL Cone technique, but this time with respect to the connectivity nets on the Gate Level net-list. Gate Level description (net-list) is the result of synthesis on the RTL net-list and it practically represents the functional relations among physical elements encountered on the layout. Ultimately, the set of flip flops affected directly or indirectly by the localized spot is extracted and stored for further analysis. The same procedure takes place for all localized spots under examination. The sets of flip flops extracted from the layout are referred as "Virtual Spot Sets", in this thesis.

Finally, the validation of the methodology is carried out by checking if the Virtual Spot Sets, extracted from several localized spots, match exactly or constitute subsets of the RTL Intersection Sets. In other words, for each Virtual Spot Set of flip flops we check if it can be found in, at least one, RTL Intersection Set. That is the main reference point for associating the early stage of RTL design with the final stage of layout implementation. When a Virtual Spot Set is contained in one or more RTL Intersection Sets, the proposed fault model succeeds and we have an estimated prediction of the consequences of a potential laser attack, already at an early phase of design. It is a valuable information that can ultimately lead to the evaluation of existing and the design of new countermeasures against laser attacks.

1.3 Organization

This thesis is structured as follows. Chapter 2 gives general introductory information on hardware security, mentioning the need for cryptographic algorithms and the aspects of life where we meet secure integrated circuits. Furthermore, there is an extensive reference on attacks against secure systems, the nature of existing attacks a hacker can unleash and the impact on the semiconductor. In this thesis, what we mainly focus on is the laser attack, which is part of fault injection attacks. After presenting briefly the Advanced Encryption Standard, which is implemented in many secure systems, some examples are stated in order to show in more depth the procedure of injecting hardware faults that results in the exposure of secret key. Finally, it is stated that, nowadays fault attacks can be used as a validating and testing methodology for the resilience of hardware systems.

Chapter 3 focuses on Fault Modeling, presenting different types of faults and fault models that are used to describe the impact of IC attacks. Terminologies such as fault and fault model are explained, as well as previous fault injection techniques, in order for the reader to get an overview of the different methods used to introduce faults on integrated circuits.

Chapter 4 details on the fault methodology which our work is based on. Cone partitioning at the RTL is explained thoroughly, terms like logic cone, faults the method represents and the assumptions on which the methodology is based, are presented. Finally, some limitations on the method are stated.

Chapter 5 presents the development of the Layout Extraction tool, which was used for the validation of the RTL fault method. The tool was designed using the OpenAccess™ C++ API. Particular coding functions and techniques for implementing the spot partitioning are explained. Finally, there is

a reference on Glade™, a layout viewer that facilitates the analysis and offers a visual perspective on the designs under test.

Chapter 6 describes the validation algorithm and the technical processes that took place for its completion. Extraction of RTL and Gate Level Intersection Sets, as well as extraction of Virtual Spot Sets is explained in detail, according always to the theory and assumptions of the RTL fault methodology.

Chapter 7 presents the results produced by the analysis, which was applied on four different designs. Tables and layout mappings offer a reflection of the analysis outcome and validation percentages are explained revealing method's efficiency.

Chapter 8 contains the conclusion of the thesis, the most notable points in the internship and a glimpse on future work concerning the specific subject.

Chapter 2: Hardware Security

2.1 Introduction

Nowadays, hardware security is a major subject concerning many scientists and engineers. Even more, methods that break down hardware security is a topic susceptible of research. Early evaluation on the design process of ICs may prove decisive for the installation of proper countermeasures that improve the resilience of digital systems. That is the main point that concerns the work, in the context of the internship. Earlier manufacturing stages on IC design flow, such as RT level can provide the means for implementing accurate safety measures. Nevertheless, the term "hardware security" is mostly associated with cryptographic mechanisms. Usually, secure systems are the center of attention for hardware hackers, as valuable information is secretly banked in these systems. Security on hardware annotates the incorporation of cryptographic algorithms on digital systems. Thus, cryptography is the main scientific domain behind system security.

Cryptography is the study and the practice of methods for secret communication and writing of messages. Its aim is hiding their meaning to everybody except an intended recipient, who will be the only one who can uncover the secret and read the message. Cryptography, in general, may be used to provide any of following properties:

- Confidentiality: To prevent the unauthorized disclosure of data, only an authorized receiver should be able to extract the message contents from its encrypted form.
- Integrity: The receiver should be able to determine whether he receives the original message or an altered version.
- Authenticity: The receiver should be able to check from the message the sender's identity and the message origin or the path it followed.
- Non-repudiation: The sender should not be able to deny sending the message.

Modern cryptography is based on mathematics, computer science, and electrical engineering. Cryptographic algorithms, known as ciphers, use secret keys for encrypting the given data, known as plaintext, thus generating a cipher-text, and for decrypting the cipher-text to reconstruct the original plaintext. The keys that are used for the encryption and decryption steps can be either identical (or nearly related), leading to what are known as symmetric key ciphers, or they can be completely different, leading to what are known as asymmetric key (or public key) ciphers. Symmetric key ciphers have simpler, and therefore faster, encryption and decryption processes, compared to asymmetric key ciphers. Symmetric ciphers have the main weakness that the secret key is shared, which may lead to its discovery by malicious hackers, and therefore, must be changed in frequent periods.

Consequently, cryptography is an indispensable tool for protecting information in electronic circuits and computer systems. Today's cryptosystems contain secret keys for cryptographic algorithms

used to protect confidential information or to provide authentication mechanisms. These keys are the target of malicious hacking activity.

The need for secure chips, nowadays, is more necessary than ever. Cryptographic algorithms are being implemented in an increasing number of, not only consumer products, but also services. As an example, some categories using cryptographic ICs are mentioned:

- Car industry: anti-theft protection, spare parts identification
- Service providers: access cards, payment token, RFID tags, electronic keys, software license dongles
- Mobile phone manufacturers: batteries and accessories control
- Manufacturers of entertainment systems: copy protection, consumables and accessories control
- Manufacturers of devices and equipment: protection against cloning and reverse engineering, IP protection (hardware, software, protection of algorithms)
- Banking industry: secure payment cards, secure processing
- Military applications: data protection, encrypted communication

For this reason, they are always the subject of much research aimed at improving their security and resistance to any unauthorized interference. The current work aims towards this direction, meaning to provide with the proper feedback for the design of countermeasures that resist attacks on ICs, mostly on the cryptographic ones. The particular attack our work has focused on is the laser beam, which constitutes a state-of-the-art technique for IC fault manifestation. But, in general malicious attackers eventually use different kinds of techniques in order to accomplish their purpose and snoop secret information. Detection of the secret key may require the parallel usage of multiple different attacks, concerning the most commonly used cryptographic algorithms.

2.2 Different Attacks on ICs

In this sub-chapter, different attacks against digital systems are described thoroughly in order to give the reader a notion for the outbreak of malicious attacks. Attacks on digital (cryptographic) systems can be divided into cryptanalytic or mathematical attacks and hardware attacks. It is common for an IC hacker to make use of more than one type of attack so as to accomplish his purpose.

- **Cryptanalytic or Mathematical Attacks**

These attacks search for vulnerabilities in a cryptographic schema or algorithm in order to deduct the keys by mathematical methods. When an opponent is not able to find any weakness in a cryptosystem that could help him perform a cryptanalytic attack, he may use an exhaustive searching of the key. An exhaustive search, or as it called brute-force attack, for finding the key is a cryptanalytic attack that can, theoretically, be used for finding a key that maps a plaintext to its corresponding cipher-text. It requires

checking all possible keys until the correct one is found. In practice, it requires checking, on average, half of the entire search space for the key. For this kind of attack, an automated software can be used in order to generate a large number of consecutive guesses as to the value of the desired data.

The key length of reliable cryptographic algorithms increases continuously beyond the computer's capability of calculating and finding the keys. So, a brute-force search for the keys is not able to give any answer in a reasonable amount of time, except if it has been applied as a complement of another attack that can reveal a significant part of a key.

The example of a real cryptographic algorithm is quoted. The Data Encryption Standard, called DES, was, historically, approved by former US National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the United States in 1976. DES is a block cipher that encrypts messages with a 56-bit key and it was considered as a secure encryption method at that time. But, as a result of the growth of computation capabilities, it was announced by the US National Institute of Standards and Technology (NIST) that they wished to choose a successor to DES. They mentioned that the new standard will be known as Advanced Encryption Standard or AES. The new encryption algorithm was chosen with 3 key sizes of 128, 192 and 256-bit, all greater than the 56-bit key of the outdated DES.

The previous fact, revealing the historical transition from DES to AES, states the growth of minimal key size from 56 bits to 128 bits, underlining the extremely important attribute of key size against brute-force attacks for information security. This example is much more obvious when we consider that each additional bit doubles the required computations in an exhaustive search (e.g. 2^{128} computations for 128 bits). Nowadays, cryptanalytic attacks, although being very cheap, are not applicable as long time and effort is required for leaking out the secret information. Thus, attackers have turned to other efficient attacking techniques that need extra hardware material. Many times, a combination of cryptanalytic and hardware attacks is used.

- **Hardware Attacks**

This large family of attacks targets mostly secure hardware, and by hardware it is meant the physical implementation of crypto-algorithms on integrated circuits. Hardware attacks are divided into invasive and non-invasive. Invasive attacks are those which leave behind such a footprint. They are a penetrative to the material and leave tamper evidence or even destroy the physical circuit. Non-invasive are able to hide their presence so as to have no effect on the system other than the introduced faults. Some common techniques, used against hardware security, are explained as follows:

- Side-channel attacks: technique that allows the attacker to monitor the analog characteristics of power supply and interface connections and any electromagnetic radiation.
- Micro-probing: technique used to access the chip surface in physical way, so one can observe, manipulate and interfere with the device.

- Reverse engineering: technique used to understand the inner structure of the device and learn or emulate its functionality. This method requires the use of the same technology available to semiconductor manufacturers and gives similar capabilities to the attacker.
- Fault Attacks: usage of abnormal environmental conditions to generate faults in the system that provide additional access.

2.2.1 Side-channel attacks

Secret information, such as the key of the encryption algorithm, can leak out through side-channels. A side-channel attack is a non-invasive attack, performed based on information gained from the physical implementation of a crypto-system. This new class of physical attacks against cryptographic circuits is drawing much attention from attackers' part.

Side-channel attacks do not process or open the package of target systems. The attacker only observes side-channel information from system modules. This category include timing attacks using operation times, as applied in [2]. Clock side-channel attacks are based on the fact that the individual computation steps that are required during the encryption are highly dependent on the bits of the secret key and, thus, the time needed for these steps is directly correlated to the bits of the secret key. Moreover, power analysis attacks using power consumption, as stated in [3] is another example of side-channel attack. The authors in [3] apply Differential Power Analysis (DPA), a technique to automatically locate correlated regions with respect to power consumption levels, so attacker needs little to no information about details on the target system, as long as they hold the information of power flow. In addition, electromagnetic radiation consists a passive side-channel attack and turns out to be a particularly serious issue for devices that pass keys or secret intermediates across a data bus. This example of attack is described in [4]. As it is stated, even a simple AM radio can detect strong signals from many cryptographic devices, allowing experiments to be conducted for further investigation.

Side-channel attacks have proven to be effective and incur a relatively low cost. Furthermore, once a side-channel attack technique has been developed and become public, high technical skills or expensive equipment are not required to apply it in practice. Side-channel attacks have become a major industrial concern in the last years and resulted in an intensive research effort to develop suitable countermeasures that can defeat the attacks, or at least make them more difficult and time consuming to perform. Many different types of countermeasures against this type of attack have been developed, including: restructuring of the cryptographic algorithm, shielding of the device, randomizing the computation, using power independent implementation, and others.

2.2.2 Micro-probing

Micro-probing consists an invasive attack. Its major component is a special optical microscope. On an arm of the microscope, the attacker installs a probe, which is a metal shaft that holds a long tungsten-hair, which has been sharpened and allows the attacker to establish electrical contact with on-chip bus lines. The probe is connected via an amplifier to a digital signal processor card that records or overrides processor signals and also provides the power, clock, reset, and I/O signals needed to operate the processor via pins.

In [6], the authors show that by locally observing the value of a few RAM or address bus bits (or possibly a single one) during the execution of a cryptographic algorithm, typically by the mean of a probing needle, an attacker could easily recover information on the secret key being used. The attacks presented in the article apply to public-key cryptosystems such as RSA, as well as to secret-key encryption schemes including DES and RC5.

Technological progress concerning countermeasures against micro-probing, is increasing the costs to the attackers. For modern deep submicron semiconductor chips, attacker must use very sophisticated and expensive probing technologies in order to remove layer after layer and reach the target point on the surface of the IC. Especially, in case there exist voltage, light or top metal sensors that prevent an opened chip from functioning, attackers are forced to turn to other attacking methods.

2.2.3 Reverse engineering

Reverse engineering is an invasive and destructive form of analyzing a crypto-IC. The attacker grinds away layer after layer of the IC and takes pictures with an electron microscope. With this technique, it is possible to reveal the complete hardware and software parts. The major problem for the attacker is to bring everything into the right order to find out how everything works. The IC manufacturers try to hide secret keys and operations by mixing up memory positions, using bus scrambling. Moreover, they implement sensors to detect and prevent such attacks. This kind of attack is not very common because it requires a large investment in effort and special equipment that is generally only available to large chip manufacturers. Furthermore, the payoff from this attack is low since other attack security techniques, like sensors, are often employed.

An example of reverse engineering attack is stated in [7], where the authors describe a mostly automated process that can be used to cheaply determine the functionality of previously unknown cipher on the NXP Mifare Classic RFID tag, the world's most widely used cryptographic RFID tag. This is done by using a combination of image analysis of circuits and protocol analysis and can be feasible also for larger chips. It is stated that reverse engineering silicon is a cheap and effective way of overpassing IC security, even when very little is known about a cipher or about any software implementation.

2.2.4 Fault Attacks

Fault Attacks is the category that gathers most of the attackers' efforts, thus this type is what mainly concerns our work. It constitutes the intentional introduction of faults in hardware systems. These attacks are considered as semi-invasive attacks, with intermediate cost of implementation and very effective results. It is mostly preferable by the attackers, while it provides some serious advantages over the previous attack techniques. A variety of fault attacks exist, where some hardware fault (an unexpected condition or defect) results in a processing mistake that is advantageous for the attacker. Methods of introducing faults include: supplying noise power or clock signals, voltage glitching, excessive temperature, radiation or high energy beams such as UV, laser, etc.

This category, although being the most important kind of hardware attacks, was intentionally left last to analyze as it deals with the fault model discussed in the thesis. Our work is associated with the study of laser fault attacks: How they are injected and how faulty outputs are used to endanger the secrecy of cryptographic devices. For this reason, we focus on the fault injection technique even more, and explain how it can be used for testing the resilience of digital systems.

2.3 Fault Attack Techniques

Nowadays, most of the research dedicated on IC hack prevention, focuses on fault attacks, which is considered a modern and effective manner of intruding into secure systems. These kind of malicious assaults consist in introducing faults and forcing a cryptographic device to execute erroneous operations, hoping that the result of that wrong functionality will leak out information about the secret parameters involved. Fault attacks have proven to be practical and pose a risk against the secure operation of crypto devices. Contrary to side-channel attacks, where the side channels (power consumption, electromagnetic radiation, etc.) of an integrated circuit are observed in order to reveal information, fault attacks try to have an active impact on the IC's operation by skipping or corrupting security operations, corrupting registers and in general perturbating the IC's core operations. Being the most technologically advanced (state-of-the-art) and effective method, allowing high controllability, to break hardware security, it is reasonable for our research to focus on and study thoroughly this category, and more specifically, the laser attacks. However, for information fullness towards the readers, we describe briefly certain types of Fault attacks. The most common methods of fault injection into digital ICs or embedded devices are mentioned as follows:

- Voltage/Clock fault injection, by introducing dips or spikes in the VCC/Clock line of the target
- Temperature fault injection, by heating/cooling the IC outside of its thermal tolerance range
- Electromagnetic fault injection, by using a magnetic field close to the IC.
- Optical fault injection, by targeting certain areas of the IC with a laser

2.3.1 Voltage/Clock Fault Attack

Voltage and Clock fault injection techniques turn out to be a common and quite successful series of attacks against ICs. They are also the most applicable ones, by using particular voltage and clock glitch sensors, respectively. Voltage glitch sensors may not allow the voltage in the supply line to exceed a certain range and, in the same way, clock sensors control the level of voltage in the clock input. Both these fault injection techniques require preparation for the target, leaving evidence of intrusion, in the form of isolating the power/clock lines (invasive attack). An example of voltage glitching is demonstrated as follows:

The normal operation of an IC is at its nominal voltage (say 3.3V). If one interferes by dropping the voltage down to 1V, he provokes a fault injection (Fig. 2.1). At that moment, the input voltage to certain gates within the chip will be too low due to the lack of supply voltage. Thus, these gates will receive an input voltage which is below the threshold that indicates whether the signal is a zero or a one, no matter what value it was supposed to be. By increasing the voltage again to the nominal voltage of 3.3V, we get a functioning chip that just failed to execute one of its operations. For instance, it failed to execute a conditional jump and fell through to the code that was expected to have executed.

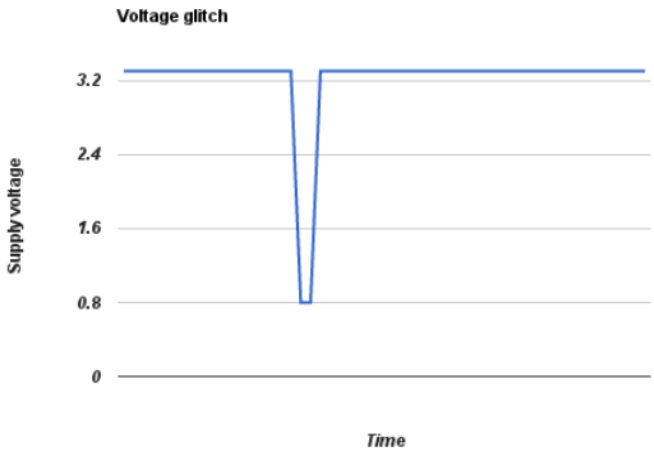

Fig 2.1: Example of voltage glitching. The supply voltage is set to 0.8V during a short moment of time.

In this case, the trick is to discover some proper parameters for the glitch: voltage drop level, length of the glitch and the timing. Typically, if voltage drop and length of glitch are too small, the chip will function properly. If they are too large, then the chip will just either mute or reset, or even maybe get physically damaged. Of course, it is mandatory that the attack timing is accurate, otherwise the attacker will never see the effects he wants to get.

Clock glitching is similar to V_{CC} glitching in the sense that it affects another critical parameter of the chip that can be controlled by the attacker. In this case, the attacker is injecting spurious clock cycles that are way shorter than the original clock cycle (Fig. 2.2). Since the internal logic of the chip operates based on its clock, a short clock cycle will trigger a new operation before the results of the previous one were completely computed or propagated through the device. For instance, proper function includes

multiplication of two values, and then addition of a third value to them. Normally, multiplying values takes longer than adding them up. Thus, the clock frequency for a chip that only performs these two operations would be long enough for the multiplication to occur and its result to be ready at the input of the next stage, since that is the critical operation. In case addition precedes the result of the multiplication, then the data will turn to be invalid. Thus, there will be failure at computing the correct result. Clock glitching exploits exactly that situation. Again, finding the right parameters in this case is the key to success.

Fig 2.2: Example of clock glitching. A very short spurious clock cycle is inserted at the beginning of a normal cycle.

2.3.2 Temperature Fault Attack

Cryptographic circuits operating outside of the specified range of temperature will start to malfunction. That situation is exploited by attacker in order to perform temperature fault injection. This type of fault injection is a hard technique to be achieved and controlled because of the exact timing needed between the target operations and the temperature variations that are to take place. Usually, a combination of voltage and temperature fault attack takes place. The authors in [8] demonstrate a way to break even state-of-the-art ciphers, by lowering V_{DD} to the point when individual logic gates are not able to switch, while increasing ambient temperature. They state that low-cost voltage and temperature manipulations can be used for high-precision fault injection required to break state-of-the-art ciphers.

2.3.3 Electromagnetic Fault Attack

A new type of fault attacks is introduced, which uses an electromagnetic field to induce faults in the target device. The Electro Magnetic field Fault Injection (EMFI) perturbation is effective and non-invasive. This attack can bypass the countermeasures, such as light or motion sensors and, by its nature, it is harder to detect during run-time, leaving little or no evidence of intrusion. Article [9] considers the use of magnetic pulses to inject transient faults into the calculations of a RISC micro-controller running

the AES algorithm. This technique enables to fault every byte of the AES state on a non-protected software implementation of an AES, running on an 8-bit micro-controller.

2.3.4 Optical Fault Attack

Optical fault attack, or most commonly referred as laser fault attack, uses a light beam to inject faults into semiconductor devices. The light beam basically consists of a number of photons carrying a certain amount of energy. Roughly, when these photons reach a semiconductor (typically the silicon in electronic devices), their energy is absorbed by the semiconductor. Given enough energy, electrons that would otherwise be within the semiconductor will start to move, creating current. So, this means that some of the transistors in the chip will actually change their state, when such change should not happen.

The big difference between this fault attack technique and the previous mentioned ones is that, in this case the attacker actually has spatial selectivity (or resolution). In a laser attack the opponent usually controls the beam's diameter, wavelength, the amount of emitted energy, and the exposure's duration. Attacker can choose which parts of the chip to attack by pointing the laser beam on them. Of course, this is very powerful but at the same time it increases the complexity to the attack, because he needs to find the sensitivity spots of the chip. As before, there are a number of parameters one needs to take into account in order to successfully inject faults. Some of them are beam exposure timing and length, wavelength of the injected light and amount of energy injected.

