THESIS TO OBTAIN THE DEGREE OF DOCTOR
OF THE UNIVERSITY OF MONTPELLIER

In Automatic and Microelectronic Systems

Doctoral school: Information, Structures, and Systems

Research Unit: LIRMM, University of Montpellier

Body biasing fault injection: modeling

Presented by Geoffrey Chancel

2023-11-14

Under the supervision of Philippe Maurine and Jean-Marc Gallière

Thesis Committee:

Philippe Maurine , Associate Professor ?? , University of Montpellier	Thesis Director
Jean-Marc Gallière, Associate Professor ?? , University of Montpellier	Thesis Supervisor

Abstract

Résumé de la thèse

Acknowledgements

Before getting started, I would like to express my gratitude to all the people who have given their support and made this thesis possible.

First, I wish to thank my thesis director, Philippe Maurine, and my thesis supervisor, Jean-Marc Gallière, for their guidance and support throughout these three years. Thanks to their advice, knowledge...

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-19-CE39-0008 (project ARCHI-SEC). They also acknowledge the French Ministère des Armées – Agence de l'innovation de défense (AID) under grant ID-UM-2019 65 0036.

Contents

List of Figures

List of Tables

List of Acronyms

AES	**A**dvanced **E**ncryption **S**tandard
BBI	**B**ody **B**iasing **I**njection
BSIM	**B**erkeley **S**hort-channel **I**GFET **M**odel
CPS	**C**yber-**P**hysical **S**ystem
DES	**D**ata **E**ncryption **S**tandard
DoM	**D**ifference **o**f **M**eans
DFA	**D**ifferential **F**ault **A**nalysis
DPA	**D**ifferential **P**ower **A**nalysis
DW	**D**ual-**W**ell
ECC	**E**lliptic-**C**urve **C**ryptography
ESD	**E**lectro**s**tatic **D**ischarge
EMFI	**E**lectro-**M**agnetic **F**ault **I**njection
FAM	**F**ault **A**nalysis **M**apping
FIB	**F**ocused **I**on **B**eam
FSA	**F**ault **S**ensibility **A**nalysis
FSM	**F**ault **S**usceptibility **M**ap
GFI	**G**litch **F**ault **I**njection
HFI	**H**ardware **F**ault **I**njection
IoT	**I**nternet **o**f **T**hings
LFI	**L**aser **F**ault **I**njection
PCC	**P**earson **C**orrleation **C**oefficient
PLL	**P**hase **L**ocked **L**oop
RAM	**R**andom **A**ccess **M**emory
RSA	**R**ivest **S**hamir **A**dleman
SCA	**S**ide **C**hannel **A**ttack
SCS	**S**tandard **C**ell **S**egment
SMA	**S**ub**M**iniature version **A**
SPA	**S**imple **P**ower **A**nalysis
TW	**T**riple-**W**ell
WLCSP	**W**afer-**L**evel **C**hip-**S**cale **P**ackaging

General introduction

Over the past years, various fault injection methods, representing a significant threat for secure integrated circuits, have been extensively studied, like laser fault injection (LFI), or more recently electromagnetic fault injection (EMFI). The purpose of these studies is to propose efficient countermeasures at the right cost. They have had multiple objectives, such as understanding the various phenomena at the origin of fault creation, or being able to simulate fault propagation over multiple abstraction levels.

Voltage pulse substrate fault injection, commonly called Body Biasing Injection (BBI), while being contemporary to EMFI, led to very few researches and studies in comparison. Up to the best of our knowledge, three scientific papers existed at the beginning of my thesis, back in 2020.

The LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier: Computer Sciences, Robotics and Microelectronics Laboratory of Montpellier), inventor of this technique in 2011, proposed this thesis to answer various questions such as:

- What are the phenomena at work leading to fault injection?
- What kind of spatial resolution does BBI offer?
- What is the time resolution of this method?
- Is it relevant to thin the silicon substrate of BBI target ICs?
- Can constraining fault attacks be performed with this method?

These questions have guided my thesis work through the last three years. This has led me to propose various improvements for the practice of BBI, in addition to introducing a CMOS integrated circuits simulation flow for a BBI context. My thesis manuscript is structured in five chapters. Each one of them attempts to provide answers to the preceding interrogations.

The first chapter of this manuscript provides an overview of the existing fault injection techniques, such as power glitch fault injection, electromagnetic fault injection or esle laser fault injection, with a particular emphasis on body biasing injection.

The second chapter describes the improvements I propose for the practice of BBI. These improvements have been conceived and obtained through my studies concerning BBI resolution and accu-

racy, both in time and space. Additionally, this chapter describes the practical results of a differential fault attack requiring single-bit faults performed thanks to BBI and the platform improvements.

The third chapter is dedicated to presenting a CMOS integrated circuits simulation flow under BBI. It introduces the established simulation models, in addition the designed algorithms allowing to simulate circuits subjected to BBI. The models and methods introduced allow me to simulate circuit behavior in reasonable duration, which allows me for instance to perform parametric analysis of BBI effects.

The fourth is dedicated to the understanding of fault creation in circuits subjected to BBI. It presents a follow-up of the simulation flow presented in the third chapter, allowing me to appreciate the actual BBI effects on logic gates. Then, it allows deriving a fault model from the simulations.

The fifth chapter discusses a common practice in fault injection methods: the thinning of integrated circuits substrate. While this topic has been extensively addressed concerning LFI, it is not the case for BBI. It relates to studying IC behavioral differences and BBI efficiency on different substrate thicknesses circuits. Various models are introduced to get different approaches, allowing to predict differently electrical and physical phenomena at work. Mathematical models are also derived from the previous models, enabling the calculation of optimal experimental parameters, in addition to predicting circuit behavior.

Eventually, the last chapter presents a general conclusion of my thesis work. In addition to this, outlooks are provided. The latter are interrogations remaining unanswered by my thesis works, mostly concerning more specific BBI effects on integrated circuits, such as FLASH memory modification or RAM fault injection.

I

Introduction and state of the art

Contents

1.1 Summary

This chapter reviews the state-of-the-art concerning fault injection methods. It first defines the interest of studying fault injection and its context. Then, various fault injection techniques are presented and their differences, advantages and disadvantages are analyzed. Specifically, platforms equipment across all methods is described alongside the different techniques employed to perform such fault injection. Eventually, body biasing injection is introduced, and we will study its interests in a fault injection context.

1.2 Introduction

In our time, almost every business sector and every part of our surroundings, directly or indirectly, use integrated electronics circuits. It ranges from smart-cards to supercomputers, through military devices, cell-phones, Cyber-Physical Systems (CPS) and Internet-of-Things (IoT) objects to name but a few.

Traditionally, integrated circuits design mainly focused on performance upgrades over the generations. Performance was measured thanks to two factors: computation speed and silicon surface. Within this context, power consumption was not a design constraint, therefore, integrated circuits became more and more energy-consuming. However, with the advent of portable devices, power consumption became a predominant design factor over speed and space, and it got included into the former design flows. Nevertheless, less space and more speed does not physically equate with less energy. Alongside, new systems have emerged and have massively grown these past decades: some fall in the field of CPS, while others lies in the field of IoT. On the one hand, CPS are often systems where hardware and software are interlaced and thought together, and can be drastically different from one application to another. On the other hand, IoT systems have often less coordination between hardware and software, but are commonly more flexible.

Whatsoever, both of these systems have something strong in common: their security is fundamental. Therefore, in this context, as it has been proposed in [4], and because security had been adopted as a countermeasure after the design flow, it had to enter as a fourth design rule when creating integrated circuits. This is required because a secure system has to ensure that every data going in and out of it are subject to the following criteria:

- Authenticity: data received have to come from the sender
- Integrity: data cannot be altered in any way
- Confidentiality: data cannot be accessed (read or written) by third-parties

Consequently, it is imperative to study and comprehend the strategies for enhancing IC security in order to develop future integrated circuits that are designed with security in mind from the initial stages of development to its completion.