Moreover, this attack is semi-invasive, meaning that attacker needs to open up the chip package so that the light radiation can reach the level of the die. Otherwise, the light will be blocked by the package or, in case of a smart-card, the plastic around the die. Thus, this attack provides additional power at the cost of additional complexity, as usual. In terms of hardware level protections, this is also the most difficult attack to prevent. Typically light sensors are scattered around the chip, but manufacturers cannot place sensors everywhere because of high cost, so there is always open spots. In this work, we have focused on laser attacks in order to validate a fault injection model that describes fault locality at the RT Level. In conclusion, the reasons for specifically deepening into the effects of laser attack underlie to its attributes over the rest of the attacks, which are summarized as follows:

- Complete controllability over the fault location. That means that the attacker can turn against selected components on the chip, thus affecting specific bits.
- Precise controllability on the timing of attack (the desired exact time can be met).
- Advanced controllability over the range and spreading of attack, meaning that attacker is able to roughly select the number of bits affected. Therefore, attacker can target to single faulty bit, or few faulty bits (e. g., a byte or word), or even a random number of faulty bits (bounded by the length of the affected variable).
- Variable power level of the laser leads to different kinds of impacts on fault location. By operating the laser at a low level of power, attacker can induce transient faults, i.e. faults that cease to exist after a short period. In case of high power level, the impact might be destructive, thus irreversible.

Concerning IC protection, countermeasures against fault attacks require a combination of hardware and software prevention and detection mechanisms. Typically, what is needed are sensors at the hardware level and double-checking and redundancy at the software side. Due to the difficulty of completely preventing this kind of attacks, fault attacks are nowadays one of the main threats to secure hardware. In addition this difficulty, combined with the physical nature of the attacks, indicates that simulating or emulating on testing devices these attacks is typically not enough to assure appropriate protection levels, making fault attacks a notable testing key for secure hardware [10]. The next chapter presents various efforts of studying and testing the AES algorithm against fault attacks.

2.4 Fault Attacks on the Advanced Encryption Standard AES

2.4.1 The Advanced Encryption Standard (AES)

In this sub-chapter, we present some fault attacks applied on the Advanced Encryption Standard. This algorithm was studied during the internship and testing efforts are dedicated on some implementations of AES with different countermeasures. AES is a symmetric method and is based on Rijndael cipher. It can grant a high level security using a reasonable calculation time. AES was quickly adopted for many systems and products after NIST validation in 2001. Many types of attacks have been studied by researchers with the intention of improving AES incorporations by suitable countermeasures.

AES is an algorithm that performs message encryption processing by data blocks of 128 bits at input and output using a key size of 128, 192 or 256 bits respectively in 10, 12 or 14 rounds (after a short initial round) according to the size of the key. Encryption includes two separated processes:

- Key Scheduling to derive the round keys from the secret key
- Data encryption

Decryption also is divided into two separated processes:

- Key Scheduling to derive the secret key from round keys
- Data Decryption

For the initial round in AES-128 bits, the algorithm uses the secret key as the round key. But for each following round, the corresponding round key is calculated from the previous one. Figure 2.3 shows

the different operations of the AES algorithm. We use AES to refer to AES-128 and we use the "K" prefix plus the number of a round to refer to a round key (e.g.K_1 for the round key of the 1rst round)

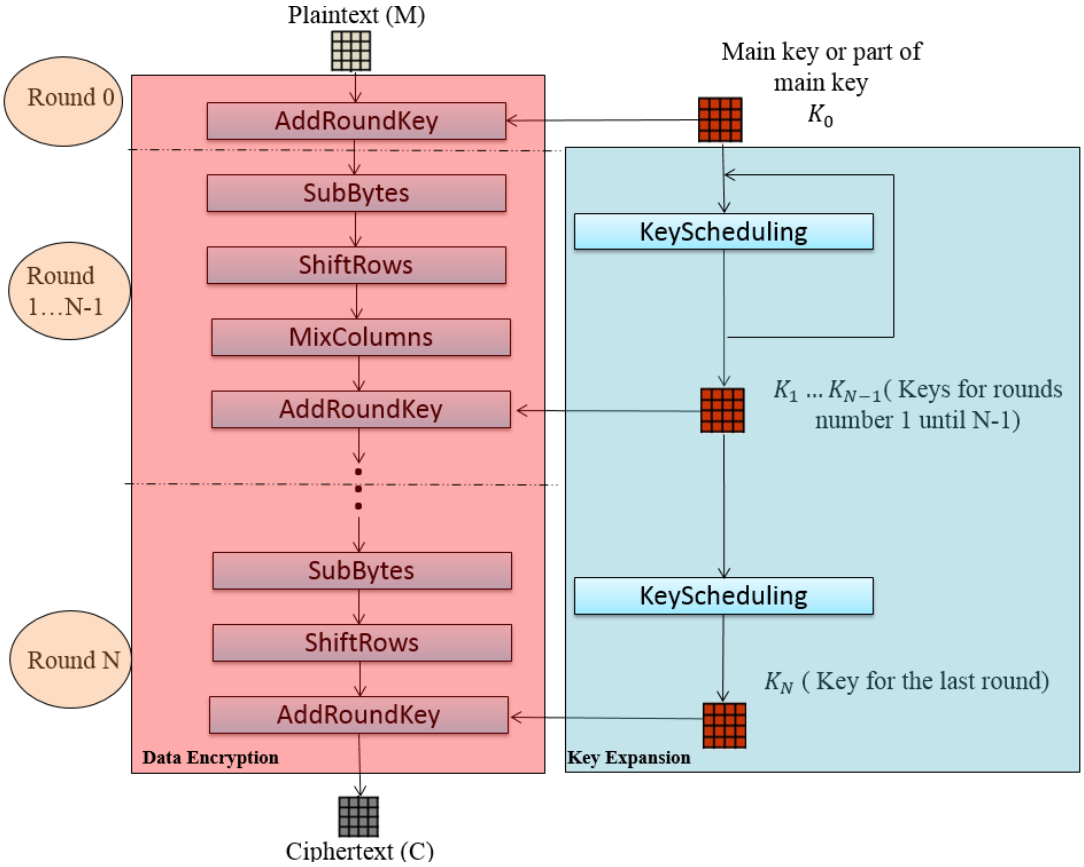

Figure 2.3: AES general outline

To encrypt a plaintext, namely M, according to the implementation of AES, usually at the beginning of algorithm execution, all the round keys are computed from the main key and are stored in the memory. Then, the encryption process begins and takes separated blocks of 16 bytes (128 bits) from M as input and put each block in a matrix of 4x4 bytes. Each round of the algorithm, except the initial and the last ones, includes 4 steps:

1) At the beginning, it exchanges the value of each matrix element, i.e. one byte value, by the corresponding value in a fixed substitution table (SubBytes or SB).

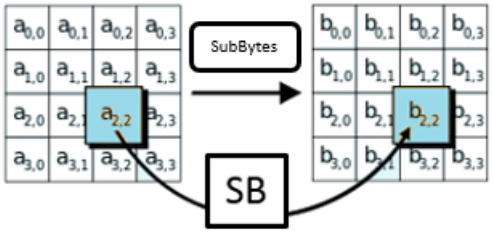

Fig 2.4: In the SubBytes step, each byte is replaced with its entry in a fixed 8-bit lookup table, *SB*; $b_{ij} = SB(a_{ij})$.

2) Then, it executes a rotational operation on the matrix rows (ShiftRows or SR). It cyclically shifts the bytes in each row by a certain offset. The offset is decided according to the row index (first:0, second:1, third:2,..). That means the first row is left unchanged.

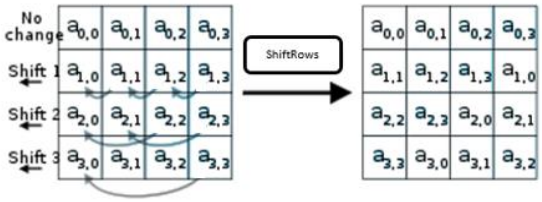

Fig. 2.5: In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left, according to row index

3) In the third step, the algorithm applies a linear transformation to each element and combines it with other values of the same column with a different coefficient of 1, 2 or 3 for each element (MixColumns or MC) under the specific rules of $GF(2^8)$. This step guarantees the distribution of the information of each byte on 4 bytes and increases security of encrypted messages.

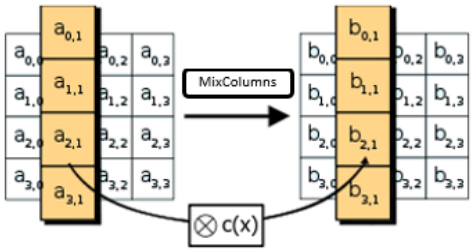

Fig. 2.6: In the MixColumns step, each column is multiplied with a fixed matrix

4) Finally, in the last step of each round, a bitwise XOR operation is performed between the value of each element and the corresponding byte on the round key (AddRoundKey or ARK).

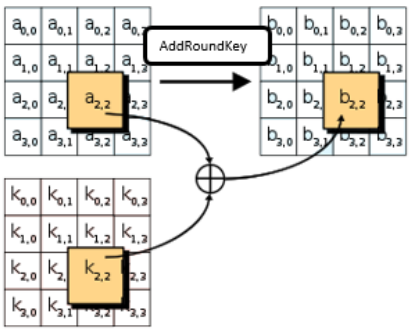

Fig. 2.7: In the AddRoundKey step, each byte is combined with a byte of the round-dependent key using the XOR operation

Concerning the procedure of round key computation, the 128-bit AES algorithm takes the main key and performs a key expansion routine to generate 10 round keys. Each expanded round key consists of a linear array of 4-byte words, denoted as W[i]. There are three transformation functions in the key expansion process as follows:

- RotWord is a function that takes a word $[a_0, a_1, a_2, a_3]$ as an input, performs a cyclic permutation and returns the word $[a_1, a_2, a_3, a_0]$.
- SubWord is a function that takes a word composed of 4 bytes and applies Sbox to each byte.
- Rcon[i] is a round constant word given by $[x^{i-1}, \{00\}, \{00\}, \{00\}]$, with x^{i-1} representing powers of x (x is denoted as $\{02\}$ in the field GF(2^8). Note that i starts at 1.

The following figure shows the AES key expansion process. RK_0 is the initial round key identical to the main secret key. The rest of round keys are generated by the key expansion process.

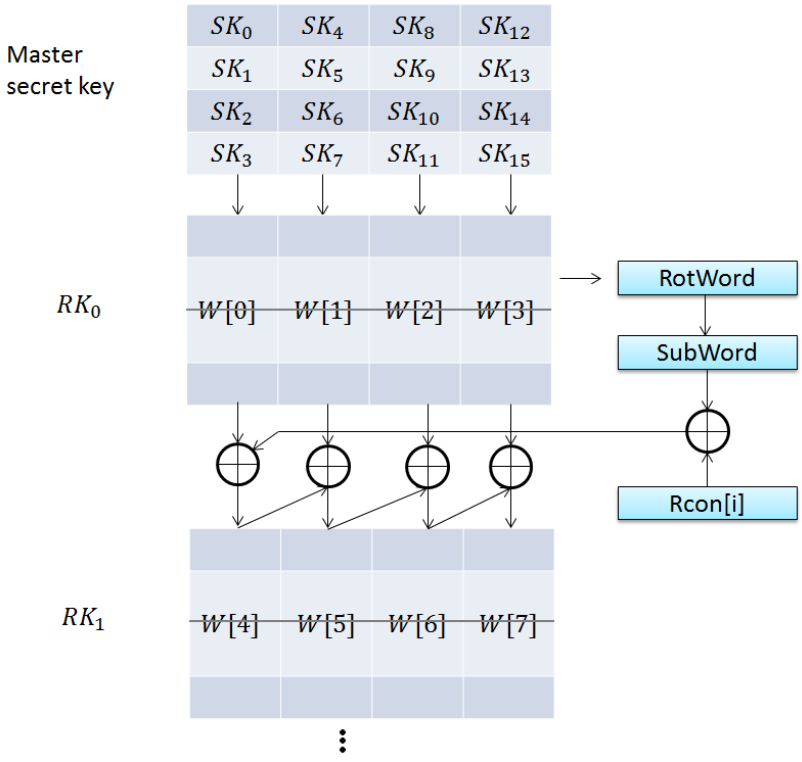

Fig 2.8: AES key expansion process

Currently, AES encryption is widely used for governmental, military and commercial purposes. Therefore, it has opened a new and large domain of research on the security of cryptographic circuits.

2.4.2 Different Methods of Fault Attacks on the AES

Different types of fault attacks on AES have been studied in general by researchers. These specific fault attacks can be categorized in certain categories, according to their methodology or mathematical implementation. Also, some evidence, describing the outcome of these attacks is stated.

- **Differential Fault Analysis (DFA)**

This attack depends on introducing faults into key-dependent cryptographic operations through physical intrusion. It is based on gaining some insights into the secret data handled by the circuit and then finding the secret key by comparing faulty cipher-texts with the corresponding (correct) cipher-texts.

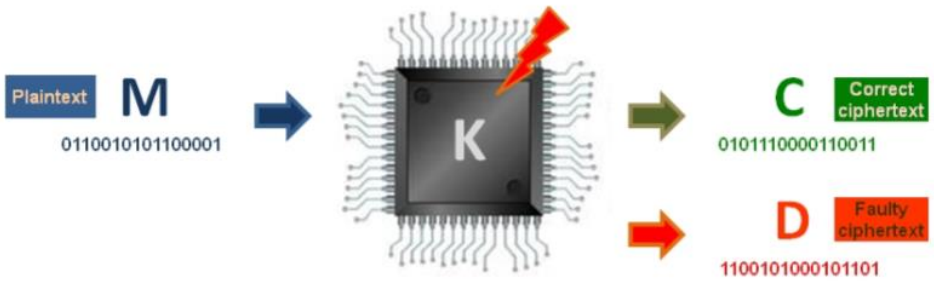

Fig. 2.9: An overview on Differential Fault Analysis [12]

The authors of [12] presented a theoretical DFA attack on AES. This attack required the injection of a single-byte fault into the temporary cipher-text between the MixColumns output of the antepenultimate round and the MixColumns input of the penultimate round to be successful.

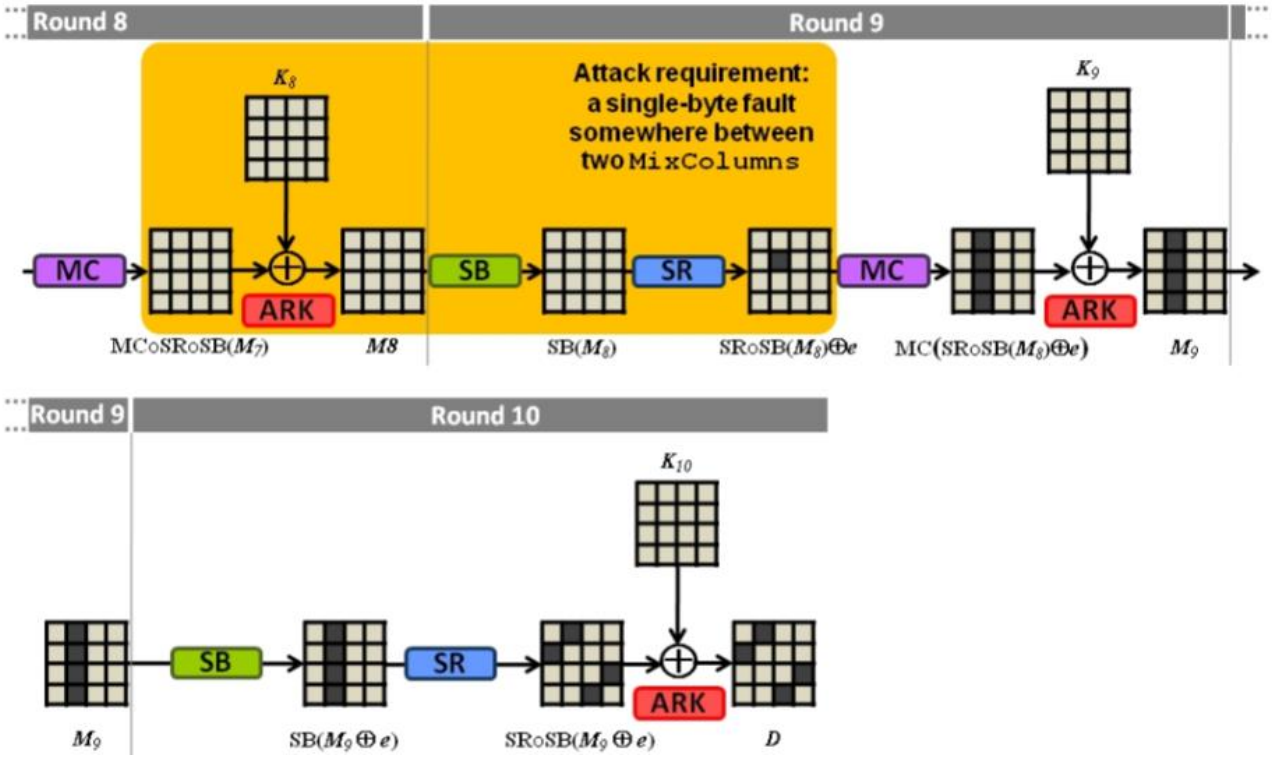

Fig. 2.10: Propagation of a single-byte fault at MixColumns input of the round 9 [12]

Figure 2.10 shows a tuning window that extends between MixColumns exit of round 8 and MixColumns entry of round 9 for a single-byte fault injection as the attack's requirement. The same Figure also presents the fault propagation and diffusion into four bytes. The attack scheme allows to infer some information on the four corresponding bytes of K_{10} by processing the correct and faulty cipher-texts and checking over the list of all the related possible single-byte faults. By repeating this process twice (i.e. by iterating the attack for a different plaintext) the exact value of the four bytes of $K_{10}$is found with a success rate of about 98%. The procedure is repeated to target K_{10}'s remaining bytes. Finally, $K = K_0$ is inferred by reversing the key expansion operations.

- **Round Reduction (RR)**

Many cryptographic algorithms, such as AES, are based on repetition of identical sequences of transformations, called rounds. A significant part of theses algorithms' strength against cryptanalysis is based on their repeated rounds. Any decrease on the number of rounds reduces their security. The Round Reduction belongs to the family of attacks by algorithm modification. For instance, suppose an attack by the opponent that makes a jump after the execution of few instructions from the first round at the beginning of algorithm to its end. So, the remaining encryption processes are skipped and the final cipher-text is the product of few algorithm processes that may reveal easily the key.

Principle of Round Reduction is based on decreasing the number of rounds in an algorithm in order to facilitate subsequent cryptanalysis. This method was first presented in the article [13]. It

illustrates that a transient glitch on the VCC may change the round counter value of a repetitive cipher. The opponent may break the algorithm execution at end of the first round. In this case, the cryptanalysis will be very fast and easy. Its complexity no more corresponds to the cryptanalysis of correct execution of entire 10 rounds for the reported algorithm. Application of laser attack aiming on such technique is also feasible [13].

- **Safe-Error Analysis (SEA)**

This analysis method searches for existence of any behavioral difference of a cryptographic circuit instead of faulty cipher-texts. A fault attack, may release an alarm or stop the operations. These signs of a behavioral difference in comparison with a normal execution may lead to find secrets from the circuits. The first SEA is presented in [14]. It consists in the injection of a fault by laser on a temporary register value and then observing the consequences on the output. One year later after the publication of [14], the authors in the article [15] reported a safe-error based attack by inducing a temporary random computational fault in addition to a temporary memory fault. Some other publications, such as in [16], tend to distinguish the two attacks, by considering the first method as a Memory or M Safe-Error that targets memory or register contents and the second one as a Computational or C Safe-Error Analysis focusing on the operations. However, in general the target of attacks against AES algorithms, such as DFA and RR, is mostly the temporary cipher-text, the round keys, the SubBytes table or the round counter.

2.5 Fault Injection for validating robustness of a design

Except for breaking system's security, fault attacks are recognized by scientists as a particularly attractive and valuable method for testing the robustness of hardware designs. Fault attacks that are used for validating purposes are usually referred as fault injection techniques. Fault injection can provide a method of assessing the dependability of a design under test. This is done by intentionally inserting faults into the system and monitoring system's reaction with respect to these faults. Fault injection allows validating robustness or dependability of a target system by providing:

- An understanding of the effects of real faults and, thus, of the related behavior of the target system in terms of functionality and performance.
- An evaluation of the efficacy of the fault tolerance techniques that are included into the target system and, therefore, a feedback for their enhancement and correction.
- Estimation on the failure coverage and latency (for example, timing, voltage level) of fault tolerant mechanisms.
- A forecasting method of the erroneous function of the target system, in terms with encompassing a measurement of the efficiency provided by the fault tolerance mechanisms.

- Exploration of the effects of different workloads (different input environments) in regards with the effectiveness of fault tolerant techniques.
- Identification of the weak spots in the design, as an example, parts of the system that because of a single fault could lead to severe consequences.
- Study of the system's behavior in the presence of faults, for example, propagation of fault effects between system components or the degree of fault isolation and determination of the coverage of a given set of tests.

With that said, engineers and designers use fault injection techniques to test the hardware systems. The next chapter offers an insight in several fault injection methodologies based on existing literature, as well as in the application of fault models that are used in combination with fault injection techniques. The objective is to deepen more into testing fault techniques concerning early levels in the design flow, such as RT and Gate Level and, finally focus on the main subject of the internship and, in extension, of the thesis. It is reminded that the early levels of abstraction may provide an effective source of information to lead towards the development of new countermeasures against malicious attacks on ICs.