Currently, electronic devices implement security in two distinct ways, namely from a software or hardware standpoint. To ensure security, encryption algorithms have been integrated in integrated circuits. It is possible to distinguish two distinct categories of encryption algorithms, namely symmetric and asymmetric algorithms.

In short, symmetric cryptographic techniques use a unique key for encrypting and decrypting messages. The most popular algorithms are the AES (Advanced Encryption Standard) [5], DES (Data Encryption Standard) [6], IDEA (International Data Encryption Algorithm) [7], RC5 (Rivest Cipher 5), and TDES (Triple DES) [8], not to cite them all. The key must be kept confidential and only shared among parties in order to maintain a confidential connection between them. The requirement for a single key is the main drawback of symmetric encryption methods. As a result, every possible step must be taken to safeguard key secrecy, such as avoiding key exchanges on public networks. However, symmetric encryption has a clear advantage over asymmetric encryption. As a result of utilizing a single key, symmetric algorithms are typically simpler than asymmetric algorithms, resulting in a reduction in computing power required for encryption and decryption. It is therefore possible to encrypt a large amount of data in a short amount of time.

In contrast, when it comes to symmetric cryptographic techniques, commonly referred to as public key cryptography techniques, a pair of keys is employed. The keys are usually referred to as public-key and private-key. The public key is used to encrypt a message, and anyone can use it. The private-key is, however, kept confidential to ensure that only authorized parties can decrypt a message that has been encrypted with the public-key. The primary motivation behind having two keys is that it is impracticable to reconstruct the public-key from the private-key. The most commonly employed asymmetric algorithms include the RSA (Rivest–Shamir–Adleman) [9] algorithm, the ElGamal encryption system [10], the ECC (Elliptic-curve cryptography) [11], and the Cramer-Shoup system [12], to name a few. The main drawback of symmetrical algorithms is that they involve large mathematical calculations, which implies a higher time complexity. Hence, these techniques are capable of encrypting a limited quantity of data. Therefore, to achieve this objective, in the majority of systems, a hybrid approach is employed, involving both encryption methods, thereby ensuring optimal security and brief ciphering times.

On the one hand, if all the previously mentioned algorithms are mathematically reliable, their reliability decreases when they are implemented on actual integrated circuits. Indeed, when an IC operates, it alters the behavior of some physical quantities according to the data being processed. Thus, it inevitably and unintentionally communicates information to the surrounding environment through these quantities. Among them, one can identify the electric current, and therefore the resulting electromagnetic field. In a normal context, where the data is not secret, it is not a problem. However, as soon as the data security is a major concern, the IC communicates to its surrounding data information through the various physical quantities. This is called leakage. An adversary can therefore measure these leakages to retrieve confidential data thanks to mathematical and statistical tools. These methods are called **"side-channel attack" (SCA)**.

On the other hand, physical quantity measurement is not the only flaw in actual algorithm im-

plementations. In fact, every physical IC has specifications under which it can operate properly. It includes temperature, clock frequency, power supply voltage, and the electromagnetic environment. When pushed beyond its specifications, any integrated circuit exhibits unpredictable behavior. However, it is still possible to control the behavior of an IC outside its specifications with a certain degree of success. By doing so, it is possible to run the calculations incorrectly by finely controlling how much time and by which amount the IC is outside its specifications, thus enabling, with specific mathematical algorithms, to retrieve hidden data manipulated by the IC, like secret encryption keys or sensitive data. This process is commonly referred to as a **"fault injection attack"**.

I have identified two potential hardware threats on robust algorithms that have been implemented into actual integrated circuits. However, it is customary to categorize cyberattacks into three distinct categories based on their execution methods.

Despite being technically advanced, noninvasive attacks are the most materially trivial. SCA are included in this set, which do not require any hardware modification to the targeted ICs. It is a delicate task to detect them; hence, they are deemed to be highly dangerous and are commonly considered in the initial stages of designing integrated circuits.

Then, it is possible to distinguish semi-invasive attacks. Systematically, they are accompanied by device physical preparation, which was entirely devoid of noninvasive attacks, but they are not accompanied by device physical modification. ICs integrity is therefore fully preserved. A typical IC modification involves the removal of the chip package. It enables access to either the front or back side of the integrated circuit, thereby facilitating micro-probing, laser injection, or substrate pulse injection. Furthermore, substrate thinning is also commonly considered and used, as it facilitates the fine-tuning of certain fault injection techniques, such as laser fault injection (LFI). These attacks necessitate specialized hardware, tools, and expertise and are frequently challenging to establish and execute.

Eventually, there are invasive attacks. They imply further physical modifications to integrated circuits. For instance, it is common to eliminate the layers of a chip, thereby enabling the photographing of the various layers and the reverse engineering of the target. A focused ion beam (FIB) can also be used to change the IC target internally by creating electric connections that did not exist before, or destroy existing ones. Invasive attacks often involve the definitive destruction of the target, primarily because physical integrity is not preserved during the process.

My doctoral thesis is dedicated to the study of a specific fault injection method: Body Biasing Injection. In this particular context, I examine in the next paragraphs the current state of the art in relation to side-channel attacks and fault injection techniques, as outlined in the literature. This allows me to explain the interests of the current work regarding hardware security.

In the first place, I briefly discuss side-channel attacks. Then, I examine the various fault injection platforms commonly described. Eventually, I ponder the interests of BBI in this context.

1.3 Side-channel attacks

1.3.1 Timing attacks

The first side-channel attack was initially introduced in 1996 [13]. This attack involves determining the duration required to execute cryptographic computations. By executing this method, the adversaries are able to obtain a variety of algorithmic keys, specifically for the RSA and ECDSA [14] algorithms. The computation cost of this attack is low, thereby enabling it to execute swift attacks. Indeed, as per the RSA algorithm, as outlined in [9], the encryption of a message necessitates the calculation of the following relationship:

$$C \equiv M^e (\mathrm{mod}\ n) \tag{1.1}$$

M denotes the message to be encrypted, while C is the ciphertext and (e, n) the encryption key pair. The objective of the attack outlined in [13] is to retrieve e. To achieve this objective, the integrated circuit must perform multiple computations of the equation 1.1 for varying values of M, while maintaining identical values of e. Subsequently, the attacker must evaluate the duration of each computation. After the demonstration of this attack, countermeasures were implemented, including the implementation of constant-time cryptographic algorithms allowing the elimination of leaks through the utilization of timing analysis. More recently, other, more advanced countermeasures have also been proposed [15].

1.3.2 Power analysis and electromagnetic analysis attacks

Subsequently, more elaborated side-channel attacks were explained in 1999, as documented in [16]. This paper presents the concepts of simple power analysis (SPA) and differential power analysis (DPA).

On the one hand, SPA entails the measurement and direct interpretation of power consumption traces of a cryptographic integrated circuit. For instance, it enables the counting of DES or AES rounds to gain insights into the utilized implementation. Furthermore, it allows for the observation of power consumption variations depending on the executed instruction. A proposal has been made to prevent the utilization of secret keys or information during conditional branching logic, with the objective of preventing simple power analysis.

On the other hand, DPA is a more elaborate approach that aims to identify the effects and variations associated with data processed by ICs. The aforementioned variations are more subtle and frequently obscured by noise. Therefore, DPA proposes to use statistics tools to reveal hidden system information, specifically by computing the difference of means (DoM) between traces. Therefore, preventing DPA is more complicated than preventing SPA. One of the simplest methods, which is not sufficient, is to add electrical noise, which acts as a masking layer. Another technique is to reduce measurable signal amplitude. It is done first by optimizing code execution, by finely choosing

which operation is performed to reduce electromagnetic leakage. Second, it is also possible to shield the device, but it increases the IC's cost significantly. Eventually, masking countermeasures are also employed, which consists in randomly splitting sensitive variables during the cryptographic process [17].