Chapter 3: Fault Modeling and Fault Injection Methodologies

As mentioned before, the implementation of fault tolerance on digital systems dictates the testing of systems reliability. This is done by introducing faults into the system and inspecting system's response. The fault injection methodology under validation [1], proposes is a generalized RTL fault methodology, based on some other categories of fault types. That is why this chapter is focusing even more to fault modeling and fault injection techniques.

3.1 Definition of Fault

The definition is provided in accordance with the faults occurred on secure digital systems. A fault in a cryptographic system refers to an accidental or an intentional condition that causes the encryption or decryption process to deviate from its correct execution or result. In this case, the cryptographic system may act abnormally or the result of encryption or decryption may be incorrect, thus considered as faulty. A faulty execution or result is considered reproducible, if it occurs consistently under the same circumstances.

3.2 Different types of Faults

The present work concentrates on the hardware faults. Faults on the electronic circuits can be classified into three general categories, according to their persistence [17]:

- Provisional or transient faults: These faults are temporary or short-term. As the fault introduction is interrupted, the provisional faults disappear. So, after some time has elapsed, the chip recovers its normal execution without circuit reset. For instance, by heating a circuit (Temperature fault attack) faults are created that result to extended propagation times. The circuit resumes its correct functioning after temperature decreases.
- Permanent faults: The permanent faults are persistent but reversible. As the corrupted area is modified or changed by another part of the circuit or as the circuit is reset, these faults disappear. Thus, they are not destructive and don't damage the circuit. For instance, a fault injected on a SRAM cell persists until memory rewrite or circuit reset.
- Destructive faults: The interferences may create a perpetual defect on hardware. Once infected, such destructions affect the chip's behavior permanently. For example, a laser emission with a high energy level on a memory cell may permanently destroy some memory cells. In this case, the memory cells can no longer be rewritten or recovered by circuit reset.

3.2.1 Permanent faults

Effects of permanent faults are reversible. After a system reset or when the fault's stimulus is interrupted, the circuit will recover at its original behavior. There are two kinds of transient faults, and are explained below [11]:

- Single-event upsets (SEUs): It is interesting to note that this kind of attack was first noticed as an effect of cosmic rays during a space mission. Research then began on mechanisms of such faults into the circuits. SEUs consist in a cell's logical state flipping to a complementary state without any damage to the circuit. If the fault is produced in a system that recovers its original values after a reset, its effect is temporarily. SEUs can be created using focused laser beams.
- Multiple-event upsets (MEUs): They consist of several SEUs occurring simultaneously. So, MEUs can be considered as a generalization of SEUs. With the augmentation of integration density, the risk of generating such faults is increased.

3.2.2 Destructive faults

Destructive faults are due to an effect on the circuit that remains permanent and creates expanding faulty behavior or value. Due to their permanence on the circuit material, these types concern the highest level of abstraction in the design, indicating the semiconductor components. Different types of faults are included in this class, such as [11]:

- Single-event snap back faults (SESs): These kinds of faults are created by the self-sustained current by the parasitic bipolar transistor in channel n of MOS transistors. It seems that they do not occur in low supply voltage devices.
- Single-event latch-up faults (SELs): A latch-up consists in the activation of a parasitic thyristor structure formed in CMOS circuits. The transient current induced by a laser beam, for instance, may activate the parasitic thyristor resulting in a high current flow.

3.3 Definition of Fault Model

A fault model is an engineering model of something that could go wrong in the construction or operation of hardware. From the model, the designer can predict the consequences of a particular fault. In electronics, a fault model constitutes a description of how elements in a defective circuit behave. Usually, it is attached with several assumptions on fault manifestation and spreading. The goal of fault modeling is to model a high percentage of the physical defects that can occur in the device at the highest possible level of abstraction. In digital systems, high levels are described by the Gate and RTL net-lists.

3.4 Different Fault Models

The injected faults on the circuit can be described with different fault models, concerning bit level. The following fault models can be applied in an RTL fault injection analysis, during simulation. For the sake of describing some of these models accurately, we consider $T_1 = \{b_1, b_2, b_3, \dots, b_n\}$ as the initial values of an arbitrary set of targeted bits. Let $T_2 = \{b_1', b_2', b_3', \dots, b_n'\}$ be values of T_1 after a fault attack. Now, we review the effect of some existing fault models on the targeted set [11]:

- **Bit-flip** or **Bit inversion**: When the values of targeted bits are changed to their opposite values, we consider the fault type as bit-flip or bit inversion, if and only if:

$$\forall i: 0 \leq i \leq n, \qquad b_i' = 1 - b_i$$

- **Stuck-At**: In this fault model, the targeted bits are set permanently to their previous value. Therefore, even if new values must be affected to the targeted bits, the memory write operation cannot change them. This effect is usually considered as a destructive fault due to a wire, gate or memory cell damage, but it might be a permanent fault that disappears after a circuit reset.
  The fault model is considered as stuck-at 0, if and only if:

$$\forall i: 0 \leq i \leq n, \qquad b_i' = b_i = 0$$

  The fault model is considered as stuck-at 1, if and only if:

$$\forall i: 0 \leq i \leq n, \qquad b_i' = b_i = 1$$

  In this category of faults, the values of targeted bits are usually unknown to the opponent before and after the attack. A stuck-at fault has a noticeable effect only when it must be rewritten to its opposite value. At this point, it may create a change in the system behavior or results.
- **Random**: When the value of at least one of targeted bits is changed, but the value changes are random. In other words, the fault model is random, if and only if:

$$\forall i: 0 \leq i \leq n, \qquad b_i' \in \{0,1\}$$

- **Set** or **Reset**: In this fault model, the targeted bits are set or reset to whatever is their previous value. The fault model is considered as set, if and only if:

$$\forall i: 0 \leq i \leq n, \qquad b_i' = 1$$

  Otherwise the fault is considered as reset, if and only if:

$$\forall i: 0 \leq i \leq n, \qquad b_i' = 0$$

Between these fault models, the random faults are usually considered to be the most realistic. Such simple fault models, describing the perturbation of bits in hardware are usually implied in more complicated fault models, such as those studying fault manifestation in early levels of design flow (Gate and RT level)

3.5 Fault Modeling at RTL and Gate Level

Fault models constitute a representation of fault impact on a small part of the electronic device. Based on such models, scientists develop large-scale fault injection methodologies to test the robustness of IC designs. The need for early evaluation of the IC design flow with respect to fault-based attacks has led to the development of fault injection models at a high level of abstraction, indicating RTL and Gate Level. Logic synthesis transforms the RTL description into an optimized technology-specific hardware description in the form of Gate Level net-list, without altering design's original functionality. This sub-chapter makes a brief reference on previous work concerning fault models that are applied on earlier stages of manufacture, such as RTL and Gate Level. The fault injection models that are mentioned here, make use of statistical and probabilistic methods, as well as simulation techniques. The current thesis is attempting to validate a fault injection methodology that takes place at the RTL [1].

Gate Level is widely accepted as a good compromise between abstraction level and the ability to represent most of the defects in designs under test. By representing the device under test (DUT) as a gate-level model, fault injection models have the ability to increase testing efficiency in the design flow. As stated in the article [18], testing has been historically performed using gate-level fault models. Much research has focused on gate level modeling of attacks. In [27], the author mentions several Gate-level fault models and explain how error properties induced by a fault attack in a logic circuit can be modeled in terms of those models. It is a fact that faults in Gate-level can be further modeled, according to the way they occur. For instance, single (or multiple) stuck-at faults make the assumption that one line in a gate is (or multiple lines in many gates are) faulty and that fault is permanent as opposed to transient. Stuck-open model constitutes another gate fault model that assumes a single physical line in the circuit is broken and the resulting open node is not tied to either V_{DD} or GND. Finally, fault models related with the delay of signal propagation and the short-circuiting between two or more lines on the circuit are presented.

In [19], the authors focus on the evaluation of circuit reliability under probabilistic methods that can capture both soft errors, such as radiation-related errors, and spatially-uniform manufacturing defects. This task can be used by synthesis procedures to select more reliable circuits and to estimate yield for electronic nanotechnologies where high defect density is expected. In their work, they propose a matrix-based formalism to compute the error probability of the whole testing circuit based on probabilities of specific gate errors. This formalism is related to that of quantum circuits, but also it is revealed that the numerical computation of error probabilities can apply on larger circuits.

In other articles sampling techniques are used as fault injection modeling. Sampling techniques, where a randomly selected subset of faults is simulated to estimate the fault coverage, can reduce the performance penalty of gate-level fault simulation. The author in [20] introduced the sampling technique to gate-level fault simulation to decide whether or not the fault coverage of a given test exceeds a given bound. This technique was elaborated by the author in [21] to provide upper and lower bounds for the coverage. He also proposed a method that uses a fault sample of a fixed size. The estimation of fault coverage by simulating only a fraction of gate-level faults requires only a fraction of time and resources required for the complete gate-level fault simulation. Similar approaches based on statistical sampling techniques are proposed by McNamer et al. [22] and Daehn [23]. Even though the fault-sampling technique reduces the size of the fault-list used for simulation, it requires a complete gate-level fault-list, meaning all the combinations of faults occurred on gates and, therefore, cannot be used prior to logic synthesis. Post-synthesis findings of test generation and fault simulation efforts are too late in the design cycle to be utilized for architectural changes to improve system resilience. It is, therefore, desirable to develop the fault injection models at a higher level of abstraction than the Gate level.

Mao and Gulati [24] proposed an RTL fault model and a fault injection methodology using simulation. The fault model used is the single stuck-at fault for each bit of all variables in the RTL net-list. The RTL fault simulator they developed, supports RTL testability analysis on circuit designs. They were able to generate quantitative RTL fault coverage and provide information for design modifications, leading to the testability at the RT level. Their approach also required to run fault simulation twice (first in an optimistic and then in a pessimistic mode) and to use the average of the results to reduce the difference between the RTL and the gate-level fault coverages. Their work showed that RTL fault coverage results in the improvement of fault coverage at the Gate Level. Nevertheless, the RT level description is at a higher level of abstraction and may not cover all the gate level faults.

Hayne and Johnson [25] developed a fault model based on finding an abstraction of the industry standard single-stuck-line faults in the behavioral domain. This fault model was developed such that for every possible gate-level fault in the circuit there is a corresponding faulty RTL circuit. The gate-level net-list changes drastically with every synthesis run and there are numerous possible structural implementations for the RTL code. The modeling of all possible gate-level failure mechanisms at RT level is clearly inefficient and one can use only limited cases.

In general, Gate Level fault injection methodologies increase test-generation efficiency, thus they are preferable for validation in the design flow, but their application is not an easy thing. There are many parameters need to be calculated in such models, for instance the delays in the input nets of gates, concerning the propagation of signal, the assumption that open-circuit faults (faults that deal with the state of the wires) are excluded and more. For a fast, accurate and efficient fault injection model, RTL proves to be a good solution. RTL net-list serves as a common database for various post-synthesis steps, such as timing simulation, placement, routing, static timing analysis, etc. As described in [25], previous research efforts in the RTL fault modeling area have taken the approach of modifying RTL code to model all gate-level failure mechanisms. These efforts have not been successful, primarily due to the fact that the gate-level net-list changes drastically with every synthesis iteration, creating many distinct

gate-level fault lists. It is impossible to model all the gate faults of every possible net-list at the RT level. Instead, in this thesis, a theoretical RTL fault injection algorithm is developed such that the RTL fault list of a design becomes a representative sample of the Gate Level fault list.

While research results in the area of high-level synthesis show great promise, the proposed techniques are mostly applicable to data-flow intensive designs. More work is needed before high-level test synthesis can be used in the mainstream ASIC design arena. Most of the VLSI design work is still done at the RT level while high-level test synthesis aims at facilitating testing for behavioral designs [25]. Though high-level test synthesis holds great promise for futuristic behavioral level designs, the fundamental problem of the lack of an RTL fault model for test generation and evaluation needs to be solved for the contemporary mainstream RT level designs.

Chapter 4: State-of-the-art Laser Fault Modeling at RTL

4.1 General Aspects

In this chapter the multiple fault injection methodology, already published in the literature [1] is summarized. The work of the current thesis is based on this particular methodology, concerning the modeling of laser attacks on ICs at the RT Level and in this chapter it will be elaborated. Lasers, as mentioned in Chapter 2, provide a very effective means to perform fault injection attacks on ICs, mainly because of their high precision locality, accurate timing and high occurrence probability. In the case of faults caused by a laser, the fault analysis should deal efficiently with the added complexity dictated by the laser characteristics. The complexity rises from the fact that a laser attack, especially in recent manufacturing technologies (e.g. 45nm, 32nm, 28nm), provides to the attacker the flexibility of an excellent controllability over location and timing. Stuck-at and bit-flip models can be used to model the effects of a laser on an integrated circuit [1]. However, single bit flipping in flip flops does not describe the phenomenon accurately and, that it the reason multiple bit flipping fault models need to be used for the fault injection methodology.

Until article [1], there did not exist any other RTL Laser Fault Model. In multiple different approaches, 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, cannot be known 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 our analysis [1]. Also, the complexity of such 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 in a later step of simulation and emulation analysis, which make the simplification of the models a necessary step.

4.2 Cone Partitioning

4.2.1 Definition of Logic Cone

RTL fault methodology dictates the partitioning of the whole elaborated (non-optimized) RTL net-list of a design. The partitioning is done using as basic block the logic cone. A logic cone is defined as the set of all the nets and combinational instances that reside in the fan-in network of the input net of a flip flop. Each cone corresponds to a single flip flop. Fig. 4.1 depicts the image of a simple logic cone, starting from a flip flop, as the top of the cone and ending to other flip flop or primary inputs, at the boundary of the cone. The last elements are not included in the cone. Cone partitioning offers a flexible way of determining the effect and propagation of fault injection over a particular section of the RTL design.

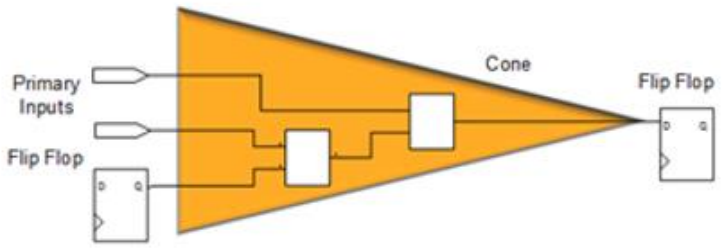

Figure 4.1: A logic Cone

4.2.2 Fault types

There are two types of intrusive faults that RTL Cone partitioning takes into consideration. The first type concerns faults that are directly injected into one or more flip flops. This happens when a localized radiation assault (either high energy particles or a laser) aims straight on these memory cells. The second type of faults has to do with faults that occur in the combinational elements (gates) of the circuit, during a localized attack on these elements. In that case, the faults are considered to propagate towards the flip flops that include the affected elements in their fan-in network (indirect attack). There is also the possibility that the fault will fade out while propagating. This can be formalized by stating that the targeted combinational elements belong to the input logic cone of the potentially affected flip flop. In other words, faults affect only the sequential logic, as the combinational elements are used as fault transmission means towards the flip flops. Figure 4.2 shows these two different kinds of faults.

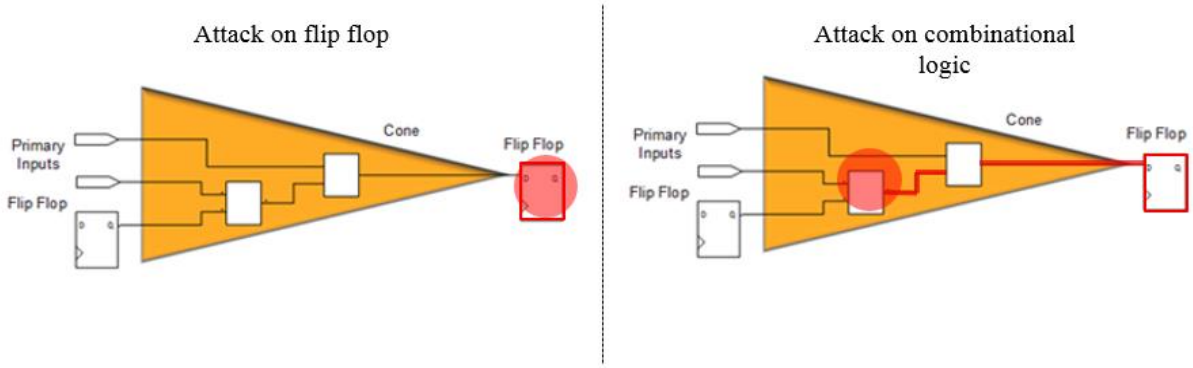

Figure 4.2: Laser attack on flip flop (Direct) / on combinational logic (Indirect)

In addition, it is able to model both attack on combinational logic and attack on sequential logic with multiple bit flipping on the flip flops of an RTL design, during simulation steps. Therefore, by injecting bit flips in one or more specific flip flops, the evaluation can cover all the faults that affect any combination of gates belonging to the corresponding cones.

4.2.3 Locality approach

A realistic laser attack offers the ability of exact controllability over space and locality of the attack. As an illustration, the more precise the laser spot is, the smaller area of the layout it affects, thus less elements are injected with faults. An advantage that the RTL fault model offers is the capability to model the spatial characteristics of the attack in regards with the controllability over the location of fault injection. This capability of the model will aid to define a measure of how successful an attack can be, in terms of the controllability over space.

4.2.4 Assumptions

There are two main assumptions stated by the RTL Cone methodology:

-**Assumption 1**: Functional relations on RTL description can be used to model fault propagation

With that said, it is stated that even after placement and routing of the design, the functional relations among the components in the RTL net-list will remain on the components of the layout. In case a localized spot affects a certain set of flip flops (directly or indirectly), then this particular set can be identified and marked for bit-flipping on the RTL analysis. On the contrary, flip-flops that reside outside the affected area on the layout do not need to be injected with faults on the RTL, unless their corresponding RTL logic cones contain elements which also exist in the cones which are considered affected by the attack. This statements get clearer in the next assumption.

-**Assumption 2**: All the elements of a cone are impacted by a laser shot at the same time

This assumption states that if a localized attack on the layout affects a flip flop, then the corresponding cone in the RTL description is considered faulty as a whole. That means all the elements of the cone are affected by the attack. Therefore, according to the fan-out network of each combinational element, the fault is likely to propagate towards as many RTL flip flops as found in the fan-out networks. A simple example is demonstrated in order for the reader to comprehend the methodology. In Fig. 4.3 we get a notion of RTL fault modeling. According to the previous assumptions, there exist certain RTL cones that are considered as affected by the attack. Fig. 4.3 presents the Cones 1, 2, 3 and 4 that are bounded by a starting net, connected to a flip flop (Father Flip Flop) and expand backwards, from the outputs towards the inputs, up to either flip flops or primary inputs of the circuit. These cones constitute a simplified form of the RTL partitioning of the random design. We assume that the laser spot affects only Cone 1, as it is depicted by the red coloring. This means that the spot covers either flip flop 1 (direct attack) or any of the gates i1, i2 and i3 that reside in the cone (indirect attack) or even all of them. Intersection takes place between cone 1 and cone 2, as they both include gate i3. Thus, it can be fairly assumed that fault will propagate and be stored into either flip flop 1 or flip flop 2 or both. In other words, cones 1 and 2 are candidates to be the final recipients of fault. Of course, there is also the scenario that none of the flip flops gets affected. It is certain that fault will not propagate and be stored into the flip flops 3 and 4, as their corresponding cones do not intersect with the affected cone 1. By combining the locality of a laser attack with the cone partitioning, we are able to extract which flip flops will be potentially affected by a given attack. In this example, the set of flip flops 1 and 2 are the potential recipients of fault propagation.

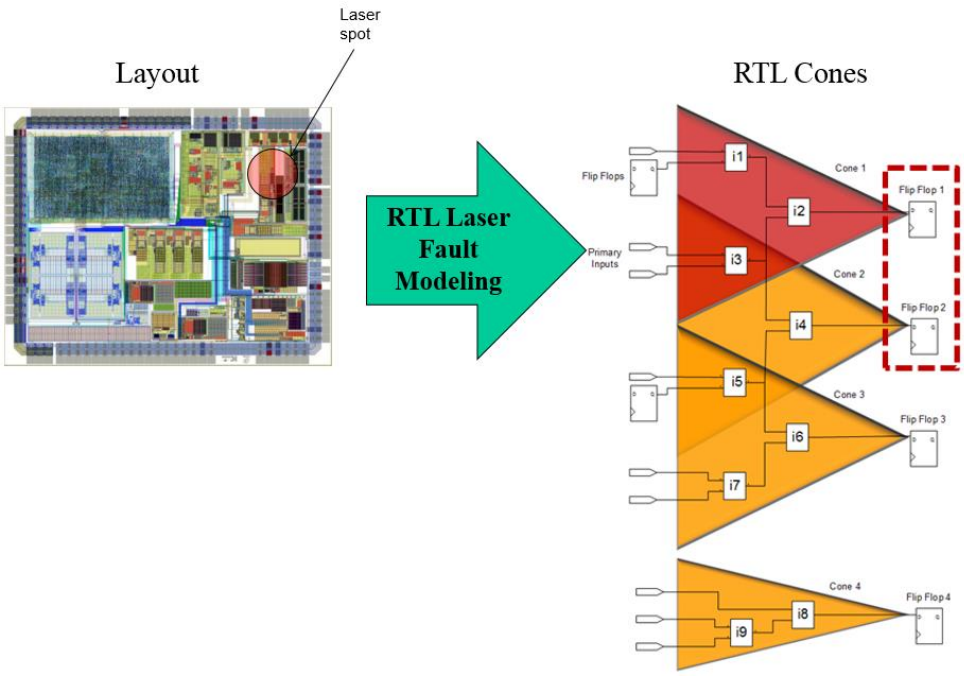

Figure 4.3: Example of a layout spot in RTL Laser Fault Modeling

The potential recipients of faults are ultimately indicated by the intersection of cones. For each cone (flip flop) a set of cones is extracted, with respect to the intersection of this cone with others in the RTL net-list. The RTL cones of different designs are studied and analyzed with the scope to identify correlations among the partitions of RTL net-list. Summing up, RTL fault methodology states that functional relations in RTL can be used to determine which cones are more likely to be simultaneously affected by a laser spot. Simulation and emulation efforts of fault injection on the circuit will aid to monitor the impact of fault attacks in a circuit.