In addition to these attacks, there is also another attack which is commonly studied: correlation power analysis (CPA) [18]. As well as DPA, CPA uses statistical tools. However, as opposite to computing the difference of means, it involves calculating the Pearson or Spearman correlation coefficient (PCC), allowing to measure the linear monotonous correlation between different power consumption traces.

It is important to note that SPA, DPA and CPA are historically performed using traces directly measured from the ICs power consumption. However, these attacks can also be performed thanks to IC electromagnetic radiation analysis [19]. Because electric charges are circulating into the IC, they inevitably generate electromagnetic waves. Therefore, it is possible to pick up these waves, and similar to power consumption, their shape depends on the data being processed. There has been numerous active research concerning this method for twenty years. It can be explained thanks to its advantages compared to bare power consumption analysis. Indeed, when measuring the entire power consumption of an IC, it is not possible to target a specific area. It leads, especially with complex ICs and countermeasures, to an impossibility to perform such attacks. On the contrary, electromagnetic analysis attacks have multiple advantages over power consumption analysis attacks:

- Reduced to no sample preparation;
- No physical contact with the target;
- It requires only little and low-cost equipment: a probe and a voltage amplifier;
- Targets specific area of the IC.

As we stated previously, power consumption analysis attacks target an entire IC, whereas electromagnetic analysis attacks allow having fine resolutions. Indeed, small probes with a size down to 50 µm have been proposed [20]. Such small probes allow focusing the measurement on the cryptographic area of the IC, while excluding from the measurement, with a certain amount, any undesirable electromagnetic emission which could potentially harm the attack efficiency. In addition to that, electromagnetic probes, depending on their design, can have very high cutoff frequency. Therefore, it allows analyzing ICs running at high frequencies, enabling attacks on recent devices such as smartphones [21].

1.4 Fault-injection attacks

Fault injections are widely described in the literature and can be utilized for a variety of purposes. For instance, during integrated circuits testing, it is common to find fault injection susceptibility tests, allowing for engineers to test fault detection circuits, recovery capabilities and reconfiguration

possibilities of ICs. In this work, we are going to take a closer look at hardware fault injections (HFI) techniques solely, which fall in two distinct categories, similar to side-channel attacks:

- HFI with physical contact;
- HFI without physical contact.

For each kind of HFI, multiple outcomes are aimed. On the one hand, the HFI can produce, in the targeted IC, branching errors leading secret codes to be revealed or protected rights to be acquired by an attacker [22]. On the other hand, HFI can produce incorrect behaviors, allowing to retrieve hidden and protected data thanks to mathematical tools. In that case, HFI targets are mostly cryptographic algorithms, and can be segmented in non-comprehensive set of categories.

One of the most performed HFI is called differential fault attack/analysis (DFA). The principle of DFA lies in inducing computation errors during the decryption process of cryptographic algorithm thanks to fault injection. Several DFA were proposed on different algorithms [23, 24, 25, 26, 27]. Every DFA implies that the attacker has access to at least two ciphertexts, a correct one, denoted C, and a faulty one, denoted C_F. In addition to that, the attacker must also control the characteristics of the induced faults, such as the amount of faulted bits and in which operation they are faulted. Eventually, it is needed to be able to induce the expected faults depending on the fault model required for the DFA.

Another common HFI is the fault sensitivity analysis (FSA) [28]. As every HFI, it is still required to have physical access to the device. FSA usefulness comes from the fact that alongside fault characteristics, other information can be used by attackers, in that case: the IC sensitivity to faults. As defined in [28], fault sensitivity is a condition where the faulty output begins to show specific characteristics. Specifically, this work defines a critical condition, similar to the PLL capture ranges (lock-in, hold-in, pull-in, etc.), where the IC starts to exhibit a faulty behavior or when it stops this behavior. Then, to perform an attack with this information, the attacker has to know the relationship between the fault sensitivity and the computed data, without knowing the insights of the cryptographic algorithm at work. It states that the algorithm will inevitably exhibit data-dependency of fault sensitivity. Hence, it allows using the IC as an almost black box.

1.4.1 Fault-injection techniques

1.4.1.1 Glitch fault injection

Glitch fault injection (GFI) are one of the first historical documented fault injection attacks. They are simple and require little equipment. For the most part, they are non-invasive, which means that they are physically reversible. Various physical quantities can be disturbed, but the power supply voltages (VDD or GND), and the IC clock are the most common. Each physical quantity can be modified at the attacker's discretion, with a certain amount. However, the disturbances have to be short enough to avoid IC shutdown concerning power supply glitches, but also not powerful enough to avoid the

IC destruction, which is very similar to BBI. On the one hand, the main advantage of such attack is its easiness to set up compared to other methods. On the other hand, their main disadvantage is the complete lack of locality with the injection effects. Indeed, disturbing IC's macro-parameters interfere with the entire chip and does not guarantee a useful faulty behavior. In addition to that, every modern IC is prepared to detect such attacks and thus protect itself by resetting its electronics.

1.4.1.2 Laser fault injection

Laser fault injection (LFI), sometimes called optical fault injection, has been introduced in 2002 [29] and is a more complex technique than GFI. However, its precision is immensely better, at the cost of being semi-invasive, and in some cases invasive. LFI consists in targeting specific regions of the IC with laser beams of specific wavelengths. Several other parameters are involved for this method to succeed, such as the light emission duration, the area/volume of the targeted region, the IC substrate thickness, etc. Although LFI requires chip preparation, it is often minimal. LFI works thanks to the fact that every silicon semiconductor device (diode, transistor...) is intrinsically sensitive to light, typically with wavelengths ranging from 400 nm to 1000 nm. Therefore, if the light conveys enough energy, it is possible to change the state of some transistors, thus affecting logical values. The main shortcoming of LFI is the platform price. Add more details.

1.4.1.3 Electromagnetic fault injection

Electromagnetic fault injection (EMFI) is a more recent and more studied technique, introduced in 2002 [30]. Its principle is basic: an electric current in a wire (probe) near an IC creates a corresponding electric current in the IC power delivery network, like to an electric transformer. Similar to GFI, the attack can be non-invasive, although this method yields better results while being semi-invasive. Indeed, the closer the probe is to the IC, the better the coupling and the mutual inductance are, which often requires removing the IC's plastic package. The method then consists in applying pulses into the probe, which will be mirrored by a certain amount in the IC power delivery network. This injection technique efficiency greatly varies depending on the probe's characteristics, the IC transistors size, clock frequency, the targeted location, or the pulse duration. Over the time, electromagnetic probes were constantly improved, and for instance, it is now common to find probes with a ferrite core, allowing for better injection locality. In 2020, a modeling workflow was proposed [2], allowing to explain how EM probe can couple to IC power delivery networks. When compared to LFI, EMFI is a lot less expensive, and require less accuracy to yield good results. Indeed, in an EMFI platform, the probe, if self-made, cost around €5 to €15 depending on the hardware used. In addition to that, cheap pulse generator can be manufactured by hand, such as very simple high-voltage transformers driven with a transistor. However, this method has shortcomings and compromises. On the one hand, as the probe is coupled with the power delivery networks, the smaller these networks are, the more transistors are targeted. On the other hand, the smaller the probe, the higher the voltage pulse has to be to produce a similar effect.