4.3 Limitations of the Method

Logic cone partitioning is an efficient technique that applies in the RTL net-list. It defines attributes such as locality and propagation of laser-induced faults and can make use of the multiple faults with e.g. bit flipping fault or other models that are suitable for representing a laser attack, during simulation efforts. Nevertheless, as a technique it presents some limitations, such as:

- The fact that a laser spot on the IC layout may affect logic corresponding to RTL cones that do not intersect with each other. This case is not taken into account by the methodology, as RTL intersecting sets miss to represent the actual impact of the laser attack.

- The model takes into consideration the area covered by the laser spot. However, a realistic laser attack can potentially affect even the adjacent region around the laser spot, leading to the influence of even more logic components.

Chapter 5: Layout Extraction Tool using OpenAccess™

After having summarized the main methodology described in [1], Chapter 5 proceeds to give an overview of the practical requirements throughout the internship. In this chapter, the design of the layout extraction tool is presented. Its implementation was based on a C++ API specialized for integrated circuits interface, called OpenAccess. In other words, several virtual partitions are applied onto the layouts in order to analyze the effect of localized or realistic laser attacks, by focusing on the elements affected, in a deterministic way. This chapter describes how this virtual partitioning is implemented and what are the main attributes of the tool that are taken into consideration for the validation process, described in Chapter 6. As an introduction, some brief description on the OpenAccess™ software and its C++ programming style is provided.

5.1 OpenAccess™ EDA Tool

5.1.1 Overview

OpenAccess™ is an advanced EDA database designed to enable interoperability among different IC design tools through an open industry-standard data access interface API, and a reference implementation [26]. It is released from Silicon Integration Initiative, an EDA/electronics industry consortium focused on electronic infrastructure standards and based on community contributions to enhance chip design flows. What OpenAccess™ manages to do is to span the EDA design space. It can be used to manage designs from post-synthesis net-lists to tape-out (last stage of manufacture).

Today's design environments are a complicated mix of design tools containing different applications and associated databases, with incompatible and difficult to analyze file formats and syntaxes. IC CAD engineers spend many hours integrating the designs with thousands of lines of translator code and the resulting flows are fragile and error-prone. As well, they are inefficient and result in longer IC design cycle times [26]. OpenAccess™ comes to make things easier, by giving a solution to the latest problems of the IC designers and by allowing the extensive analysis, research and consideration of integrated circuit issues.

OpenAccess™ provides advantages to developers of design flows and EDA tools. Nowadays, all design flows use the following file formats:

- Verilog
- LEF
- DEF
- GDSII
- SPEF

These files formats represent ports, architecture, libraries, process technology, specifications and other design attributes. Such files were encountered during the internship and their usage is explained

in the subsequent chapter. Among different EDA tools, these various data files representing a design were usually incomplete and inconsistent. Each tool needed different design information file to analyze and this fact forced designers to depend on many different software EDA packages, each one translating in an exclusive format, compatible only for this particular software. Therefore, overall translation flow was inefficient, and could often result in misinterpretations or information loss, due to ambiguities in the differing format specifications, switching from one EDA tool to another. OpenAccess™ provides the solution by integrating the management of all design data formats. First, the OpenAccess™ design flow model is more complete, unambiguous and consistent than most of the previously used EDA tools. It manages to convert all different formats into a single one, by parsing and building all data attributes of the designs on this integrated format. In addition, an OpenAccess™ database for a design can be read by applications developed through the C++ API much more efficiently, allowing a convenient managing and processing in the overall design flow.

Eventually, OpenAccess™ outdates previous EDA tools. It provides much smoother integration for a design flow than it was previously possible with tools from multiple sources. The most efficient approach is being able to develop applications to operate directly on designs having the particular OpenAccess™ data model. The development of the OpenAccess™ programming architecture was driven by a modern, object-oriented design methodology, to leverage fundamental engineering principles of design complexity such as hierarchy, abstraction, incrementalism, and iteration [26]. Within the context of a strongly-typed classes utility, the OpenAccess™ API provides the necessary means for manipulating database information in ways convenient for design activities, across a wide range of user-defined applications.

OpenAccess™, besides providing the means for a proper design translation flow, includes an object-oriented API written in C++. It was built from the beginning as a source for open community usage [26]. Using the C++ programming language, it ensures a strongly typed interface, preventing many programming errors. Consistency was emphasized during the design of the API, in order to make it easier to conceive and handle. The OpenAccess™ programming model covers a large portion of this EDA tool. It can handle both logical and physical design hierarchies and connectivity, as well as an occurrence model which relates the two. It includes custom geometry, routing topology and floor planning information, parameterized cells and technology node information. Various mechanisms are supported, with the capability of filtering for different types of usage. The API supports efficient searching utilities such as Region Query, and name mapping capabilities. Actually, within the context of the internship, layout extraction tool was firmly based on such searching techniques. The Reference Implementation has been tuned for improved performance and memory efficiency. Finally, the API supports defining extensions to most built-in objects as well as new kinds of objects. The extension mechanism is highly efficient, and can be used by developers to extend the database to support their application's needs.

Nowadays, there are many companies whose actual work depends highly on OpenAccess™. In addition, many university laboratories have at their disposal this powerful EDA tool. In summary, the

reasons for an EDA company or a research laboratory to include in its technical arsenal this powerful tool are many. Some of them are presented as follows [26]:

- Enables tools (EDA vendor products, proprietary tools, university research) to be integrated to form a complete solution
- Provides true interoperability and concurrency
- Eliminates costly/lossy data exchange
- Allows customer to dictate the design flow

5.1.2 Translation Flow – Layout Import

It has been stated multiple times that, OpenAccess™ offers the attribute to integrate all data file formats that describe a design into on single data type, with the extension "OA". This is done using certain executables OpenAccess™ provides to users, called OA translators. All of the OpenAccess™ translators share certain common functionalities and use a common subset of command-line options to find and process design libraries and determine the design management system that is used during a translation. All OpenAccess™ translators use a specific file, called library definition file (lib.defs), to find the technology libraries that are available and to record new library definitions if new libraries are created. Each translation step updates this particular file. OpenAccess™ translators can handle design data files, such as Verilog, LEF, DEF, GDSII and SPEF files. These type of files that describe a design are presented as follows:

- **Verilog**

Verilog is the popular hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the RTL level of abstraction. Such Verilog file is usually the output of synthesis of the RTL description (VHDL, Verilog, SystemVerilog). It usually consists of one big module filled with gates and registers, connected with wires. This is often referred to as flattened design, because all the individual modules from the original RTL design have been flattened into one big module and all hierarchical information is gone. In our analysis, all employed designs were flattened for simplicity in the translation process.

- **LEF / DEF**

For abstracting circuit layout's topological information, the necessary files are LEF (Library Exchange Format) and DEF (Design Exchange Format) [30]. The LEF/DEF files are used to describe an IC layout in an efficient electronic form. The first one defines the geometry (size and form) of each element of the technological library, while the second defines the position of each gate within the circuit,

including the net-list and design constraints. Many designs may be described with more than one LEF files. However in our analysis, designs included two LEF files:

- Technology LEF file, including the information about the technology library and
- Main LEF file, including all cell information. Main file may consist of multiple LEF files, according to how many different types of gates, flip-flops, and instances in general, are used in the design.

Technology library (standard cell library) used for our tested designs is Nangate 45nm, provided by FreePDK™. DEF file is strictly unique for each design and produced after Placement and Routing of the design layout.

- **GDSII**

GDSII stream format, common acronym GDSII, is a database file format which is the de facto industry standard for data exchange of integrated circuits or IC layout artwork. It is a file format representing planar geometric shapes, text labels, and other information about the layout in hierarchical form. The data can be used to reconstruct all or part of the artwork to be used in sharing layouts, transferring artwork between different tools, or creating photomasks. GDSII files are usually the final output product of the IC design cycle and are given to IC foundries for IC fabrication.

- **SPEF**

Standard Parasitic Exchange Format (SPEF) is a standard for representing parasitic data of wires in a chip in ASCII format. Resistance, capacitance and inductance of wires in a chip are known as parasitic data. SPEF, though, does not include inductances. SPEF is used for delay calculation and ensuring signal integrity of a chip which eventually determines its speed of operation. SPEF is the most popular specification for parasitic exchange between different tools of the EDA domain during any stage of design. SPEF is usually extracted after routing in Place and Route stage. This file contains the R (Resistance) and C (Capacitance) parameters depending on the placement of cells and the routing among them.

Translation flow may include all the upper files, however for a typical conversion flow, the way it was implemented during the internship, the main files needed to translate design data into OA format are only Verilog, LEF and DEF. The important factor to use OpenAccess™ for translation flow is to eliminate all dependencies across different translating tools, but in order to import compatible designs into an OpenAccess™ flow several translation steps are required. These steps were followed for each design tested during our analysis, need to precede layout partitioning procedure, which is described later. The following information presents the steps for translating layout data to OA format. For a typical flattened design, as those used for spot partitioning during our research, the flow requires the following

files (example quoted from the analysis of the design layout “AES Parity” using technology library Nangate 45nm):

- LEF files: NangateOpenCellLibrary.tech.lef and NangateOpenCellLibrary.lef
- A Verilog file: aes_parity.v
- A DEF file: aes_parity.def

The procedure has to follow strictly the order of the following three translation phases:

1) Use *lef2oa* to create technology data and the reference libraries. This is done by executing the following commands:

```
lef2oa -lib techLib -lef NangateOpenCellLibrary.tech.lef
lef2oa -lib techLib -lef NangateOpenCellLibrary.lef
```

2) Use *verilog2oa* to import the logical description of the design. The command that implements the conversion from Verilog to OA is:

```
verilog2oa -lib stdLib -verilog aes_parity.v
```

3) Use *def2oa* to annotate the logical description of the design with the physical implementation. This is done using the command:

```
def2oa -lib techLib -def aes_parity.def
```

The outcome of translation process outputs the following:

Figure 5.1

These files constitute the format which OpenAccess™ is compatible with. They store the IC information on the disk and is used to exchange IC data among applications. The OA format :

The Technology directory with its various IC layer directories. Technology directory contains the technology parameters that correspond to a particular design methodology and fabrication technology. These parameters are typically contained in a technology database. All these types of data are stored in this single directory.

The Library directory with cells that are IC structure directories. Cells describe the different types of gates and registers, encountered on the layout. Each IC structure directory is a cell with at least a single view file. A view is referenced as a layout.

The lib.defs file is the top-level file including directory names that are the OA libraries. It contains the paths of Technology directories and of the Library directory, as well as some additional design information. The lib.defs file is an ASCII file that holds a symbolic name and a path for each library. The library paths can be either relative or absolute. Relative paths are in relation to the location of the library definition file. OpenAccess™ identifies the imported design by first using lib.defs and then having access to the libraries, which is a list of the logical names and paths for the libraries in a design project.

Once the upper directories and file are produced successfully by the translation procedure, testing design has smoothly been converted in a format that OpenAccess C++ API can recognize and handle properly. Layout has become a compatible input for any application developed on the API. The next sub-chapter deepens into the coding aspects of the OpenAccess, by focusing on the main classes used for the implementation of the Layout Extraction tool.

5.1.3 Design of Layout Extraction Tool – C++ Classes

OpenAccess™ C++ API contains countless classes built-in to facilitate the IC interface and allow users to endorse design methods or examine electronic components such as gates, registers, metal routing tracks, vias and electrical pins on the layouts [26]. The most used C++ data types that contributed to the design of the Layout Extraction tool are presented as follows:

oaDesign: Using this class, it is feasible to import the OA-translated design into the C++ implementation. It is the main description of the design under test. It contains the database that hold all the design data describing elements of the design. All net-lists, schematics, layouts and other design representations exist as a set of oaDesign in OpenAccess™. It is also container for the connectivity, geometry, hierarchy and floor-planning information about a design. Each oaDesign is identified by three parameters: library, cell, and view. These entities were extracted from translation flow. A demonstration of loading a design in the API, using special C++ commands, is shown below:

```
oaDesignInit();
oaDesign *design = oaDesign::open(libName, cellName, viewName);
```

At first, initialization of oaDesign is implemented in order to be imported for the tool to accept the new layout, and close any other open design in the program. Then, a pointer is created with reference

on the imported design. Method “open” allows the correct import of the design, as soon as the necessary parameters are inserted: library name, referring to Library directory of the translation output, the cell name, which is the top module (again existing in Library directory), and the view name (the layout schematic of the top module). Applying the open function on the previous example on AES Parity, the command is modified as shown:

```
oaDesignInit();
oaDesign *design = oaDesign::open(Stdlib, aes_parity, layout);
```

Thus, AES Parity layout has been properly implemented into API Reference Implementation.

oaInst: The oaInst class is an abstract base class used to represent and manage instances. An instance in OpenAccess™ constitutes the inclusion of one electronic component as a part of the contents of the layout. The design containing the instance is considered the parent design and the design that is included is the master of the instance. Entities such as metal, vias, gates, flip flops are considered as instances. The most common instances on the layout are presented below in Figure 5.2

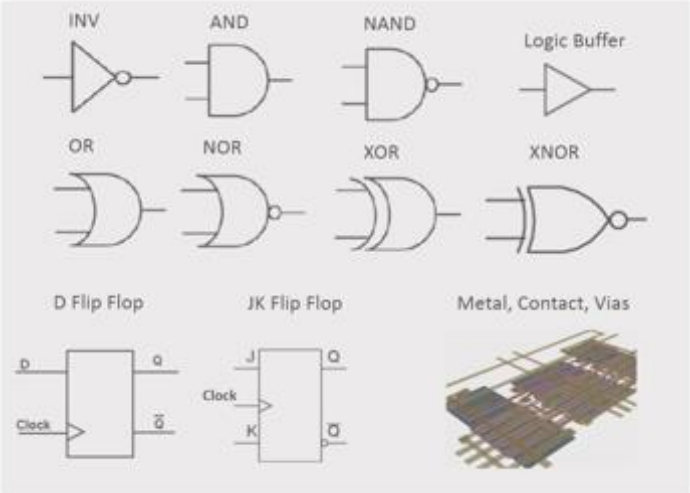

Figure 5.2: Instances

The following code block demonstrates an example of the usage of the important class oaInst :

```
oaInst *inst;
inst->getCellName(oaNs, name) ;
inst->getName(oaNs, name);
inst->getBBox(bbox);
```

The first command assigns a pointer (named inst) that refers to a specific instance of the design. The user is able to extract useful information for that specific instance. This is done using the rest of the commands. Attributes such as type, identifier, location and dimensions of the instance are a sample of what information can be extracted from functions written for oaInst. Function getCellName is used to reveal the identity of the instance, for example if it is an AND, OR, XOR or other gate, if it a D Flip

Flop or JK Flip Flop or another register in general. Function getName returns the ID of the instance. ID is a unique code name that specifies one and only entity. Finally, as shown in the example, function getBBox returns the orthogonal coordinates of the instance. It is noted that for OpenAccess every instance's spatial capacity is enclosed in a two-dimensional rectangular region. This approach facilitates the definition of location coordinates on the layout in only four values. Method getBBox inherits its attributes from oaBox class, which is described in the next paragraph. So, what getBBox does is to return the lower left horizontal and vertical coordinates, as well as upper right ones. Being aware of the exact coordinates of an instance on the layout is very important and, as it is explained later, it constitutes a basic structure for forming the layout extraction tool.

oaBox: The oaBox class implements a two-dimensional rectangular region with integer coordinates. This class is used throughout the database to represent the bounding boxes of instances. It is the main attribute that permits the localization of a potential laser attack on the layout. The area covering on the IC surface is examined in terms of a hypothetical laser impact. In other words, bounding box is the virtual spot on the layout. Figure 5.3 shows the notion of the bounding box.

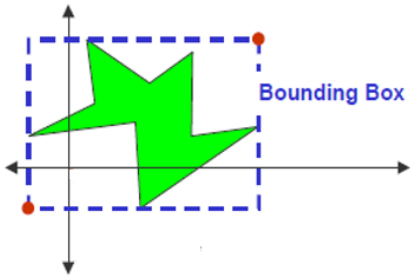

Figure 5.3: Bounding box

As it is shown, the box includes the leftmost and rightmost point of the instance, as well as the lowest and highest point. The coordinates are in integer data base units (DB) which get mapped to defined distances by settings in the technology data base. When the data base units are set to 1000 for OpenAccess™ format, the realistic value is 1000 nanometers or 1 micrometer. DEF file acts as the regulator of the corresponding DB unit set for each design. An example is presented using the following code lines:

```
bbox = oaBox(0, 0, 67000 , 46000);
printBBox(bbox);
```

The brackets of oaBox contain the left, bottom, right and top side location of the box, respectively. Function printBBox simply prints the dimensions of the box. The values in the brackets are strictly be in DB units, so in case DEF file dictates that data base units are set to 2000 per micron, the real dimensions of the box, having as reference point the origin (0, 0) are (0, 0, 33.5, 23) in micrometers. Figure 5.4 presents the defined bounding box, using DB units as well as realistic measurement.

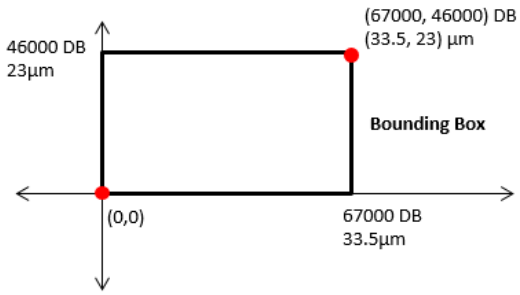

Figure 5.4: Defined bounding box (virtual spot)

oaRegionQuery: The oaRegionQuery class is an abstract class that is the parent class to query classes included in the C++ API [27]. These classes implement a hierarchical query for figures within a specified rectangular region of a design hierarchy. oaRegionQuery is used to drive the graphical display of design hierarchies as well as find the set of objects that are neighbors to a given object for analysis. Starting from the top module of the design (start Level) and reaching to the last module (stop Level), oaRegionQuery scans the region and returns the set of objects that exist below a certain region of the IC layout.

An oaRegionQuery object descends through a design hierarchy from the top design with which it is constructed, producing all objects of a specified type in the specified query region. In other words, it will not process the contents of instances unless they have the proper type and are included in the designated area. This happens regardless of the specified start Level and stop Level. If the caller wants to ensure that a specified number of levels of design hierarchy are processed by oaRegionQuery, the caller can precede the usage of functions, contained in other classes that are not specified here because they were not used in the context of the internship. For flattened designs, where start Level is assigned as 0 and stop Level as 1, application of oaRegionQuery is simpler [27].

Applications use oaRegionQuery by creating their own class that derives from one of the oaRegionQuery subclasses, then implementing functions in their class for virtual functions declared in either the base or the derived oaRegionQuery class. These user-implemented functions will be called by the database to hand off the figures found in a specified region. A region query is initiated by a query function in the figure-specific RegionQuery class. According to which figures user aims to extract, figure-specific queries can be initialized through the RegionQuery class. Users can create queries specialized in tracking design layers, vias, registers or instances in general. For the scope of our analysis, an instance query class was implemented and its functionality is explained as follows.

InstQuery: The InstQuery class is an implementation of oaRegionQuery which initializes queries in specified areas of the layout and extracts all instances that overlap the areas in a fast and efficient way [27]. This mechanism is proved to be very useful for our analysis, as our study adopts this capability and by inserting certain attributes, results in the development of a layout extraction tool approaching the laser attack spot form. Parameters are provided in order to allow our application to control how the query

is performed. For the needs of our research, code lines were written so as to create an InstQuery class containing the function query, specialized for identifying and returning all existing instances. The initialization and calling of the function is done using the following commands:

```
opnInstQuery instQuery;
instQuery.query(Design, Region, FilterSize, StartLevel, StopLevel);
```

Relating to the parameters of the query method, the first parameter concerns the design that has been imported, where the instance extraction will take place. The second parameter Region refers to the bounding box (virtual spot), thus determines the exact rectangular region on the layout which is under examination. The third parameter is a filtering threshold applied on the instances of the specified area. According to its integer value, it permits the extraction of all instances whose area is equal to or larger than the value. As far as there is no restriction on the area covered by a single instance, the value of this parameter is set to 0. Finally, the fourth and fifth parameters control respectively the start Level and stop Level hierarchy of the design. As long as the designs in our analysis are properly flattened, there is only one level of hierarchy, so the parameters are assigned 0 and 1 respectively. A visual demonstration of querying method is provided in Figure 5.5.