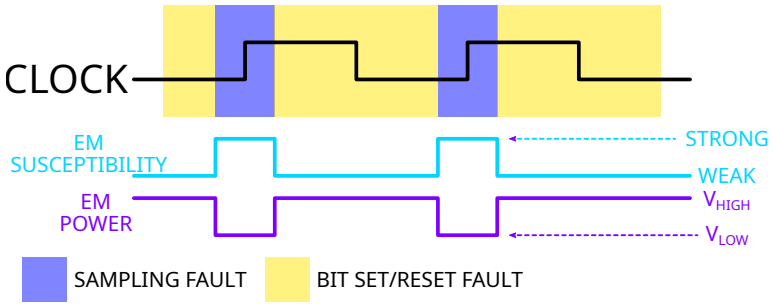


Figure 1.1: *PLACEHOLDER.*

A sampling fault model for EMFI was proposed in 2015 [31]. It takes on the fact that EMFI allows modifying enough the amplitude at the inputs of multiple DFFs, such as the data input, the clock, the set or the reset signals, at a specific time: during the sampling pulse edge. It means that either setup or hold timing constraint is violated, resulting in an incorrect data sampling from the input to the output. A schematic explanation of the sampling fault model proposed in [31] is illustrated in Fig. 1.1. According to this model, the faults appearing using EMFI are related to the IC clock, shown in black in Fig. 1.1. In addition to this is shown the EM susceptibility over two clock periods, in cyan, which is, in that case, equivalent to the DFF susceptibility: the fault probability is higher during the rising edge, as it is the time when the DFF sample the input. Therefore, during the rising edges, less power is required to induce a fault, which leads to the EM power curve in purple, in between two thresholds: V_{LOW} and V_{HIGH}. V_{LOW} is the minimal voltage required to inject faults during the DFF switching process, and V_{HIGH} is the threshold allowing to create bit set or reset, even outside the DFF susceptibility windows.

1.4.1.4 Body biasing injection

Eventually, there is another fault injection method, less studied and more recent than the others, commonly called Body Biasing Injection (BBI), which is the research topic of my thesis. This technique has been introduced in 2012 [32], and further studied in 2013 [33] and 2016 [34]. At the beginning of my thesis, a fourth article was published [1], studying the interests of BBI concerning Wafer-Level Chip-Scale Packaging (WLCSP). The principle behind BBI is fairly simple: applying voltage pulses directly onto the backside of IC targets, thanks to a metallic probe. The targets have to be prepared in a very similar way to LFI, with a partial removal of the package on the backside to give access to the substrate, and a potential substrate thinning. On the one hand, despite this simple premise, in the vast majority of cases, BBI is a semi-invasive method. Indeed, as most IC are encapsulated in a ceramic or plastic package, and because the chip preparation is similar to LFI, it is required, to access to the substrate, to partially remove a piece of the package. However, in some rare cases where Wafer-Level Chip-Scale Packaging (WLCSP) is used, there is no need to remove the plastic package as there is none. On the other hand, building a BBI platform is not expensive and technically easier when compared to LFI or EMFI. Indeed, a metallic probe with a custom armature costs around 10 euros at worst, and is easy to build at hand, while manufacturing a precise EMFI probe requires more knowledge to achieve good results, and often requires the use of flexible PCBs. Considering

that EMFI and BBI both require similar voltage pulse generator, which is often the most expensive piece of equipment, the overall platform cost is lower concerning BBI.

II

Body Biasing Injection platforms and good practices

Contents

2.1 Summary

This chapter first presents the various existing BBI platforms in the state-of-the-art, in addition to introducing my thesis BBI platform. Then, I introduce improvements over the traditional platform used, allowing for better reproducibility and control over BBI parameters when compared to state-of-the-art platforms. At the beginning of this work, I was working with a state-of-the-art like platform, and quickly noticed how difficult it could be to conduct reproducible experiments in various cases. This led me to elaborating the enhancements I propose in this chapter, which helps in reducing the variability of some parameters. In the fist place, I present a theoretical explanation of how the modifications could improve the platform. Thanks to these improvements, I was able to draw better experimental results and compare them to results obtained using state-of-the-art platforms, in addition to verifying the soundness of such platform modifications. These experimental results are presented in the last part of this chapter thanks to elementary electrical experiments, followed by a differential fault attack, that I managed to perform thanks to the enhancements on a hardware AES coprocessor. Parts of these works were published in FDTC 2022 [35] and FDTC 2023 [FDTC2023REF].

2.2 Introduction

In the first place, we are going to introduce Body Biasing Injection platforms:

- What exists in the state-of-the-art
- What I am using for my experiments

Afterward, I present a general BBI platform with its electrical model, in addition to evaluating the platform characteristics. Thanks to the model, I can perform electric simulations, allowing me to study and highlight its inherent flaws, such as:

- Poor control over the characteristics of the platform
- Obvious ringing leading to poor temporal accuracy
- Platform dependent parameters such as the ground installation quality
- Main physical quantities, such as the voltage and the pulse width, set points not met

Thereafter, I propose enhancements to overcome the previous platform shortcomings, which are:

- Matching the output impedance of the generator to reduce the ringing and bring the measurements closer to the specifications and the set points
- Bypassing the grounding installation to minimize platform dependency

After that, I present a deeper analysis of these enhancements, including ringing, set points accuracy, and load and transmission line dependency. Then, I discuss various techniques allowing to match the generator output impedance, in addition to introducing practical grounding installation bypass. Next, I perform actual experiments with our BBI platform, including measurements of such platform, illustrating the enhancements in practice. Eventually, I introduce a constraining differential fault attack set-up with our platform. It includes the attack description, followed by a thorough description of the IC target, sustained with experiments allowing me to perform the attack with more ease, with a comparison of a state-of-the-art platform with our enhanced platform, ended up by the attack results.

2.3 BBI platforms in the state-of-the-art

2.3.1 Initial BBI platforms

First introduced in 2012 by P. Maurine et al. [32], further studied in 2013 by K. Tobich et al. [33], the proposed BBI platform in both papers is fairly simple, similar to EMFI platforms, composed of:

- A decapped IC, with its backside accessible;
- An independent voltage pulse generator able to generate positive and negative voltage pulses up to 100 V, with a maximum current of 2 A. The generator is DC-coupled with the load;
- A passive custom-made probe, consisting of an SMA connector and a standard needle soldered to it;
- A positioning system to place the probe precisely onto the IC backside;
- An acquisition system, measuring various voltages.

It is important to remark that the probe is connected through a relatively long interconnection, acting as a transmission line.

2.3.2 C. O'Flynn BBI platform

The original platform had stayed identical in the literature [34], until C. O'Flynn published in 2020 [1] practical examples of BBI attacks on WLCSP integrated circuits. In this work, the platform is structured differently. There are common elements, such as:

- An IC target with an accessible backside, in that case thanks to the WLCSP;
- A positioning system;
- Various acquisition tools.

The structural differences concern the voltage pulse generation. Instead of an independent voltage pulse generator, connected to a passive probe through a transmission line, the proposed solution consists in implementing an active probe with a separate pulse trigger generator.

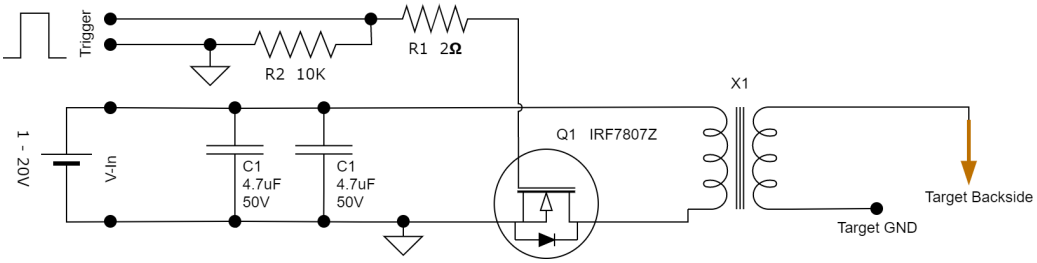


Figure 2.1: *Schematic extracted from C. O'Flynn [1]: BBI injection device proposed by C. O'Flynn [1], using a transformer to produce high voltage pulses from a low voltage power supply.*

Fig. 2.1 shows the design extracted from C. O'Flynn work [1]. The transformer allows creating high voltage pulses from a low voltage power supply. The transformer is controlled through the transistor Q1. Because the output is the secondary of a transformer, it is AC-coupled to the load, thus, no DC current can be transferred to the load. The transformer is custom-made and allows for ten times voltage multiplication, therefore enabling 300 V pulses with a 30 V power supply unit (PSU), which is fairly common for a lab PSU. The transistor is controlled thanks to an external trigger pulse, generated by another piece of equipment on this platform. It is on this point that O'Flynn's platform greatly differs from Maurine's initial platform.