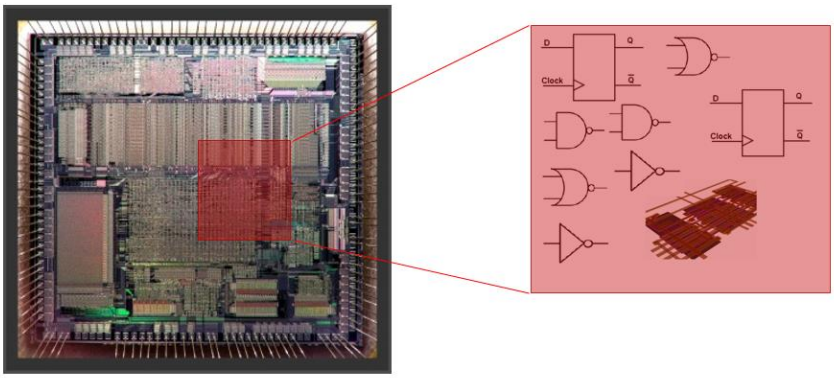

Figure 5.5: Instance query in a specified bounding box

For the needs of developing an efficient layout extraction tool that will allow the inspection of certain areas on the IC surface, Instance query is generalized to include all layout area by covering it with small partitions. These partitions are considered a good representation of potential laser spots on the IC layout and serve well for the outcome of the investigation. What really matters for our methodology is not the shape and dimensions of the partition spot, but the ability to extract every instance that is included partially or as a whole in a given area. Besides, for the laser spot of diameter 1μm, the partitioning tool creates a square partition spot, thus area coverage on the layout is bigger, leading to worst-case scenario of attack. For our methodology, it is needed to scan the whole layout surface with small range partitions. The size of a partition is controlled by the region parameter in query function. In that way we are able to scan the whole surface of the any IC layout in square partitions and

extract all the entities we are interested to. Thus, this technique can be used to make a laser spot approach that specifies locally an attack on the IC layout. In order for the reader to realize the technique applied in C++ code, a simplified example of spot partitioning is demonstrated.

```
for(j=0; j<Layout_Coordinate_Y; j=j+Bbox_step)
{
    for(i=0; i<Layout_Coordinate_X; i=i+Bbox_step)
    {
        bbox = oaBox(i, j, Bbox_length_side+i, Bbox_length_side+j);
        instQuery.query(aes_parity, bbox,0,0,1);
    }
}
```

The double-enfolded commands set the configuration of one partition on the layout, in the same manner as previously described. By adjusting the parameters Bbox_step and Bbox_length_side in desirable values, whole layout coverage in small partitions can be accomplished. Parameters such as Bbox_length_side, Bbox_step control the length and the rate of position change of the square spot respectively and contribute to the completeness of our experiments, as they are explained in the next sub-chapter, where all important attributes of the layout extraction (or layout spot partitioning) tool are presented.

5.2 Spot Partitioning Attributes

Many IC layouts were designed in order to test and validate the Fault Injection methodology described in this thesis. It has previously been shown that OpenAccess™ facilitates the inspection and monitoring of the electronic components that reside on IC Layouts. During the internship, this attribute was used and adapted with respect to the purpose of verifying the assumptions of the Cone Methodology. A platform was implemented, allowing the examination of the elements affected by a potential "localized spot". Appropriate code was added so as to offer new parameters for the configuration of the spot partitioning platform. These parameters are described as follows:

- **Spot length side**

The area covered by the spot is strictly a square shape. The tool allows the user to define the dimensions of this area. Therefore, large spots cover large square areas, resulting to the extraction of multiple instances from the layout. The number of elements extracted differs and highly depends not only on spot's spatial parameters, but also on its locality on the layout. Small bounding boxes usually cover a slight number of instances, that results in the extraction of a few flip flops. There are also cases where the spots do not hit any logic element. The input of spot length side is strictly in DB units. Selection of spot length side depends, absolutely, on the technology of the design. Layouts fabricated with semiconductor manufacturing process of 350nm require larger spots for scanning the area. Our work has focused mainly on the manufacturing technology of 45nm, therefore large spots were inaccurate to model a laser attack and, hence, unnecessary.

- **Step length**

The developed platform attempts to perform a cartography of the chip's layout. The parameter allowing this approach is the step length of the spot. This attribute controls the accurate locality and transition of spot from one region to another. Most of the times, a step, smaller than spot length side, can prove a good trend for proper laser attack impression, resulting in several overlapping spots. Step length larger than spot side misses to cover entire layout surface. Apparently, virtual spots that do not overlap extract groups with instances that appear uniquely in the analysis. This approach, though, is not consistent with the model of laser attack our validation flow tries to implement. Consequently, small step length that cause overlapping partitions are mainly taken into account. Figure 5.6 shows physical partitioning for equal values of step size and spot length side.

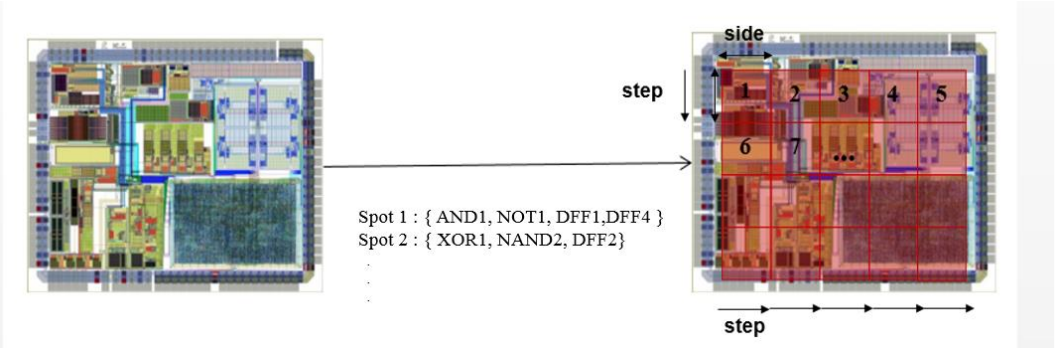

Figure 5.6: Laser spot approach (partitioning) for equal spot/step size

- **Scan entire Layout**

Our tool was configured to perform exhaustive examinations on the layouts under examination, covering the IC surface with thousands, or even millions, of partitions and extracting as many combinations of instances as possible. The exact dimensions of testing layouts are given to the tool, after inspecting the DEF format file, which gives away the information on the spatial characteristics of the design. A confirmation of the dimensions is done with another software tool, Glade™, which constitutes an IC layout viewer and is presented in the next sub-chapter. Once again, units must be in DB, not in actual units of micrometers.

- **Instance Filter**

The tool is specially designed to provide the user with the possibility to filter any unwanted layout components. Filtering is done either by type, if one needs to exclude certain instance types, or by name, if particular instances are unnecessary, or by spatial parameters. For example, user can control to extract

all instances covered by spots except for inverters. This is feasible by filtering sub-word «INV» on instances' metadata. Another filtering example involves the user being able to control not to extract instances larger than spot's size. Our analysis mostly handles instances, such as gates, registers and buffers. As a consequence, instances that describe connectivity or material equipment, such as vias (connectivity lines), metal, silicon layers or filler cells, are excluded by filtering, if needed.

- **Track Father Flip Flops**

Probably, the most important aspect of our validation platform and what really distinguishes it from a typical layout extraction tool is performing tracking of father flip flops on the Gate Level net-list of the design. As explained in much detail in chapter 6, Gate level net-list is read in order to form the Gate cones, which constitute a description relevant to RTL cones described before. The Gate level elaborated net-list is partitioned in cones in the same way as RTL cones are formed. For further explanation, a single spot on the layout may contain either combinational logic that constitutes parts of Gate Level Cones or sequential logic, implying flip flops, which show up at the top of Gate Cones. Extracted flip flops are stored for further processing. Combinational logic, though, is examined in regards with their occurrence in Gate Cones, according to its fan-out network. Each and every Father flip flop corresponding to the cones is extracted and taken into account for possible reach of fault propagation. As a consequence, each partition on the layout maps a certain number of flip flops, affected directly or indirectly.

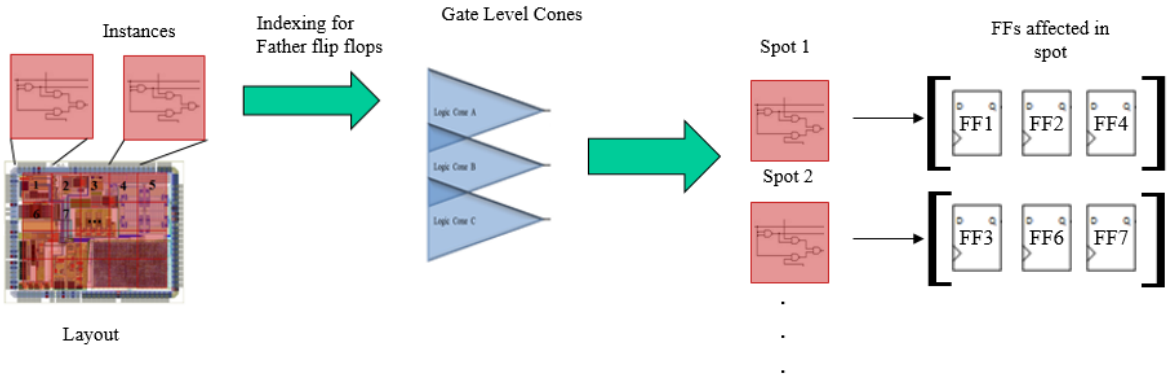

Figure 5.7: Partitioning and indexing Father FFs in Gate Cones

5.3 Glade™ Layout Viewer

Another helpful EDA tool, called Glade™, served for the realization and verification of the layout extraction tool. Thus, some information on this tool is presented. Glade™ (**G**ds, **L**ef **A**nd **D**ef **E**ditor) constitutes an IC layout viewer and editor and it is a freeware from Peardrop Design Systems, capable of reading GDSII, LEF and DEF file formats, as well as a few additional ones that were not encountered during the analysis. It offers the capability to load and display large IC designs using its fast, and rich in libraries object-oriented database. The main functions that it provides are [28]:

- A scriptable layout editor. Glade™ is extendable allowing scripting capabilities and offering access to the design database and its graphic user interface via programming functions.
- Net-list extraction. It provides with the original schematic or circuit diagram, describing connectivity of the design.
- Design Rule Check (DRC). A successful DRC ensures that the imported layouts conform to the design rules required for faultless fabrication.
- Layout Versus Schematic (LVS) comparison. LVS consists of verification that determines whether a particular IC layout corresponds to the original schematic or circuits diagram of the design.

Just like the OpenAccess™ translators, Glade™ uses the LEF/DEF files so as to load the design under examination. It can give the user a clear image of the details of the semiconductors and deepens to observe the metal, the n-well, p-substrate, the Voltage Drain (VDD) and Ground (GND) and the polysilicon, as shown in Figure 5.9. Glade™ can also have as input the GDSII stream format of a design, but this gives away only geometric information without offering substantial details on the top module of the design or its individual components.

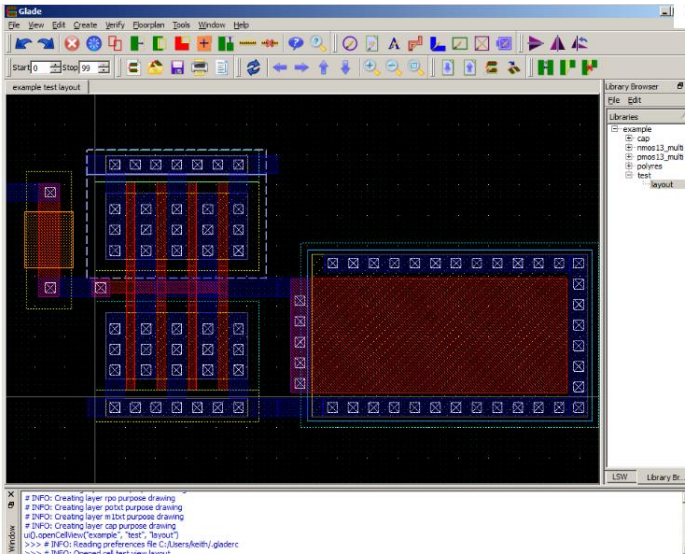

Figure 5.8: Screenshot from Glade™

Glade™ is an indispensable EDA tool and in combination with OpenAccess™ they form a complete set of tools that contributed to the testing and validation of the layout extraction tool, by inspecting the actual layout and the exact placement of gates or flip flops. During our work, all tested layouts were imported in Glade™ to inspect their spatial characteristics and determine the candidate locations for spot partitioning. Figure 5.10 demonstrates an example of importing the layout of AES Parity. All semiconductor devices are indicated with the light blue coloring in the center of the chip, while the surrounding framework offers no practical use, as it includes nets for Source and Ground. As can be easily inferred, layout extraction tool focuses on the center part of the chip to apply the partitioning analysis. The dimensions of the frame including all semiconductor devices can only be extracted by Glade™ and then fed to the extraction tool for proper configuration.

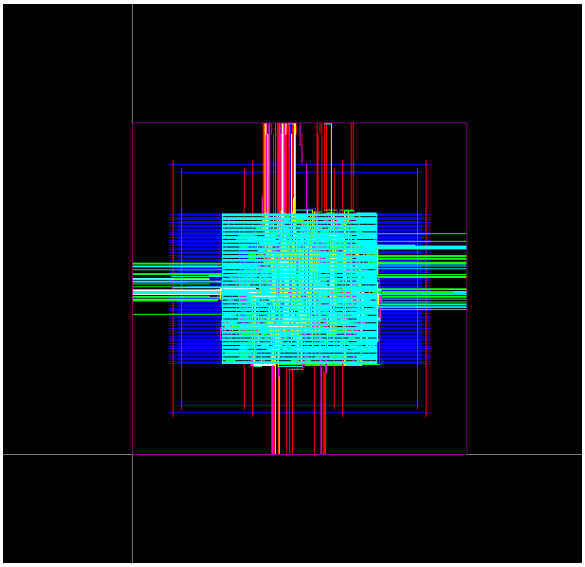

Figure 5.9: AES Parity layout view on Glade™

Chapter 6: Validation Methodology

6.1 Validation Flow

So far, our efforts have focused in the context of establishing a background on RTL fault modeling and on the design of a layout extraction tool representing localized attacks. Cone partitioning at the RTL tries to define the locality of a laser fault attack in an early stage of design flow. The layout extraction tool has been designed offering the flexibility and controllability of a potential laser spot in a realistic attack against ICs. As stated in chapter 5, each partition on the layout leads to the extraction of a certain set of flip flops. This set results from either flip flops that were included in the specific partition area or from combinational logic that indicates a connection to flip flops through their fan-out network, residing in the design surface. The number of layout sets including flip flops depends on the square spot dimensions and step. The flip flop sets deriving from the IC layout spots are referred as "Virtual Spot Sets". These are stored for further processing, as they will be compared with the flip flops sets formed at the RTL net-list.

Flip flop sets at the RTL derive from the intersection of logic cones in the RTL description. Cones are bounded by the net of the starting flip flop and expand backwards, from the outputs towards the inputs, up to either flip flops or primary inputs of a circuit [1]. Intersection technique is applied on the RTL cones so as to extract for each flip flop the set of flip flops that contain the occurrence of common instances in their cones. These sets are referred as "RTL Intersection Sets" or "RTL Sets".

After synthesis process applied on the RTL description, Gate Level net-list is produced. This net-list contains several optimizations comparing to that of RTL, concerning the combinational logic of the design, however most of the RTL flip flops remain unprocessed from synthesis and continue residing in the Gate Level net-list. Through further C++ processing, cone partitioning on Gate Level takes place and further examination is done in order to acquire the intersecting sets of this level. Gate Cones, just like RTL, are bounded by the net of the starting flip flop and expand backwards, from the outputs towards the inputs, up to either flip flops or primary inputs of the net-list. For each cone on Gate Level an intersection algorithm is implemented in order to check for instances that occur in multiple cones, leading to the extraction of intersecting sets at Gate Level. These sets are referred as "Gate Sets" and constitute the third and final input of validation analysis.

For the validation part, our purpose is to check if the Virtual Spot Sets from layout analysis form a match or constitute a subset of the RTL Sets. In that way, there exists a correlation between the early stage of design flow and the latest stage of fabrication, so a prediction of the impact of attack can be established. That is the final point on which our work wants to conclude. The information extracted from the comparison can provide the means to validate the RTL methodology which can ultimate lead to the enhancement of the IC design flow. Moreover, the checking between Virtual Spot Sets and Gate Level Sets takes place, in the same way as with RTL Sets. It is another valuable information for feedback at a next level of design, a level which does not differentiate from the IC layout.

6.2 Gate Level Cones and Intersecting Sets

The sub-chapter is divided into two, so as to present in one hand the synthesis applied, during the internship, for all designs in their early stage of design flow and, on the other hand to describe the algorithm implemented using OpenAccess™ API for the acquisition of the Gate Level cones and intersection sets.

6.2.1 Synthesis - From RTL to Gate Level Description

This section describes the procedure of design and manufacture of an RTL description to the Gate Level of abstraction. As mentioned before, combinational logic on the layout belongs to the fan-in network of flip flops. This network resembles a cone, which is different from the RTL cones for the reason that they exist only on the layout. These cones are referred as Gate Level Cones and they are the outcome of synthesis of the non-optimized net-list of the RTL description.

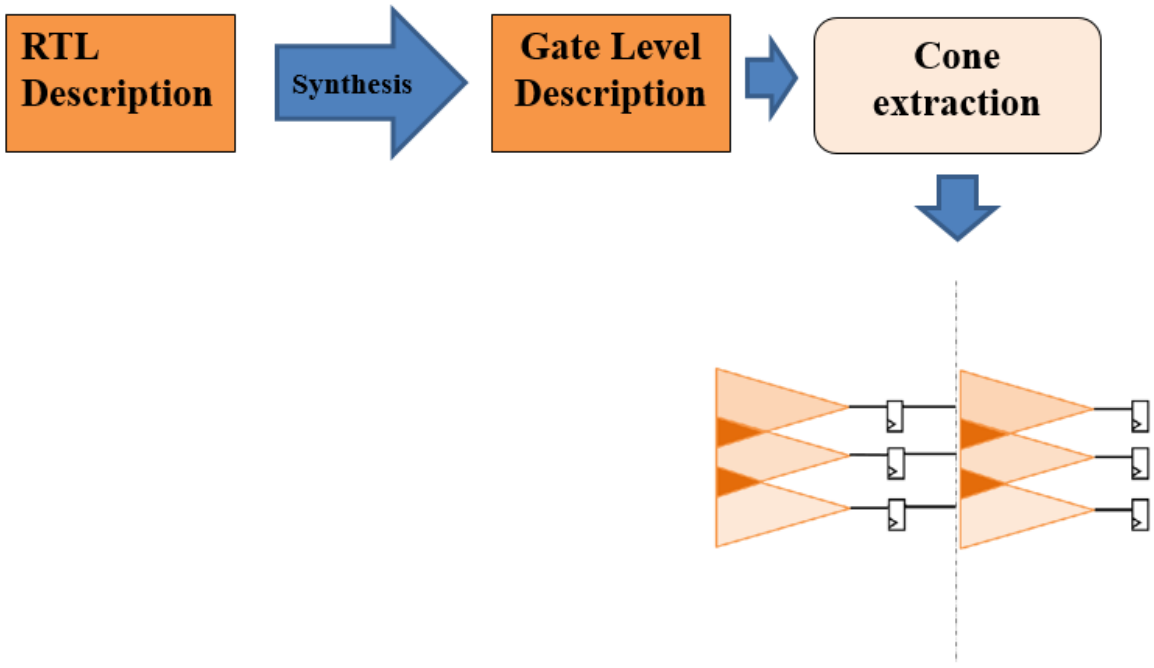

Figure 6.1: Synthesis and Cone extraction process at Gate Level

Logic synthesis takes the circuit description at the RTL level and generates an implementation in terms of an interconnection of logic elements. It is an automated design process in which high-level design descriptions, written in Hardware Description Languages (such as VHDL, Verilog, or SystemVerilog), are transformed into Gate Level net-lists. Gate Level net-list is basically a circuit

implementation of the design made of particular library components (both combinational and sequential cells), available in the technology library and their interconnections. Registers or important nodes of a design are not affected by synthesis. Typically, synthesis is done by a computer program called a synthesis tool. The net-list is generated by the synthesis tool according to several constraints, either set by design rules or by the designer. Figure 6.3 shows an overview of the synthesis that takes place on synthesis software tools.

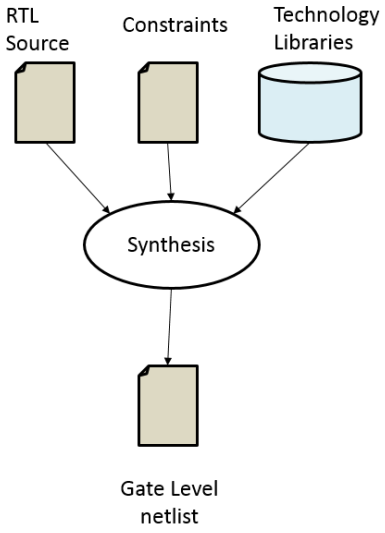

Figure 6.2: Synthesis from RTL to Gate Level

There are user specified constraints and design rule constraints. User specified constraints can be used to constrain the clock period, but they can also be used to constrain the arrival of certain input signals of the design, the drive strength of the input signals and the capacitive load on the output signals. Design rule constraints are fixed constraints which are specified by the standard cell library. For example, there are restrictions on the loads that specific gates can drive and on the transition times of certain pins. Synthesis tool will attempt to synthesize the design while still meeting the constraints.