The pulse generator used in this paper is a ChipWhisperer-Lite, an open-source tool created by NewAE Technology Inc. This tool can perform various tasks, such as pulse generation (as it is currently done), analog signals capture, or clock generation, enabling clock glitch fault injection. In addition to that, it can act as a simultaneous capture and target board, which is of great use in a BBI context.

2.3.3 Commercial platforms

In addition to documented research BBI platforms, there are multiple commercially available solutions. We are going to address the most noteworthy in the current section.

2.3.3.1 Langer EMV-Technik GmbH BBI platform

The German society "Langer EMV-Technik GmbH" proposes a ready-to-use BBI platform. It is composed of two main hardware components:

- A BBI current pulse generator, illustrated in Fig. 2.2;
- A "Burst Power Station", which is the combination of a power supply and a controller allowing to control and monitor every probe sold by the company, with a provided software.

Figure 2.2: *BBI probe proposed by the company Langer EMV-Technik GmbH.*

The core design is similar to the state-of-the-art platform, the main difference being that the system commercialized by Langer is marketed as being a current source instead of a voltage source. However, in practical operation, it does not represent a significant difference, since the major difference between a current source and a voltage source is their output impedance. Thus, one can either perform the BBI experiments with both electrical sources without much distinction, as long as the attacker is aware of these characteristics. The probe is specified with the following characteristics:

- A maximum allowable current of 4 A in a 1 Ω load;
- A rise time inferior to 2 ns;
- A maximum pulse repetition frequency of 20 kHz;
- Positive and negative polarities;
- The possibility to delay the pulse command thanks to their control module;
- A jitter of ± 1 ns;
- A pulse width of 2 ns at full power, and of 4 ns at minimum power;
- A trigger delay ranging from 70 ns to 420 ns.

According to the product's datasheet, containing actual measurements of the probe, the minimal intensity allows injecting at peak approximately 2.4 A in 1 Ω. However, contrary to the open-source ChipWhisperer-Lite, there is very little official documentation about their products, thus reducing the available knowledge.

2.3.3.2 Riscure BBI platform

Similar to Ledge, Riscure proposes a quite complete BBI platform. It is composed, as before, of two major tools: a pulse generator and a metal probe. In that case however, the probes are passive ones. The generator, called "EM-FI Transient Probe", originally designed to be used in conjunction with EMFI probes, has the following characteristics:

- A maximum output voltage of 450 V ± 45 V;
- A maximum probe current (with 0 Ohm impedance) of 64 A;

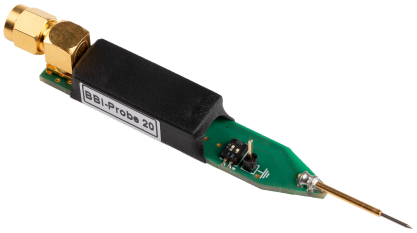


Figure 2.3: *BBI probe proposed by the company Riscure BV.*

- A pulse width at half maximum output of 50 ns at full power (there is no mention of a controllable width);
- A trigger latency of 50 ns.

On the other hand, they propose four different BBI probes, one of them being illustrated in 2.3 which, in an odd way, are specified for different polarity and voltage amplitude depending on the model:

- A positive 200 V $\pm$ 40 V with 15 ns pulse width;
- A positive 33 V $\pm$ 6.6 V with 12 ns pulse width;
- A negative 37 V $\pm$ 7.4 V with 20 ns pulse width;
- A negative 200 V $\pm$ 40 V with 23 ns pulse width.

They include an SCS connector and a spring-loaded metal tip to avoid damage to the IC backside. These BBI probes are meant to be used with the "EM-FI Transient Probe" pulse generator. Eventually, they provide a software to control their equipment.

2.3.3.3 NewAE Technology Inc. generators

Figure 2.4: *Pulse generator for EMFI and BBI proposed by NewAE Technology Inc.*

Eventually, NewAE Technology Inc. proposes various products for the practice of EMFI and BBI. Most of them can be used with one or the other fault injection method. I am going to cover two of them in this subsection.

The first one being the ChipSHOUTER®, with the pulse generator having the following characteristics:

- A voltage range comprised between 150 V to 500 V;
- A pulse width ranging from 15 ns to 480 ns depending on the connected load;
- A charge rate of 35 V/ms;
- An input jitter of 220 ps;
- A trigger latency of 50 ns;
- Python libraries allowing to interface and control the device;
- Monitor outputs allowing the user to probe internal signals.

This product costs around €4000 and the electrical schematics are available for free. It is a medium cost alternative compared to other equivalent pulse generator suh as the AVTECH AVRZ-5W-B from Avtech Electrosystems Ltd., the base model costing around $15000.

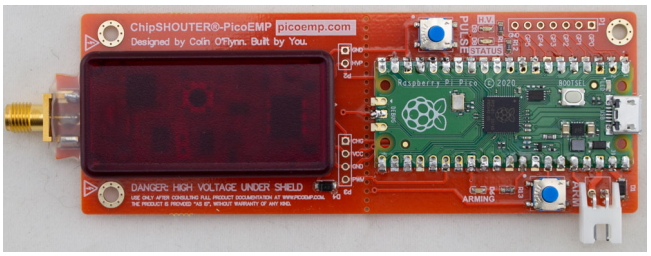


Figure 2.5: *PicoEMP: a low-cost pulse generator from NewAE Technology Inc., costing around 94 % less than a typical generator.*

The second product from NewAE is a very low-cost device: the PicoEMP. It is an open-source device, where safety and cost where the two main design rules. The tool is community maintained, and while originally designed for EMFI in mind, it was very recently studied concerning BBI in 2023 [36] by one of its contributors. A photograph of the device is shown in Fig. 2.5. Thanks to the low-cost design approach, the bill-of-materials for this tool is roughly equal to 50 $., which makes it very accessible for anyone to build it from scratch. Its main characteristics and drawbacks are the following:

- It uses a transformer to generate high voltages, therefore no DC voltage option is available at its output;
- The output transformer is low-power, around up to 200 mW;
- The recovery time is slow, measured between 1 to 4 seconds depending on operating conditions;
- The maximum voltage pulse is of approximately 250 V;
- A pulse width of about 85 ns in 50 Ω;
- There is no pre-calibration;
- It does not allow pulse width control by default. However, it is possible through drive signal control, even though being less accurate.