The designs, which were provided to me in order to complete my internship, were synthesized using Design Compiler, provided by Synopsys™. Design Compiler (DC) takes the RTL hardware description, in VHDL, and a standard cell library (technology library) as inputs and produces a Gate Level net-list as output. The standard cell library used for all our designs is Nangate 45nm. The resulting Gate net-list is a Verilog file and constitutes a completely structural description with only standard cells at the leaves of the design. Internally, DC performs many steps including high-level RTL optimizations, RTL to un-optimized boolean logic, technology independent optimizations, and finally technology mapping to the available standard cells [32]. The Verilog file, thereby, can be processed in order to isolate the description of the flip flops in the design. Finally, using special commands on Design Compiler it is possible to extract a file containing the fan-in network of each flip flop. This file is called Report Transitive Fan-in and it contains all the nets and combinational logic residing in the fan-in network of the flip flops. Basically, this file includes the raw information of Gate cone partitioning. By

applying further processing on the file, using C++ functions, I have extracted the final form of Gate cones and intersecting sets needed for our analysis.

6.2.2 Extraction of Gate Cones and Sets

The procedure of Gate Level partitioning, is described as follows: The Report Transitive Fan-in file of the design is used as input in the OpenAccess™ C++ API. After implementing specific indexing algorithm, the input is examined in regards with the fan-in network of every flip flop and instances, uniquely appeared, are extracted and stored for further processing. Explored combinational instances, are marked in order not to be stored more than once. This indexing algorithm is applied for every flip flop existing in Gate Level net-list and the final result is the formation of Gate cones. Naturally, the number of cones formed is equal to number to the flip flops at the optimized Gate Level.

After Gate cone extraction, the sets containing combinational instances of the cones are processed by means of an intersection method. This method is included in the same OpenAccess™ program that controls the layout virtual partitioning. It is written using a particular and helpful data structure, introduced in C++, called multimap. Multimaps are associative containers that store elements formed by a combination of a key value and a data value. The key values are represented by the flip flops, featuring the cones. The data values are occupied by combinational logic instances. Therefore, multimap is filled by combinations of instances and corresponding cones. Figure 6.4 demonstrates a simplified format of the multimap programming structure, based on a simplified example of Gate Cones.

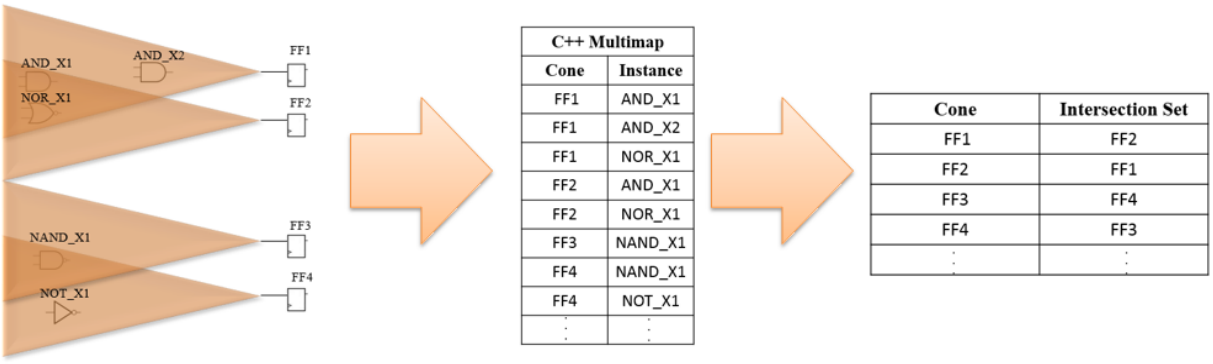

Figure 6.3: From Gate Cones to Gate Intersection Sets using multimap

Multimap contains as many rows as the number of instances. Each insertion is a pair of cone (flip flop) and instance. Based on this format, Gate Intersection sets are easily incurred. The algorithm considers all instances of a cone. It attempts, then, to track the occurrence of these instances in other cones, storing and extracting each newly detected cone to the set. This procedure is then repeated for all

the cones of the design and finally, we have at our disposal all the Gate Intersection sets, which are equal in number to the Gate Level flip flops.

6.3 Virtual Spot Sets on the Layout

Final step for the validation analysis is the acquisition of Virtual Spot Sets. This is done by the layout extraction tool designed by OpenAccess™, as mentioned in the previous chapter. But, before extracting the Virtual Spot Sets different layouts were provided to complete this thesis, in order for the spot partitioning method to take place. So, the final step is the transitive process from Gate Level net-list to the layout.

Layout design tool constitutes an Electronic Design Automation software that allows to digitize the shapes and patterns that form an integrated circuit. During the followed process for each design, the software tool used for the layout creation is SoC Encounter, provided by Cadence. Through the analysis using SoC Encounter, the Gate net-list of each design (Verilog file) is modified and optimized even more, but remains functionally equal to its corresponding HDL code. The net-list produced by the layout tool, after the layout has been done, is often called post-layout net-list. The noticeable difference between the pre-layout net-list and post-layout net-list is the inclusion of "clock tree buffers" in the second net-list. During the layout process, two phases are encountered, placement and routing. The first phase, Placement, involves deciding where to place all electronic components, circuitry, and logic elements in a generally limited amount of space. This is followed by Routing, which decides the exact design of all the wires needed to connect the placed components. This phase implements all the desired connections while following specific rules and limitations of the manufacturing process. Finally, the required DEF file containing placement and routing information of the layout design is extracted and all the necessary files (Verilog, LEF, DEF) that describe the layouts are obtained.

Summarizing for a single design under test, we have at our disposal the sets of FFs from the physical partitioning method, Gate intersecting sets and the RTL corresponding ones. Our analysis proceeds by checking if partition sets consist a subset of the RTL sets, and moreover, of Gate Level sets. Each partition represents a potential physical location on the layout, which is under attack. The fact that a set of affected flip flops may constitute a subset occurred in RTL/Gate sets proves to be valuable information with a view to predicting laser attack impacts, already from these levels of manufacture.

Chapter 7: Validation Results

Fault Injection modeling based on cone partitioning was applied on four designs. Employing the suitably configured layout extraction tool on OpenAccess™ and the validation algorithm flow, described in the previous chapters, we were able to quantify the accuracy of the RTL laser fault methodology. The process technology of our testing designs, used for validation analysis, is the Nangate 45nm open-cell library. Process technology (or technology node) with regards to digital integrated circuits, refers to the particular method used to make silicon chips. As implied by Moore's Law, the driving force behind the manufacture of integrated circuits is miniaturization, and process technology boils down to the size of the finished transistor and other components. The smaller the transistor, the more transistors there are in the same area. That means density of semiconductor devices is determined by technology node, thus it rises as technology node decreases

Each DUT was tested for several spot and step sizes, concerning localized attacks. It is noted that most of the lasers specialized in fault attacks have a diameter of 1μm, 5μm or 20μm, allowing a local attack. In our experiments, an additional spot size of 0,5μm was used for the sake for evaluation clarity, but it does not reflect a realistic dimension. The results are grouped in tables that present the following attributes: spot and step size used for each run, number of spots covering the layout, number of critical spots (i.e. spots including flip flops), spot max multiplicity (i.e. max number of flip flops affected by a single spot directly or not), critical spot max multiplicity (i.e. max number of flip flops affected by a single spot only directly), percentage of Virtual Spot Sets predicted in RTL and Gate Level sets and, finally, percentages of critical spots predicted in RTL and Gate Level. Cone method max multiplicity is a parameter used in our analysis to describe the maximum number of intersecting cones encountered in a design. As long as RTL and Gate cones are extracted for each design, RTL and Gate Level max multiplicity is displayed in the metadata of every design. Next phase is the mapping of spot partitioning for each design layout, in order to offer the reader some actual visual analysis on the results reflected by the tables. Mappings were produced using Matlab.

7.1 AES Morph

The first AES (AES Morph) constitutes an AES implementation containing hardware redundancy countermeasures. Secure chips containing such implementations are to prone to malicious activity. The physical characteristics of this AES layout, as well as max multiplicity disclosed from intersecting cones of RTL/Gate are as follows:

- Layout Dimensions
  - 260,87μm length (521740 DB units)
  - 257,6μm width (515200 DB units)
- # of Instances on Layout : 30158
  - # of Flip Flops : 2323
  - # of Gates : 27835
- Cone method max multiplicity
  - RTL : 1363
  - Gate : 2112

Validation methodology extracted the following results:

Spot (μm)	Step (μm)	# Spots	# Spots with Direct FF attacks	Spot Max mult.	Spot Direct Attack Max mult.	% of Spots predicted in RTL	% of Spots predicted in Gate	% of Direct attacks predicted in RTL	% of Direct attacks predicted in Gate
0,5	0,5	210718	58641	2064	4	93,74	96,52	93,92	96,05
1	1	54166	17479	2064	4	91,7	95,9	90,85	93,77
1	0,5	217901	72082	2065	4	91,37	95,65	90,05	93,45
5	5	2381	1286	2064	12	79,37	91,55	78,69	88,49
5	2,5	9692	5242	2066	13	79,66	91,87	79,13	88,99
20	20	182	152	2065	75	54,94	79,67	68,42	78,94
20	10	756	608	2087	76	56,08	82,01	67,43	79,77
20	4	4823	3769	2095	81	57,88	83,72	68,55	81,34

Table I: Validation Analysis for Morph AES

First observation addresses the max multiplicity of even the smallest spot (2064). This number reveals the effects of synthesis process and of other optimizations, inducing resource sharing so as to reduce the numbers of combinational logic and bring cones closer on the final layout. As spot size increases, there is a reasonable raise in the max number of affected flip flops, reaching in 2095 registers. Relatively to the maximum number of flip flops affected directly by a single spot, the value ranges from

4 to 81, indicating clearly that the impact of attack is proportional to spot size. As a reminder to the reader, step size pursues to approach the natural motion of a laser attack, in nanometer scale and with respect to spot size. In addition, it contributes to the extraction of a large number of differentiated combinations of registers, in accordance with repeated spot overlapping. With that said, the run of spot 20μm - step 4μm shows that as many different combinations of adjacent registers are extracted, the maximum number of affected flip flops tends to rise.

Another annotation which can be fruitfully inferred is that as spot size increases, percentage of spot coverage in RTL decreases, thus cone methodology success rate falls. Cone methodology's failure is due to the fact that RTL cones, affected from a layout spot, ultimately do not appear to intersect. The smaller the spot size, the higher rate of cone method success is presented. As noted before, RTL max multiplicity according to Cone methodology is 1363, a value lower than that of spot max multiplicity, for every spot/step combination. Thus, unavoidably several spots cannot be predicted in RTL sets. Another accurate point is the opposing trends between step size and RTL spot coverage for a given size spot, with an exception of spot 1μm - step 0,5μm where we meet a slight raise compared to the run of spot 1μm - step 1μm. Gate net-list is logically closer to the final formation of the layout, in terms of instances' number and connectivity network. Moreover, the trend of critical spot coverage in Gate/RTL is similar to the trend of spot coverage. It is reasonable for the small dimension spots to display larger coverage in contrast with larger spots, as the majority of small Virtual Spot Sets tends to be low in flip flop number, and therefore, it is more likely to be predicted in Gate/RTL sets.

Overall, Table I presents a comprehensive quantification of the accuracy of the RTL methodology. The results appear to be very encouraging, especially for smaller spot dimensions. Even the worst prediction rate appears to be above 50%, concerning 20μm spot. Feedback deriving from earlier stages of design indicates that cone method and laser fault modeling in the RTL or even Gate Level are accurate enough and, more importantly, this feedback can be utilized to contribute for countermeasure's appropriate design and implementation.

The following images depict the impact of laser attacks on the layout of DUT, concerning equal spot and step size values. This gives the reader a clearer indication on the important aspects of attack scenarios, as well as the most critical parts on the layout.

- **Spot**: 1μm, **Step**: 1μm

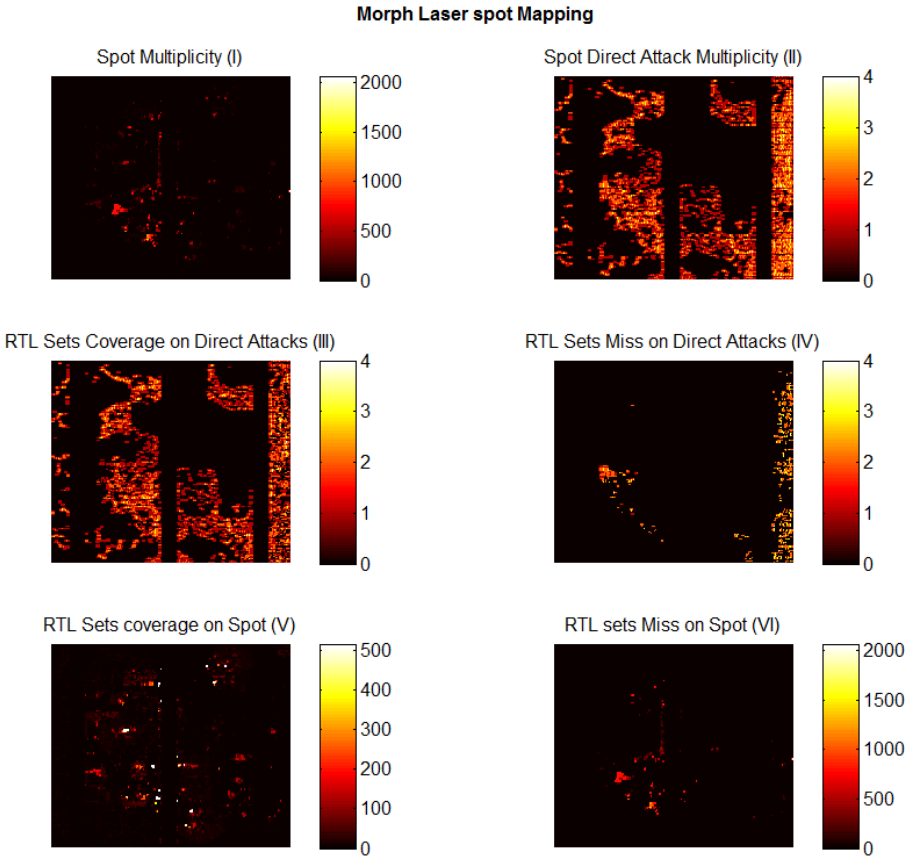

Figure 7.1: Mapping of Morph layout for spot and step: 1μm

Image I of Figure 7.1 presents spot multiplicity on the scaled Morph layout. There exist several spots containing high multiplicities, gathering mostly to the low center of IC surface. These spots are indicated with lighter coloring. The dark coloring areas indicate multiplicity levels below 500.

Image II shows the critical spots on the layout. Max multiplicity, as noted before is 4. The distribution of flip flops is prevalent in most surface area. Also, it is notable that several vertical black lines in the images indicate the absence of instances in these parts, denoting the usage mostly, for GND and VDD lines. Same trend appears in other designs, too.

Image III depicts the critical spots predicted from RTL analysis. Because critical spot max multiplicity is low, only 4, these spots are included in RTL intersecting sets, so image III looks almost identical to image II.

In image IV dark color prevails, indicating the area where RTL analysis manages to cover critical spot multiplicity. Therefore, colored spots are the ones not included in RTL sets. Once again, due to low max multiplicity, colored spots are less.

Image V depicts spots predicted from RTL analysis. Again, this includes most of surface area, but spots with multiplicity over 400 are clearly indicated to be fewer, when in the contrary many spots with small multiplicity are reasonably included in RTL sets.

Colored spots in image VI reveal failure of the RTL method. This failure is justified by max multiplicity values of spots in the low center of surface. Usually, image VI resembles to image I, excluding spots with multiplicity lower than 1500.

- **Spot**: 5μm, **Step**: 5μm

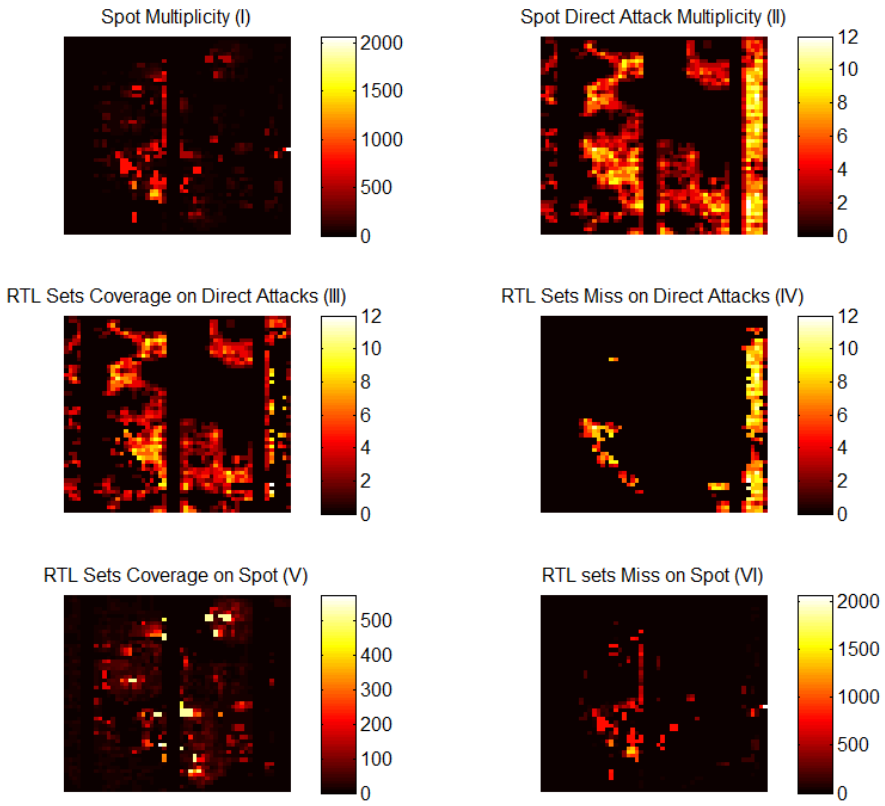

Figure 7.2: Mapping of Morph layout for spot and step: 5μm

Spot square approach is quite apparent in Figure 7.2, thus resolution is lower. Images follow the same trend as before, with the difference that detail on the attack has been reduced due to larger spot size, thus images enhance the density of flip flops in a given spot. Once again, color scaling of the images follows the pattern of max multiplicity, met in spots. It is quite apparent that the impact of a

localized attack with spot 5μm against flip flops is heavier. Max multiplicity for images II, III and IV has increased to 12, always according to Table I.

- **Spot**: 20μm, **Step**: 20μm

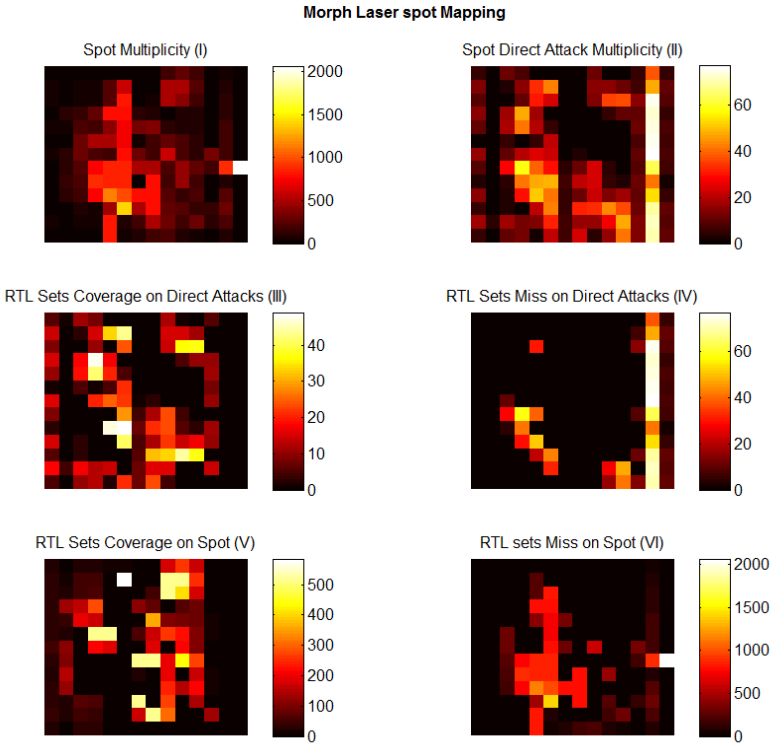

Figure 7.3: Mapping of Morph layout for spot and step: 20μm

Approach of laser attack with diameter of 20μm is shown in Figure 7.3. Due to bigger spot size, resolution is even lower. Once again, images follow the same trend as before, with the difference that detail on the attack has been reduced even more. This type of analysis facilitates the observation on the density of flip flops. From image I and with respect to the bigger scale, it is quite indicative which spots display max multiplicities. Even more, in image II density of flip flops is higher on the right side of the layout.

7.2 AES Parity

AES Parity is the second design used for examination. Several hardware implementations for AES circuit have been proposed and designed. No matter the type of implementation, the most expensive part of the circuit in terms of area is the so called SBox. Briefly, SBox contains the Substitution Table for the SubByte step of encryption (sub-chapter 2.4.2). AES Parity algorithm of encryption makes use of a novel parity bit scheme to protect the SBox core. This parity bit is used for checking for each byte element of the matrix. AES Parity is a widely used implementations of AES, so our work includes this cryptographic circuit in the validation analysis. We have to note that for this particular design reset logic was subtracted from RTL description. Attack on combinational logic of reset will result in the affection of many flip flops of the design, without allowing the extraction of valuable information, since such attacks can be easily detected. With that said, the subtraction of reset logic normally should not be taken into consideration in any layout analysis.

Again, for a certain range of laser spot side and step, a series of partition runs have been implemented. Layout attributes, as well as Cone method max multiplicities are referred below:

- Layout Dimensions
  - 170,81μm length (341620 DB units)
  - 279,2μm width (558400 DB units)
- # of Instances on Layout : 13798
  - # of Flip Flops : 932
  - # of Gates : 12866
- Cone method max multiplicity
  - RTL : 201
  - Gate : 685

Validation methodology extracted the following results:

Spot (μm)	Step (μm)	# Spots	# Spots with Direct FF attacks	Spot Max mult.	Spot Direct Attack Max mult.	% of Spots predicted in RTL	% of Spots predicted in Gate	% of Direct attacks predicted in RTL	% of Direct attacks predicted in Gate
0,5	0,5	89125	23965	651	4	86,37	93,19	91,78	93,07
1	1	22870	7229	652	4	77,58	89,60	86,49	88,75
1	0,5	92039	29958	652	4	76,33	88,91	85,89	88,20
1	0,3	255577	83013	652	4	76,36	88,89	85,92	88,20
5	5	1046	609	655	11	36,99	70,07	59,77	67,81
5	2,5	4199	2501	655	11	35,72	68,82	60,37	68,61
20	20	81	79	711	49	7,4	28,39	24,05	36,71
20	10	341	131	711	54	7,33	34,31	27,79	41,21
20	4	2159	1995	711	59	9,26	35,89	29,97	43,25

Table II: Validation Analysis for Parity AES

AES Parity constitutes a much smaller design than AES Morph. This can be easily inferred by, not only the actual layout dimensions, but also the spot max multiplicity encountered during the spot partitioning. Its range varies from 651 to 711, in contrast with AES Morph where it reaches 2095. Cone max multiplicity on RTL and Gate Level is sufficiently lower than these of the previous design and this is justified by the removal of reset logic. A first observation concerns the spots and critical spots coverage from RTL sets, where the rates have decreased in regards with the rates at AES Morph. For an actual laser attack (i.e. excluding spot/step size 0,5μm), the upper rate of RTL coverage of spots is 77,58%, and for critical spots 86,49%, met for spot 1μm – step 1μm. This is reasonably inducted from the high deviation between spot max multiplicity and RTL sets max multiplicity. Again, the proposed fault model presents a successful prediction rate for spot size of 1μm, although prediction rates cease to appear elevated for spot sizes of 5μm and 20μm. Percentages are at the order of 35-36% and 7-9%, respectively. Nevertheless, for a sophisticated, accurate fault attack that targets to the altering of encryption process and the disclosure of secret key, the appropriate laser model has the diameter of 1μm. Synthesis optimizations are quite obvious again, as it is witnessed by, not only the max cone multiplicity being 685, but also the increased rates for spot and critical spot coverage on Gate Level. Respectively, these rates range from 28,39 - 93,19% and from 36,71 - 93,07%.

Overall, localized attack is quite accurately represented on RTL (and even better on Gate Level) only for spot size 1μm. The other two types of lasers fail to offer an apt localized attack approach, as many spots include flip flops whose cones in the RTL do not intersect.

The following images present the spot partitions overview of the AES Parity, with regards to equal spot and step size values. Number of flip flops, affected directly and/or indirectly for any given partition, as well as the inclusion of partitions in RTL sets are presented.

- **Spot**: 1μm, **Step**: 1μm

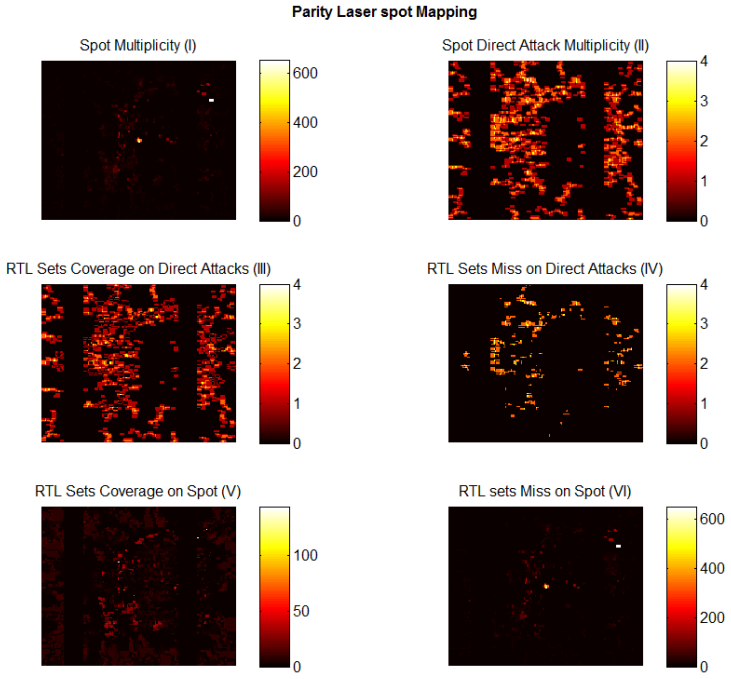

Figure 7.4: Mapping of Parity layout for spot and step: 1μm

Figure 7.4 introduces the mapping images on AES Parity layout. Image I shows the distribution of flip flops, affected either directly or indirectly, per laser spot, underlining the density of the design in instances. Darker coloring is quite prevalent, implying that most spots contain multiplicities of values below 200. With that said, and according to Table II, where spot max multiplicity was found 652, this value is met only in very few spots. Checking in depth these spots, we meet very few inverters, whose fan-out network contains very large range of flip flops. It is probable that these instances may have some connection with reset input wiring, which for this specific design, was abstracted as much as possible.

Image II indicates the distribution of flip flops affected only directly by a spot. It is necessary to state again that the two appearing black vertical lines, which include no instance whatsoever, are used for GND and VDD lines.

Images V and III sufficiently justify the elevated rates of spot and critical spot coverage from RTL sets, respectively.

- **Spot**: 5μm, **Step**: 5μm

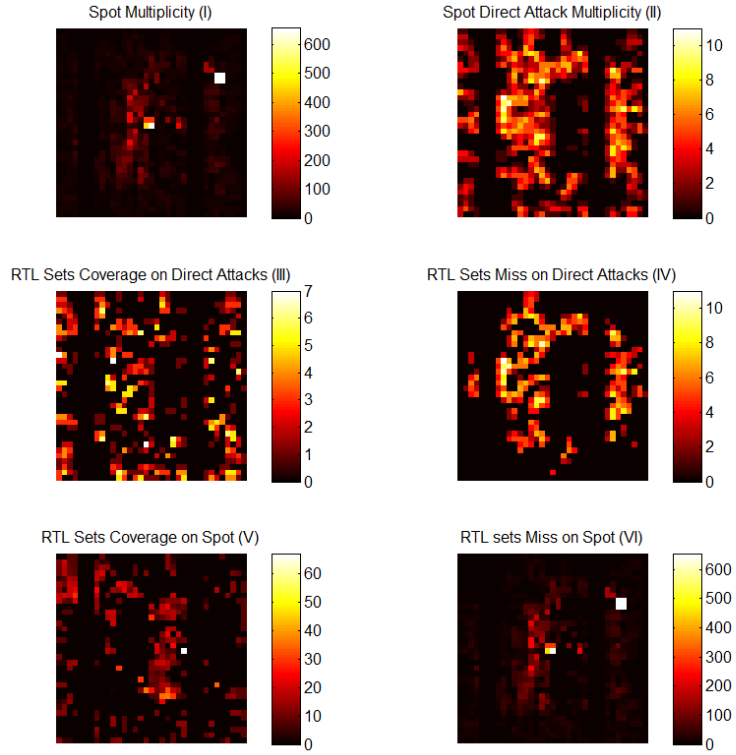

Figure 7.5: Mapping of Parity layout for spot and step: 5μm

By defocusing a realistic laser spot, it is able to succeed layout partitioning with diameter 20μm. Such an approach is presented in Figure 7.5. RTL laser fault modeling barely fails to generate remarkable inferences, in this scenario, as the spot coverage on RTL sets drops. According, to Table II, for spot 5μm – step 5μm, spot and critical spot coverage yields 36,99% and 59,77% respectively. These ratios are reflected in images V and III, where dark coloring, especially in image V, is quite prevalent. RTL set coverage on direct attacks outputs encouraging results, as many of the critical Virtual Spot Sets are predicted in the RTL analysis. On the other hand, RTL set coverage on spots has little to offer, as many of the Virtual Spot Sets do not manage to consist a subset of any RTL intersecting group.

- **Spot**: 20μm, **Step**: 20μm

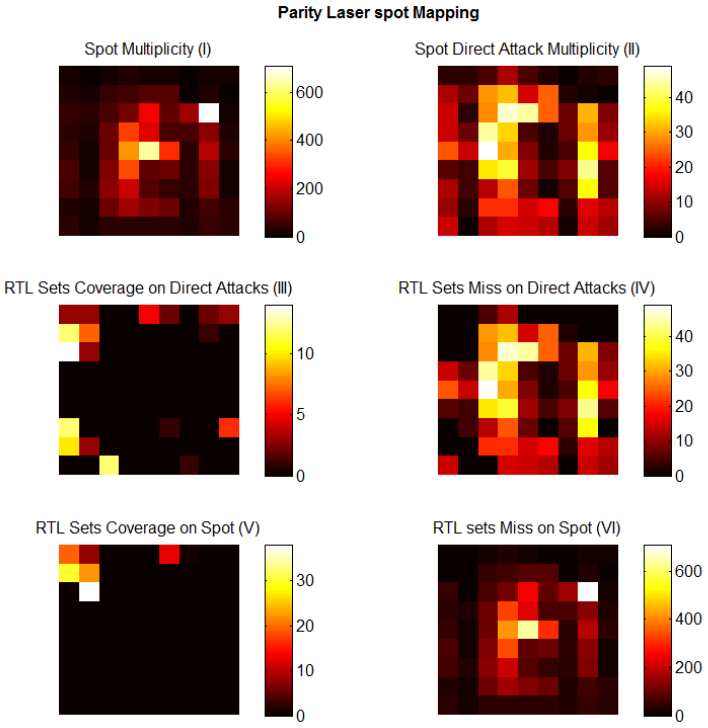

Figure 7.6: Mapping of Parity layout for spot and step: 20μm

Without being able to extract some beneficial conclusions from the run of spot 20μm – step 20μm on the layout of AES Parity, we present the corresponding spot mapping. It is clear, from images III and V, that the methodology fails dramatically to represent laser attacks on RTL, as flip flop affected from attacks correspond to non-intersecting RTL cones. On the other hand, images IV and VI indicate which spots are not contained in RTL sets, and additionally their multiplicity is indicated from the coloring.

7.3 ITC-99 Benchmark B17

Validation methodology is applied also on non-cryptographic circuits, so as to infer some valuable notes concerning their RTL net-list and cone forms. Laser attacks on system processors are not accustomed by IC hackers, but still they are feasible and can result in modification of the hardware operation flow, skipping of instruction sets or crushing the operation. ITC-99 benchmarks consist of a set of circuits whose characteristics are typical of synthesized circuits [35]. Benchmark B17, tested in the context of the validation process, includes three subsets of the 80386 processor, manufactured by Intel. The Intel 80386, also known as i386 or just 386, is a 32-bit microprocessor introduced in the mid 90's. An important annotation is that 80386 instruction set, programming model, and binary encodings are the common denominator for all 32-bit x86 processors, which is termed the i386-architecture. Although i386 had long been obsolete as a personal computer CPU, Intel and other companies had continued making this specific chip for embedded systems. Such systems are common in aerospace technology and electronic musical instruments, among others, even nowadays [36]. Some mobile phones also have used the 80386 processor. After stating some notable facts about our DUT, the validation results are presented as follows:

- Layout Dimensions
    - 182,59μm length (365180 DB units)
    - 279,2,3μm width (558400 DB units)
- # of Instances on Layout : 13731
    - # of flip flops : 1317
    - # of Gates : 12414
- Cone method max multiplicity
    - RTL : 374
    - Gate : 446

Spot (μm)	Step (μm)	# Spots	# Spots with Direct FF attacks	Spot Max mult.	Spot Direct Attack Max mult.	% of Spots predicted in RTL	% of Spots predicted in Gate	% of Direct attacks predicted in RTL	% of Direct attacks predicted in Gate
0,5	0,5	102302	40287	369	4	95,55	98,55	98,01	99,56
1	1	26315	12374	369	4	92,90	97,85	96,71	99,28
1	0,5	105836	51598	369	4	92,63	97,78	96,49	99,20
5	5	1175	984	369	9	78,38	89,78	86,89	94,61
5	2,5	4801	4054	369	9	78,96	90,14	86,75	94,62
20	20	90	89	479	47	51,11	63,33	65,16	77,52
20	10	377	368	566	53	55,43	63,92	63,04	76,90
20	4	2423	2377	584	57	57,69	67,76	63,56	76,61

Table III: Validation Analysis for ITC-99 Benchmark B17

In contrast with the two previous designs, which constitute different implementations of AES, this design does not include any cryptographic mechanism, so it is interesting to examine the cone formation at the RTL and Gate Level for such designs, as well as the outcome of the validation process. A first remarkable observation has to do with the RTL cone max multiplicity which reside at the same standards as spot max multiplicity. As opposed to the cryptographic designs, where RTL multiplicity is by far smaller than any spot max multiplicity, here RTL cone value (374) is higher for all virtual spots of 0,5μm – 5μm. That is a first clear indication of the high levels of spot coverage on RTL sets. For laser spot of 20μm, RTL cone max multiplicity is lower than spot multiplicity, not by far. The same trend is also observed with Gate cone max multiplicity.

Once again, the bigger the spot size the lower the coverage rate in RTL/Gate sets, but the smaller the step size, the slightly higher the rate gets (except for the run of spot 1μm – step 0,5μm). Percentage of spots predicted in RTL analysis varies from 51,11% (spot 20μm – step 20μm) to 95,55% (spot 0,5μm – step 0,5μm). Especially for small range spots, the proposed methodology is accurate and highly validated, as the majority of the Virtual Spot Sets are, in fact, subsets of RTL sets. Concerning the spot coverage in Gate Level, the percentages are even higher, as expected. Highest and lowest rate coverage is 98,55% and 63,33%, respectively, reflecting once more the optimizations taken place and the reduction of redundant combinational instances during synthesis. Feedback from the analysis in RTL and Gate Level can be fruitfully utilized to contribute for the design of proper countermeasures.

Concerning the critical spots (i.e. spots where only sequential logic is taken into consideration), coverage rates are quite elevated. It is stated that for RTL coverage, rates range from 63,04% to 98,01% and for Gate level coverage the same rates vary from 76,61% to 99,56%. Overall, Table III presents a validation of localized attacks on the layout for a higher level of abstraction in design flow. The results appear to be very encouraging, especially for the smaller spot dimensions. Even the worst prediction rate appears to be above 60%, concerning 20μm spot.

The following images present the layout overview of Benchmark B17, covered in partition spots. Once again, images concern equal spot and step sizes in order to avoid overlapping, and extract a clear division of the layout into multiple particles and the impact of attacks, concerning equal spot and step size values. This gives the reader a clearer indication on the important aspects of localized attack, as well as the most critical parts on the layout.

- **Spot**: 1μm, **Step**: 1μm

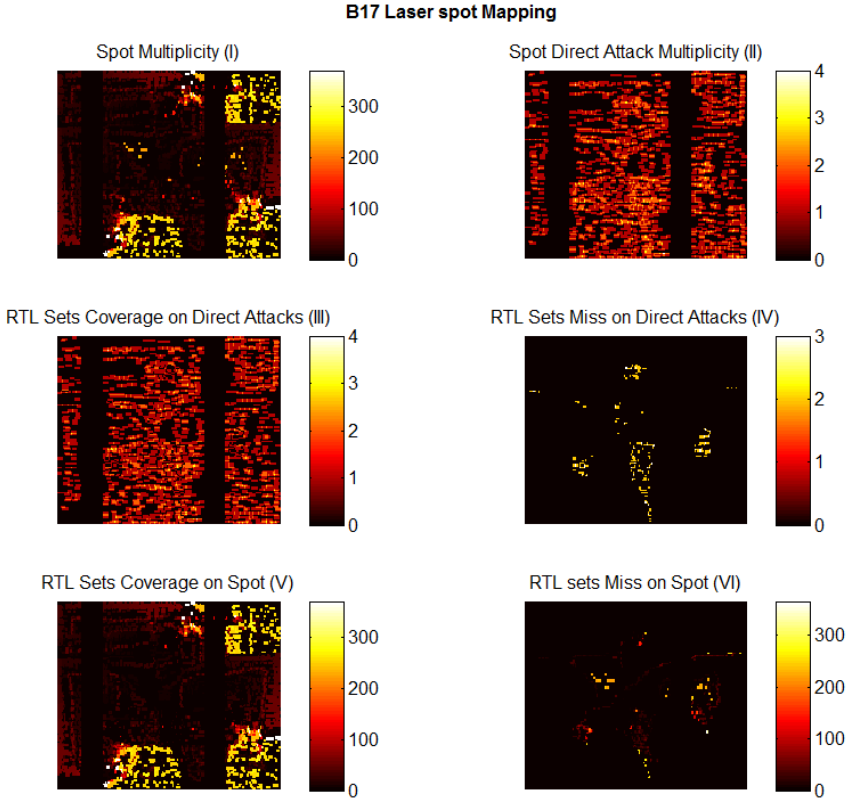

Figure 7.7: Mapping of Parity layout for spot and step: 1μm

Image I presents the spot multiplicity of B17. In the majority of spots, multiplicity is calculated below 200. In lighter coloring, three different area are distinguished, where multiplicity yields high values. By focusing on these area of the layout, we meet not only sequential, but also combinational logic (especially inverters and XOR gates) that belongs in the fan-in network of many flip flops. A potential attack in one of these areas, colored in yellow, would provoke a major impact, in terms of faults spreading throughout the circuit, that would affect almost ¼ of the registers.

Image II depicts the distribution of flip flops on the surface of the design. Max multiplicity, as noted in Table III, is 4. It appears almost identical with image III, which represents RTL coverage on critical spots, and this similarity is justified by the high percentage of critical spot coverage (96,71%), met in Table III. Thus, the resemblance of these two images can be used as a measure to decide the validation response of the proposed methodology, concerning only critical spots.

Image IV offers a visual analysis on the critical spots (spots addressing direct attacks) that do not confirm the methodology. Fortunately, they appear to be quite few, representing only 1% approximately

of total critical spots. The darker image IV appears, the more successful our validation technique is proved to be. Also, image VI reveals that approximately only 2% of the spots are not included in the RTL sets, noting once more the success of the validating our method in B17. Finally, resemblance of images I and V can also be a good measure to identify the accuracy of the proposed methodology, concerning only spots in general.

- **Spot**: 5μm, **Step**: 5μm

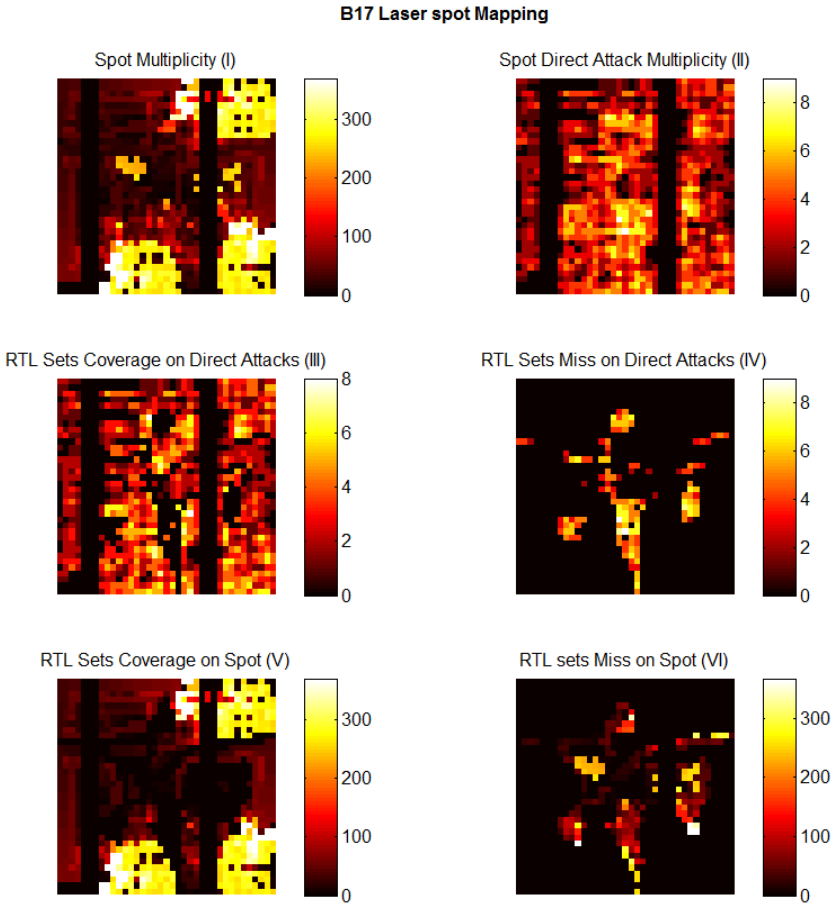

Figure 7.8: Mapping of B17 layout for spot and step: 5μm

According to the similarities between I and V, a good percentage of coverage is depicted but obviously the value is lower than in the run of spot 1μm – step 1μm. The three previously mentioned areas that are distinguished due to lighter coloring are even denser and extract greater values of multiplicities. This is due to the larger spot size in this specific run. In addition, images II and III look alike, fact that validates what has been previously stated.