2.3.4 An overview about BBI platforms

From what I described previously, there are many different platforms and tools available for the practice of BBI. Their characteristics greatly vary from one platform to another, but they share a common ground, allowing to distinguish many tools and equipment constituting a typical BBI platform, which are the following:

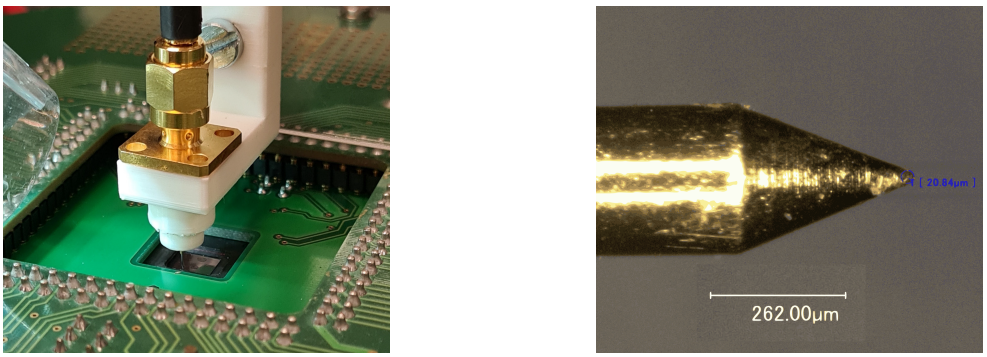
- A metallic probe, allowing to make an electrical contact with the target backside, preferably spring-loaded;
- A voltage pulse generator capable of generating very high, short and precise pulses;
- A 3D positioning table, with a precision high enough for the application, to move the probe precisely with correct pressure on the backside;
- A preferably vibration-proof table to minimize probe physical jitter due to vibration caused by other equipment or natural vibration;
- A high precision oscilloscope to measure various physical quantities that might help to practice BBI.

Some of these tools are not necessary to constitute a BBI platform, such as the positioning table, but they greatly simplify the platform reliability and reproducibility.

2.4 Our BBI platform

With these tools in mind, we are describing in this section the platform we are using in our lab in detail.

2.4.1 The probe



(a) *Custom BBI probe in mechanical contact with the IC backside.*

(b) *Custom BBI probe microscopy physical measurements.*

Figure 2.6: *Custom BBI probes photographs*

The most distinctive piece of equipment when working with BBI compared to other fault injection methods is the electrical probe. As we have seen before, it is commonly made with a metal tip, a

connector of any sort and a mechanical support to hold the structure together. Some can be active, while others are passive, and therefore less expensive. However, it is very easy to build one, and any needle size available on the market can be used depending on the needs. In the case of BBI, the probe is used to establish the electrical contact with the substrate of integrated circuits, the latter being poorly conductive, but not isolating. For this work, we designed a custom probe, allowing us to control its characteristics, around three simple parts:

- An SMA connector, to have a low-cost, small and standard interconnection available with almost every high-speed equipment;
- A spring-loaded metallic tip soldered onto the SMA connector providing a better control over the applied pressure onto the backside;
- A custom 3D-printed support holding the parts together, shaped to fit with our other tools.

Fig. 2.6 shows detailed pictures of the probe we designed, with a photograph in operation in Fig. 2.6a, and a photograph under a microscope of the probe tip-end in Fig. 2.6b, allowing to measure its actual size before its first usage. The metallic probe we had chosen has a 0.635 mm diameter and is 16.35 mm long. The specified maximum nominal current of the probe is of 1.5 A, and the electrical contact resistance measures approximately 70 mΩ. The tip has a diameter roughly equal to 20 µm, and it is important to note that this value tends to increase when the probe is utilized, due to the physical contact and the pressure with the IC backside. The bill-of-materials cost for our custom probe tool is roughly equal to 20 $, ignoring manual labor to assemble everything together.

2.4.2 The generator

The other fundamental piece of equipment when practicing BBI is the voltage pulse generator. It is, generally, one of the most expensive platform tool, similar to EMFI. Indeed, because BBI relies on voltage pulses to disturb an IC, it is necessary to provide a precise control over the pulse parameters to the user, such as the voltage set point, the pulse duration, etc.

Figure 2.7: *Front side of the Avtech Electrosystems Ltd. AVRK-4-B High Voltage Pulser, used during all my thesis experiments.*

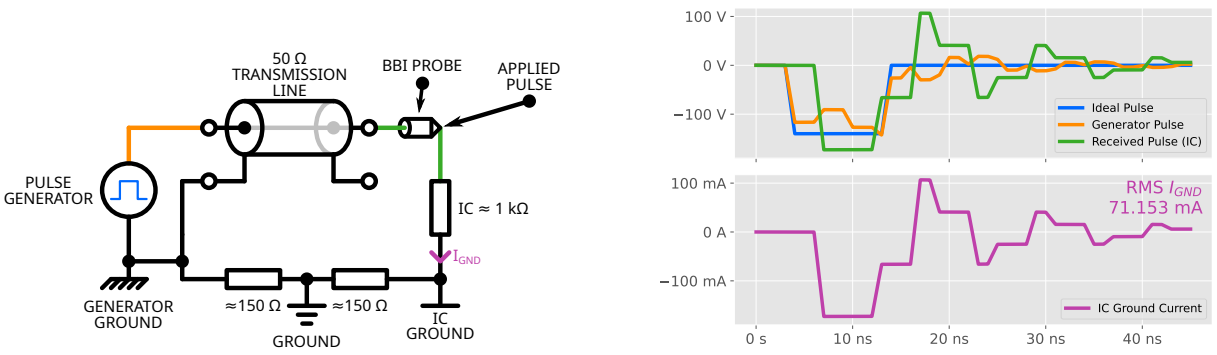
For my thesis, I am using a precise high speed and high voltage pulse generator to be able to finely study the voltage pulse characteristics effects on BBI, more specifically the AVRK-4-B from Avtech Electrosystems Ltd. It is shown in Fig. 2.7, and costs around $14500 in its most basic configuration. Similar to the low-cost generator described previously, it is commonly used for EMFI, but is also suitable for BBI. Its main specifications are the following:

- The voltage pulse amplitude is specified between 150 V and 750 V with positive and negative polarities. The generator can go below and above these thresholds, however, there is no guarantee of the set point value correctness;
- The pulse width is specified between 6 ns and 20 ns. Similar to the voltage, the generator can go down from 4.5 ns, up to 22 ns, but is not specified out of the default range;
- Rise time (resp. fall time) for positive (resp. negative) pulses is specified to be precisely of 4 ns. Fall time (resp. rise time) for positive (resp. negative) pulses is not specified and depends on the generator load characteristics;
- The recovery time is inferior to 1 ms, allowing a pulse repetition frequency up to 1 kHz;
- The minimal propagation delay measures 150 ns, and can be raised up to 1 s;
- The jitter measures $\pm$ 100 os $\pm$ 0.03 % of the propagation delay;
- The output is DC-coupled, allowing the generator to continue providing energy to the load (if resistive or inductive) during the pulse plateau;
- All the specifications presuppose that the generator is loaded precisely with 50 Ω.

2.5 BBI in practice

With actual BBI platform in mind, let me introduce a typical BBI platform model that we are going to study in details. These models allow us to concisely understand, evaluate and simulate BBI platforms behavior, limitations and room for improvement. We will therefore analyze the platform's performance and point out its weaknesses.

2.5.1 Typical BBI platform model



(a) *Model of a state-of-the-art BBI setup, with its key components.*

(b) *Simulation results of the state-of-the-art BBI platform model.*

Figure 2.8: *State-of-the-art BBI platform electrical model and its simulation results.*

To be able to quickly predict and analyze BBI platforms, I developed a very simple electrical model, illustrated in Fig. 2.8a. This model represents the key components of a BBI platform, which are:

- The voltage pulse generator;
- The transmission line, used to connect the probe to the generator;
- The BBI probe;
- The targeted IC, modeled by an electrical resistance;
- The grounding installation, consisting of electrical resistances connected between equipment grounding.

2.5.2 Platforms evaluation criteria

For the purpose of evaluating BBI platforms, we decided to focus on two important criteria, allowing to represent the platform quality:

- The characteristics of the voltage pulse measured at the generator output, allowing to observe how the generator behave when loaded with the transmission line and the IC;
- The characteristics of the target ground current waveform, allowing to monitor exactly what is actually injected into the IC.