- **Spot**: 20μm, **Step**: 20μm

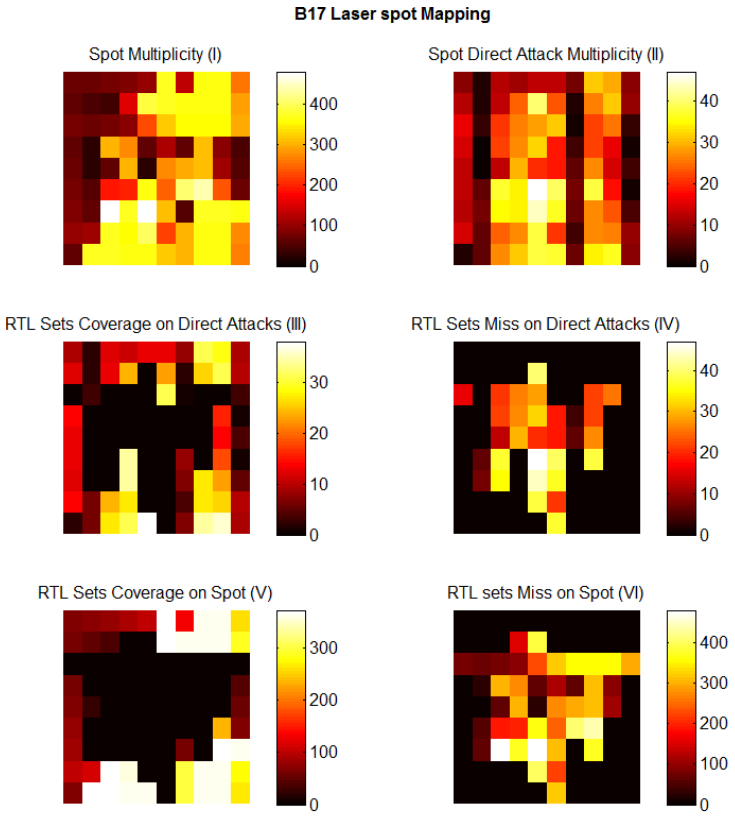

Figure 7.9: Mapping of B17 layout for spot and step: 20μm

Figure 7.9 reveals a high deviation between images I and V, reflecting the low validation success rate, noted at 51,11%. For one more time, it is prevalent that large spot sizes tend to fail validation technique, as non-intersecting cones in RTL are affected by the large layout spots.

7.4 ITC-99 Benchmark B18

From the same ITC-99 IC benchmarks, B18 is tested through the validation methodology. Benchmark B18 consists of six subsets of i386 processor and two subsets of Viper processor and constitutes the second largest benchmark design in ITC-99 series. Viper is a 32-bit micro-processor design created by Royal Signals and Radar Establishment in the 1980s, intended for use in safety critical systems, such as avionics [37]. The physical characteristics of B18 layout are presented as follows:

- Layout Dimensions
  - 316,54μm length (633080 DB units)
  - 313,6μm width (627200 DB units)
- # of Instances on Layout : 39747
  - # of flip flops : 3020
  - # of Gates : 36727
- Cone method max multiplicity
  - RTL : 560
  - Gate : 446

Validation methodology extracted the following results:

Spot (μm)	Step (μm)	# Spots	# Spots with Direct FF attacks	Spot Max mult.	Spot Direct Attack Max mult.	% of Spots predicted in RTL	% of Spots predicted in Gate	% of Direct attacks predicted in RTL	% of Direct attacks predicted in Gate
0,5	0,5	315300	93152	478	4	88,66	90,59	97,94	99,40
1	1	81587	28951	478	4	87,21	90,37	96,72	99,17
1	0,5	328149	119996	478	4	86,96	90,28	96,54	99,07
5	5	3541	2545	481	10	78,50	87,23	89,11	96,07
5	2,5	14383	10290	496	10	78,05	86,60	89,15	96,16
20	20	256	239	633	55	55,85	68,75	61,92	81,17
20	10	1082	980	633	55	61,46	71,25	66,93	82,55
20	4	6906	6177	633	56	62,45	72,22	68,59	83,27

Table IV: Validation Analysis for ITC-99 Benchmark B18

B18 is the largest IC layout that was examined, in terms of dimensions and of instance plethora. The fact that RTL cone max multiplicity has higher value than the corresponding in Gate Level is quite remarkable. The trend resembles to that of the B17. A large percentage of RTL coverage is calculated, varying from 55,85% to 88,66%. As for the Gate spot coverage, it ranges from 68,75% to 90,59%. Once

again, RTL fault methodology is highly validated, especially for smaller virtual spots. This is inferred by the high spot coverage in RTL. Gate Level presents even higher spot coverage, as expected.

A remarkable observation, concerning both non-cryptographic circuits, addresses the lower levels of spot max multiplicity, as opposed to cryptographic circuits, where cones are overly adjacent. This can be formalized by saying that in cryptographic implementations a potential laser attack is going to affect a large number of flip flops, making it difficult for the attacker to accomplish his purposes.

- **Spot**: 1μm, **Step**: 1μm

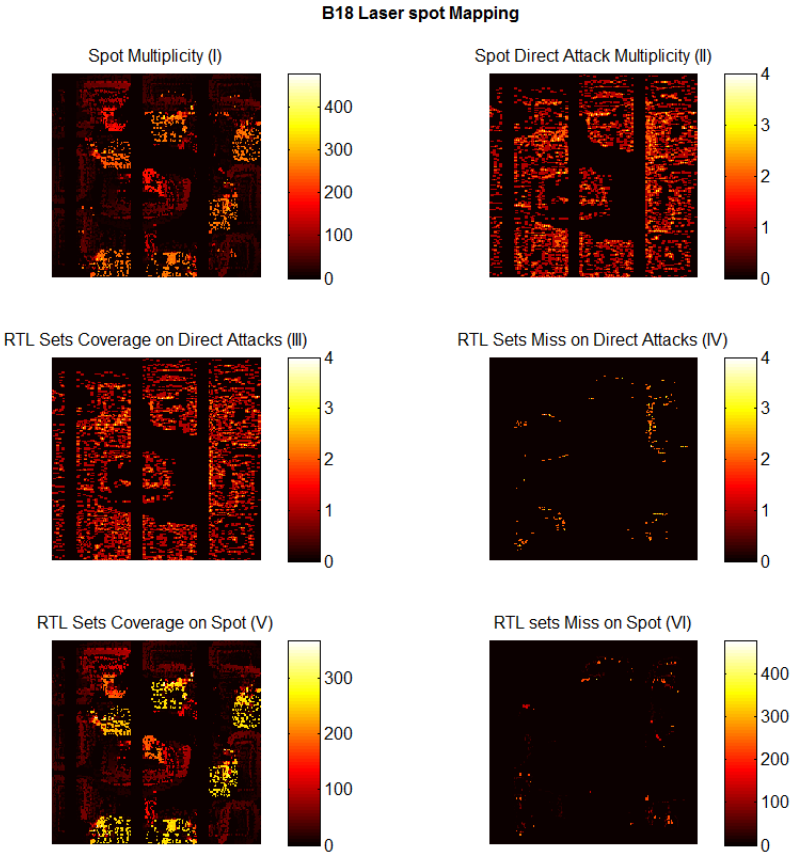

Figure 7.10: Mapping of B17 layout for spot and step: 1μm

Image I presents spot multiplicity on the scaled B18 layout. There exist several spots containing high multiplicities, scattered all over the IC surface. The dark coloring areas indicate multiplicity levels below 200. A similar pattern is observed between image I and V, revealing the success of our validation methodology. Images II and III, regarding direct attacks on spots, are shown identical, giving away the high percentage (96,72%) of critical spots coverage in RTL sets. Images IV and VI, due to the prevalent dark coloring, present the successful rates of coverage.

- **Spot**: 5μm, **Step**: 5μm

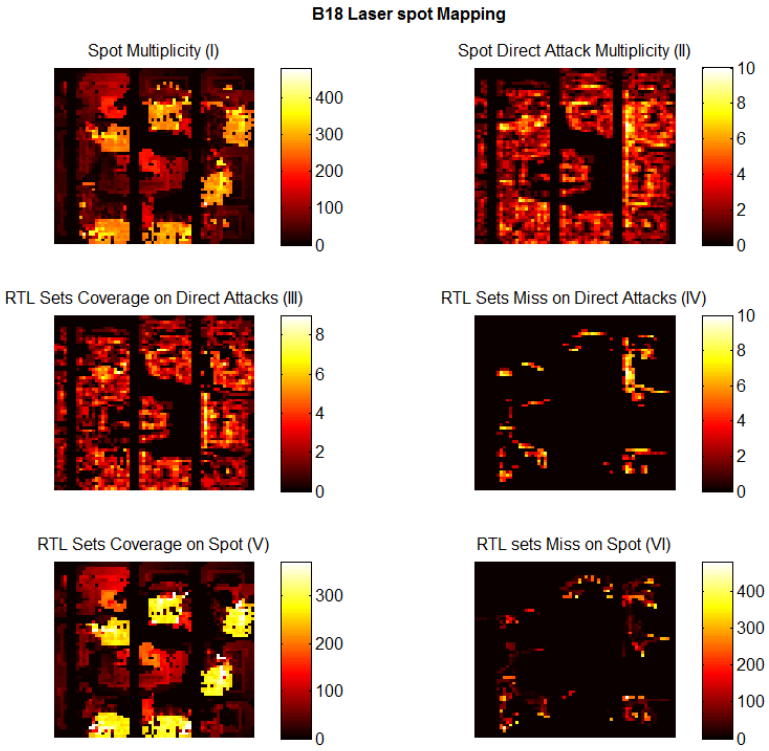

Figure 7.11: Mapping of B17 layout for spot and step: 5μm

Due to larger spot size, validation methodology derives lower success rate, as opposed to spot/step size 1μm. Image II and III are similar, thus critical spot coverage on RTL presents very high percentage (89,11%).

- **Spot**: 20μm, **Step**: 20μm

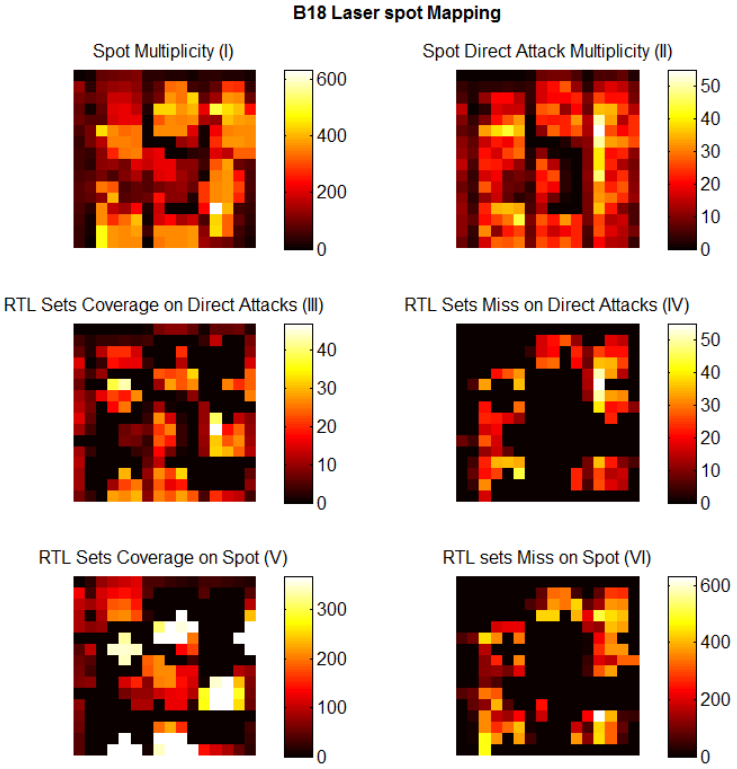

Figure 7.12: Mapping of B17 layout for spot and step: 20μm

The lower percentages of spot and critical spot coverage on RTL are reflected by the images, for spot/step 20μm. Nevertheless, validation results seem optimistic, as the percentages derived are above 50% for spot and above 60% for critical spot RTL coverage.

Summing up for all the verified designs, RTL fault methodology was greatly validated on the layouts. High percentages of spot coverage in RTL sets were encountered, especially for small size virtual spots (1μm, 5μm). Such size is what concerned us the most, as an accurate and sophisticated localized attack dictates the usage of small diameter laser. In general, laser spots affecting multiple flip flops can be highly predicted already at the RTL analysis. This leads to a significant feedback for manufacturing integrated circuits and, in particular, for establishing effective countermeasure with high precision. Moreover, analysis efforts on Gate Level produce another appreciable source of information with respect to predicting attacks from a subsequent, but still abstract level of design. Validation of the methodology on the layout of two AES sub-circuits, as well as of two processor-based benchmarks yielded optimistic results.

Chapter 8: Conclusion

Digital integrated circuits, the building blocks of modern computer hardware systems, have literally invaded in our lives. From the smallest to the largest electronic devices, ICs make their presence perceptible and their appropriate and safe operation is a subject that draws the attention of hardware designers and engineers, as well as of common users. Testing and validation methodologies are applied in all circuits during fabrication stages, however they prove to be inadequate in case of deliberate fault introduction. Protection implementations against malicious attacks on hardware is an aspect that concerns safe IC functionality and is accomplished by the installation of countermeasures. Hardware hackers tend to unleash multiple attack techniques against ICs in order to compromise the protective nature provided by countermeasures. Especially, security-oriented ICs constitute the main target for malicious activity, as they are the ones that keep hidden and confidential information. Cryptographic algorithms, such as AES, DES and RSA are implemented in such designs.

The security of digital integrated circuits can be compromised by covertly inserted malicious attacks. Hardware attacks pose a threat to cryptographic circuit implementations [1]. There exist many types of attacks against ICs, but fault attacks are the dominant type. Lasers provide a very effective means to perform fault injection attacks on integrated circuits, mainly because of their high precision locality, accurate timing and high occurrence probability. Therefore, proper countermeasures have to be employed to secure cryptographic circuits from such attacks, by not allowing the exposure of critical information to the attacker. Designers have already recognized the importance of incorporating fault tolerance into microelectronic devices. However, they often performed this task late in the process, when the design was near completion. As hardware systems become more complex, designers have already considered fault tolerance throughout the design process to allow early estimation of reliability and fault coverage. Study and analysis at the Register Transfer Level of abstraction can enhance the design flow, offer a fast evaluation and lead to the exposure of vulnerabilities of security oriented IC designs, and at the same time to the implementation of both defensive and preventive mechanisms [1].

Fault injection is a great technique for the evaluation of design metrics such as reliability, safety and fault coverage. Scientists realized the need for representing actual faults in hardware systems, so as to drain feedback for design and manufacture stages. For the proper representation of faults occurring in system components, a plethora of fault models has been established. A small but basic sample includes bit flip, stuck-at, random and set/reset, as mentioned in this thesis. In previous literature many fault injection techniques based on simulations and emulations are described. In addition, laser fault injection techniques at Gate and RT level have been introduced, representing fault introduction by statistical and probabilistic methods.

The current work aims to the development of a tool towards the validation of an RTL level laser fault model by using layout information. The Layout Extraction Tool is developed in order to create the localized attack approach. This tool allows to inspect the components of the layout and assess the impact of a potential laser attack, with regards to the number of flip flops affected. Testing, aiming at the validation of the RTL fault methodology, is applied on the layout of different designs. Virtual spots on

the layout are examined with respect to the sequential logic affected, so as to find out if the RTL cone analysis covers the set of flip flops by a potential laser attack. The validation algorithm dictates the comparison between flip flops affected by a localized spot (Virtual Spot Sets) and the intersection sets on the RTL (RTL Sets), as well as comparison between Virtual Spot Sets and the intersection sets at Gate Level (Gate Sets). The correlation of layout activity with RTL Cone analysis is the main reference point, described in this thesis. A technique, where laser fault attacks can be studied in the early RTL analysis, has firmly been established.

The RTL methodology was validated on four different IC layouts, two concerning AES implementations and two benchmarks, including older processors. Overall, results are quite promising and reveal that valuable feedback can be collected from RTL analysis and contribute to the design and evaluation of countermeasures. Local attacks in the greatest part of IC layouts (concerning small range attack) can finally be predicted at the RTL. Next phase of the research includes the study of layout spots that were not predicted by the methodology. By deepening into the logic components that led to the failure of the model, proper information can be extracted indicating the reasons methodology failed, and revealing inner mechanisms of synthesis, placement and routing in the design flow of ICs that ultimately could not be taken into consideration by the methodology.

References

[1] Papadimitriou, A., Hély, D., Beroulle, V., Maistri, P., & Leveugle, R. (2014, March). A multiple fault injection methodology based on cone partitioning towards RTL modeling of laser attacks. In *Proceedings of the conference on Design, Automation & Test in Europe* (p. 206). European Design and Automation Association.

[2] Kocher, P. C. (1996, January). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology—CRYPTO'96* (pp. 104-113). Springer Berlin Heidelberg.

[3] Kocher, P., Jaffe, J., & Jun, B. (1999, January). Differential power analysis. In *Advances in Cryptology—CRYPTO'99* (pp. 388-397). Springer Berlin Heidelberg.

[4] Homma, N., Aoki, T., & Satoh, A. (2010, July). Electromagnetic information leakage for side-channel analysis of cryptographic modules. In *Proc. IEEE Int Electromagnetic Compatibility (EMC) Symp* (pp. 97-102).

[5] Biham, E., & Shamir, A. (1997). Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology—CRYPTO'97* (pp. 513-525). Springer Berlin Heidelberg.

[6] Kömmerling, O., & Kuhn, M. G. (1999, May). Design principles for tamper-resistant smartcard processors. In *USENIX workshop on Smartcard Technology* (Vol. 12, pp. 9-20). [7] K. Nohl, D. Evans, S. Plotz and H. Plotz "Reverse-Engineering a Cryptographic RFID Tag"

[8] Kumar, R., Jovanovic, P., & Polian, I. (2014, July). Precise fault-injections using voltage and temperature manipulation for differential cryptanalysis. In *On-Line Testing Symposium (IOLTS), 2014 IEEE 20th International* (pp. 43-48). IEEE.

[9] Dehbaoui, A., Dutertre, J. M., Robisson, B., Orsatelli, P., Maurine, P., & Tria, A. (2012). Injection of transient faults using electromagnetic pulses-Practical results on a cryptographic system-. *IACR Cryptology ePrint Archive*, *2012*, 123.

[10] Hsueh, M. C., Tsai, T. K., & Iyer, R. K. (1997). Fault injection techniques and tools. *Computer*, *30*(4), 75-82.

[11] Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., & Whelan, C. (2006). The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, *94*(2), 370-382.

[12] Piret, G., & Quisquater, J. J. (2003). A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In *Cryptographic Hardware and Embedded Systems-CHES 2003* (pp. 77-88). Springer Berlin Heidelberg.

[13] Choukri, H., & Tunstall, M. (2005). Round reduction using faults. *FDTC*, *5*, 13-24.

[14] Yen, S. M., & Joye, M. (2000). Checking before output may not be enough against fault-based cryptanalysis. *Computers, IEEE Transactions on*, *49*(9), 967-970.

[15] Sung-Ming, Y., Kim, S., Lim, S., & Moon, S. (2002). A countermeasure against one physical cryptanalysis may benefit another attack. In *Information Security and Cryptology—ICISC 2001* (pp. 414-427). Springer Berlin Heidelberg.

[16] Lu, C. C., Tseng, S. Y., & Huang, S. K. (2005, March). A Secure Modular Exponential Algorithm Resists to Power, Timing, C Safe Error and M Safe Error Attacks. In *AINA* (pp. 151-154).

[17] Ziade, H., Ayoubi, R. A., & Velazco, R. (2004). A survey on fault injection techniques. *Int. Arab J. Inf. Technol.*, *1*(2), 171-186.

[18] Breuer, M. Abramovici MA, and Arthur D. Friedman. "Digital systems testing and testable design." *AT&T Bell Laboratories and WH Freeman* (1990).

[19] Patel, K. N., Markov, I. L., & Hayes, J. P. (2003, May). Evaluating circuit reliability under probabilistic gate-level fault models. In *Proceedings of the International Workshop on Logic and Synthesis* (pp. 59-64).

[20] Case, G. R. (1988, June). A statistical method for test sequence evaluation. In *Papers on Twenty-five years of electronic design automation* (pp. 338-341). ACM.

[21] Agrawal, V. D. (1981). Sampling techniques for determining fault coverage in LSI circuits. *Journal of Digital Systems*, *5*(3), 189-202.

[22] McNamer, M. G., Roy, S. C., & Nagle, H. T. (1989). Statistical fault sampling. *Industrial Electronics, IEEE Transactions on*, *36*(2), 141-150.

[23] Daehn, W. (1991). Fault simulation using small fault samples. *Journal of Electronic Testing*, *2*(2), 191-203.

[24] Mao, W., & Gulati, R. K. (1996, October). Improving gate level fault coverage by RTL fault grading. In *Test Conference, 1996. Proceedings., International* (pp. 150-159). IEEE.

[25] Hayne, R. J., & Johnson, B. W. (1999). Behavioral fault modeling in a VHDL synthesis environment. In *VLSI Test Symposium, 1999. Proceedings. 17th IEEE* (pp. 333-340). IEEE.

[26] www.si2.org

[27] Si2 OpenAccess API Tutorial

[28] www.peardrop.co.uk

[29] Garcia, R. (2001). Rethink fault models for submicron-IC test. *Test and Measurement World*, *21*(12), 35-46.

[30] 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.

[31] www.verific.com

[32] www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler

[33] www.cadence.com

[34] www.encyclopedia.com

[35] www.cad.polito.it/downloads/tools/itc99.html

[36] www.wikipedia.org/wiki/Intel_80386

[37] www.wikipedia.org/wiki/VIPER_microprocessor