2.5.3 Raw results

To that end, we will deeply analyze the platform's simulation results shown in Fig. 2.8b. Four signals are displayed, with their colors matching the colors in Fig. 2.8a for greater clarity. There are three voltage waveforms and a current waveform. The blue waveform is the ideal voltage pulse an attacker want to apply to the backside of an IC during a body biasing injection. Its characteristics are the following: a voltage set point of -140 V and a pulse width set point of 10 ns. It is a steep, fast and precise pulse with controlled rise and fall times, pulse width and voltage. However, when performing real experiments, which the model allow us to evaluate, this ideal pulse falls apart. It can be seen thanks to the orange and green waveforms, representing respectively the pulse observer at the generator output and the pulse effectively applied onto the backside of the IC target. There are multiple obvious observations that can be made concerning the received pulse (green) signal:

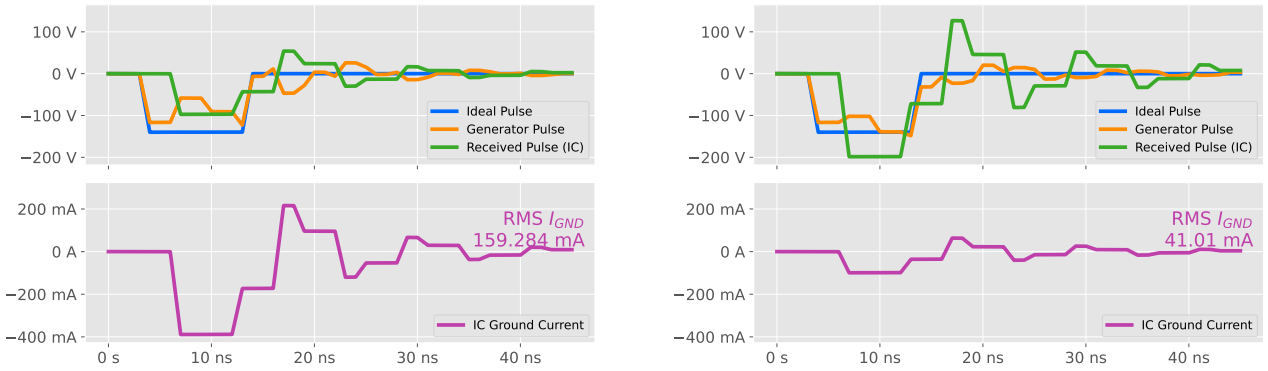
- The voltage set point is not respected, with a 23.5 % negative percentage overshoot (PO) on the falling edge, and a 107 % positive percentage overshoot (PO);
- There is obvious ringing, causing the pulse width to be longer than expected in addition to damped oscillations

These effects can also be observed on the IC ground current waveform (purple), as it is a mirror of the applied pulse due to the pure resistive nature of the IC in that model.

2.5.4 Analysis conclusions

In order to understand the implications of such observations, let us analyze each one of them.

The first important thing to note is that the various model numeric values are extracted from our actual platform. Therefore, the 150 Ω grounding and the 1 kΩ IC are average measured values of actual devices. Thus, these parameters, in addition to the transmission line characteristics, will inevitably vary with a certain amount from one platform to another.



(a) *State-of-the-art BBI platform simulation results with an IC load equals to 250 Ω.*

(b) *State-of-the-art BBI platform simulation results with an IC load equals to 2 kΩ.*

Figure 2.9: *State-of-the-art BBI platform simulation results with different IC load values.*

Indeed, the backside surface of an IC does not equal to a constant load. In addition to this, if the IC substrate is thinned, these values will change even more. Therefore, depending on the probe location and the IC substrate thickness, the generator might not see the exact same load. To illustrate the induced effects of such differences, I performed simulations with various IC values, representing typical measured values for my IC target when thinned down to 50 µm, up to more than 700 µm, and the results are shown in Fig. 2.9, both for a 250 Ω load (Fig. 2.9a), and a 2 kΩ load (Fig. 2.9b). In both cases, due to the non-zero generator output impedance, the latter forms a voltage divider with the IC load. On the one hand, with an IC load value one quarter lower, there is more current in it, while the applied pulse amplitude is 30% lower. On the other hand, with an IC load value two times higher, there is less current in it, while the applied pulse amplitude is 40% higher. It represents a 70 % range around the set point value, which is excessively high. However, in both cases, the ringing is still present with the same amount relative to the pulse amplitude.

Eventually, all of these observations allow us to spot three major flaws of such platform:

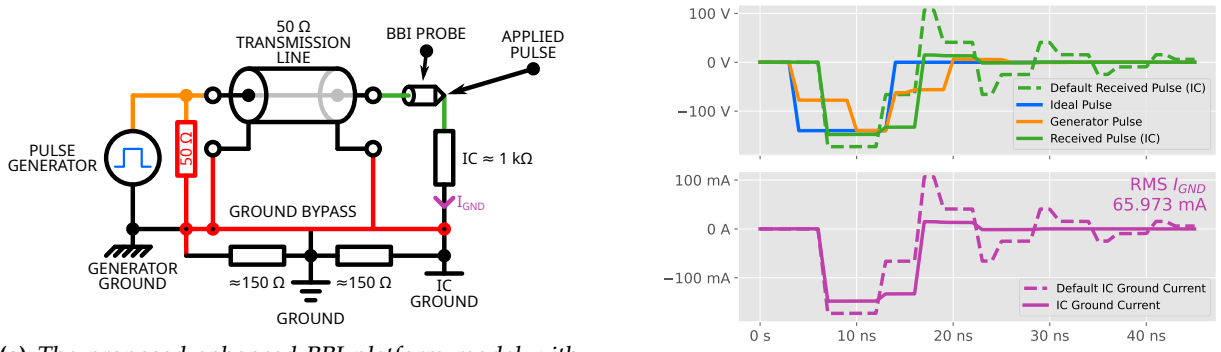
- The platform parameters are difficult to control, leading to unknown values concerning pulse width, voltage set point, etc.;
- It leads to a poor temporal accuracy, thus minimizing the chances to perform a precise and repeatable fault injection;
- At last, all parameters are platform dependent, leading to a low reproducibility rate, thus lowering the credibility of experiments performed on such platforms.

In this context, I present in the next section various simple improvements to the BBI state-of-the-art platform.

2.6 Enhanced BBI platform model and simulation

In this section, I propose platform enhancements over the state-of-the-art BBI platform previously introduced. These improvements aim at being low-cost, fast and easy to set up, to represent an interesting addition without drastically increasing the platform financial cost. Eventually, I am able to draw conclusions on such improvements thanks to simulation results.

2.6.1 Matching the generator output impedance



(a) *The proposed enhanced BBI platform model with its key components and upgrades.*

(b) *Simulation results of the enhanced BBI model.*

Figure 2.10: *Enhanced BBI platform model, alongside its simulation results.*

The first proposed improvement concerns the generated voltage pulse characteristics. As I have shown previously, the various set point parameters are not met. In a fault injection context, it is an undesirable behavior, as it is required to finely control the generated pulse to produce controlled disturbances into the targeted IC. Therefore, and because most high speed high voltage pulse generator are specified to be loaded with a precise impedance, I simply propose to connect a known load directly at the output of the generator. In my model, a 50 Ω resistor is loaded at the generator output, as illustrated in red in Fig. 2.10a. Thus, the generator sees the impedance network formed by the compensation load, the IC, the transmission line, and the grounding installation. However, because the grounding installation is platform dependent, it is required, in order to perform a better impedance matching of the generator output, to improve the grounding, which leads to the other platform enhancement described in the following section.

2.6.2 Improving the grounding installation

In many platforms, the grounding installation might be perfectly fine, and the following section may not apply to them. However, with our platform, we quickly observed that the grounding impedance

was far from negligible. Indeed, with an average IC impedance around 1 kΩ, and inter-equipment ground impedance around 150 Ω, it represents a 15 % increase in the total impedance seen by the generator. Therefore, in order to transfer a maximum amount of energy into the IC, especially in areas where the IC impedance might be closer to the grounding impedance, it is required to cancel as much as possible the latter.

To that end, I propose a very simple setup modification. It consists in keeping the platform as is, and adding short copper wires between equipment grounds. Therefore, they shunt the platform ground and creates a low-impedance path for electric charges, thus allowing the previous section approximate impedance matching to perform better.

2.6.3 Simulation results

To verify the soundness of the previously proposed enhancements, I performed simulations thanks to the model presented in Fig. 2.10a. The simulation results are shown in Fig. 2.10b.

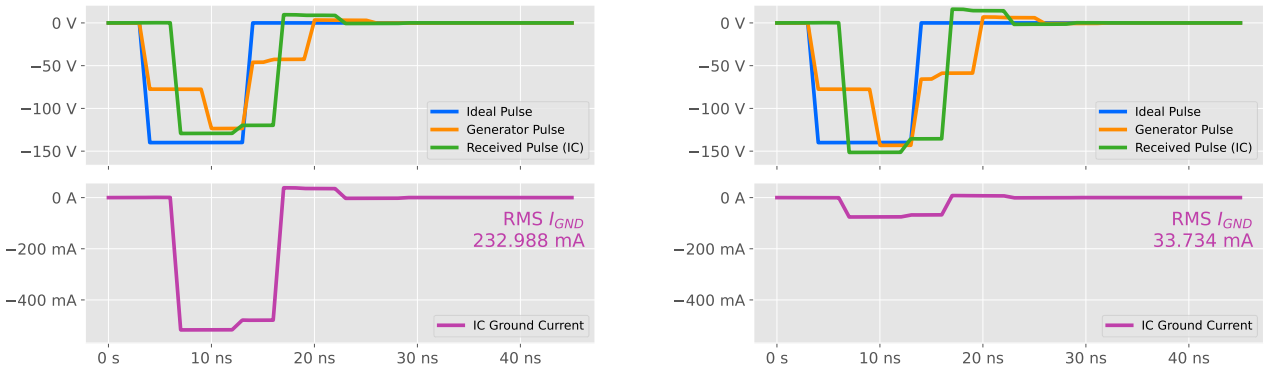
In that case, unlike in the state-of-the-art platform, the voltage set point is almost met concerning the received pulse (green waveform), with a slight undershoot of 6%. It is mirrored on the IC ground current waveform, where the ringing is drastically reduced, which leads to a steeper and more accurate pulse. It is especially noticeable when directly comparing the state-of-the-art waveforms in dotted lines. Concerning the generator pulse (orange waveform), it is still distorted as the ringing has not disappeared, but is less of a concern since the waveform of interest is the one effectively applied to the IC backside.

2.6.3.1 Load dependency

To further analyze the benefits of the proposed improvements, I performed, as for the state-of-the-art platform, additional simulations with various loads. As before, 250 Ω and 2 kΩ were chosen to have a common point of comparison. As I stated previously, these values are chosen to match the average typical value of my IC target when thinned down to 50 µ, up to 700 µm.

Fig. 2.11 presents the simulation results for such loads. For both loads, the measured voltage moves away from the set point by around 7 % in each case. It represents a 14 % range around the -140 V set point, which is immensely better than in the previous platform. It is still not perfect, but the platform is overall less dependent to the IC load, which is desirable in order to have repeatable voltage pulse across the entire IC backside.

Then, quite naturally, for the 250 Ω load, the current is higher than for the 1 kΩ, and with the 2 kΩ, it is lower. In addition to that, thanks to the 50 Ω resistor placed at the generator output, it reduces the range in which the effective load (the compensation load in parallel with the IC) changes. Indeed, it goes from around 42 Ω to about 49 Ω, instead of going from 250 Ω to 2 kΩ in the previous case. Eventually, in addition to all of the above, these enhancements have also drastically reduced ringing,



(a) *Enhanced BBI platform simulation results with an IC load equals to 250 Ω.*

(b) *Enhanced BBI platform simulation results with an IC load equals to 2 kΩ.*

Figure 2.11: *Enhanced BBI platform simulation results with different IC load values.*

which contributes to the applied pulse amplitude being closer to the set point.

2.6.4 Simulation conclusions

All of this leads to better control over the various platform parameters, allowing for more accurate and shorter pulses, closer to the expectations. In addition to that, the platform is less design dependent thanks to the minimization of impedance mismatch and poor grounding installation. It leads to a better time accuracy, enabling potentially more controllable fault injections.

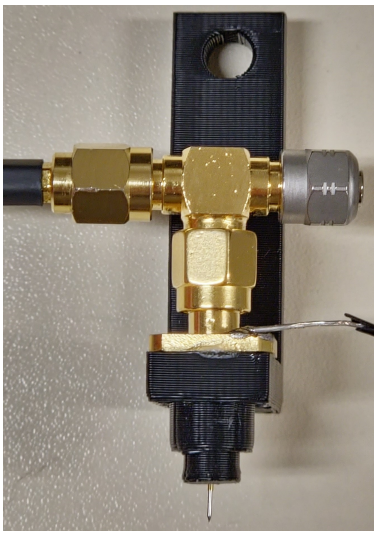
2.7 Actual enhanced BBI platform

The previous models being a useful tool to draw quick conclusions and predictions, it does not represent the reality. To that end, I set up the various presented enhancements in an actual BBI platform in order to verify the soundness of all the outcomes. In the first place, we are going to discuss how to perform the approximate impedance matching. Then, I will explain how to set up an efficient grounding bypass. After that, we will take a look at actual measurements allowing to spot the improvements.

2.7.1 Generator impedance matching in practice

The impedance matching solution proposed and used during my experiments is shown in Fig. 2.12a. However, it is far from being perfect. For instance, an ideal impedance matching implementation should be adaptive and dynamically change the impedance seen by the generator to perfectly match 50 Ω in every case. It would require a system with feedback, capable of measuring in real time the impedance presented by the IC target, in addition to the transmission line characteristics, to be able to dynamically adapt the compensation load impedance value. However, this is not the approach I have chosen. Indeed, the goal here is to minimize financial cost and platform modification, while

(a) *Actual implementation of the proposed impedance matching.*



(b) *Better alternative implementation of the impedance matching.*

Figure 2.12: *Actual pictures of a simple way to match the output impedance of the generator.*

allowing for better control over the platform parameters, whilst keeping a constant probe design. However, there is a trivial improvement which can be done, and is shown in Fig. 2.12b. Instead of connecting the compensation load at the generator output, one can connect it at the transmission line output, also known as the probe input. Although it was not used during my experiments, it is a solution which can be considered.

Another room for improvement would be to do a single and static measurement of the average IC backside impedance over its entire area, or over the targeted area (such as the cryptographic core for instance). Then, thanks to this value, the compensation load impedance could be chosen to better match the required 50 Ω.

Eventually, other shortcomings that neither of these solutions consider are the exact nature of the IC load. Indeed, the IC presents a complex impedance, which cannot be strictly approximated to a resistive load. Due to its internal structure, its impedance also has capacitive and inductive components. Therefore, considering them should lead to even better impedance matching. However,because the chosen solution requires little to no change to an existing platform and has proven to be good enough thanks to the previous models and the following results, I decided to stick to it.

2.7.2 Grounding installation bypass in practice

As we discussed previously, the grounding installation can drastically vary from one platform to another. Its effective impedance can be very high, such as in our platform, where equipment is grounded thanks to the platform earthing, with inter-equipment ground of around 150 Ω. To alleviate the effects of such ground impedance, I simply decided to shunt the existing earthing with short low-resistance copper wires. To that end, I chose an arbitrary piece of equipment as the reference, and connected every other piece of equipment local ground to the reference. It allowed reducing the effective platform ground impedance to a value close to 0 Ω.

2.7.3 Practical analysis

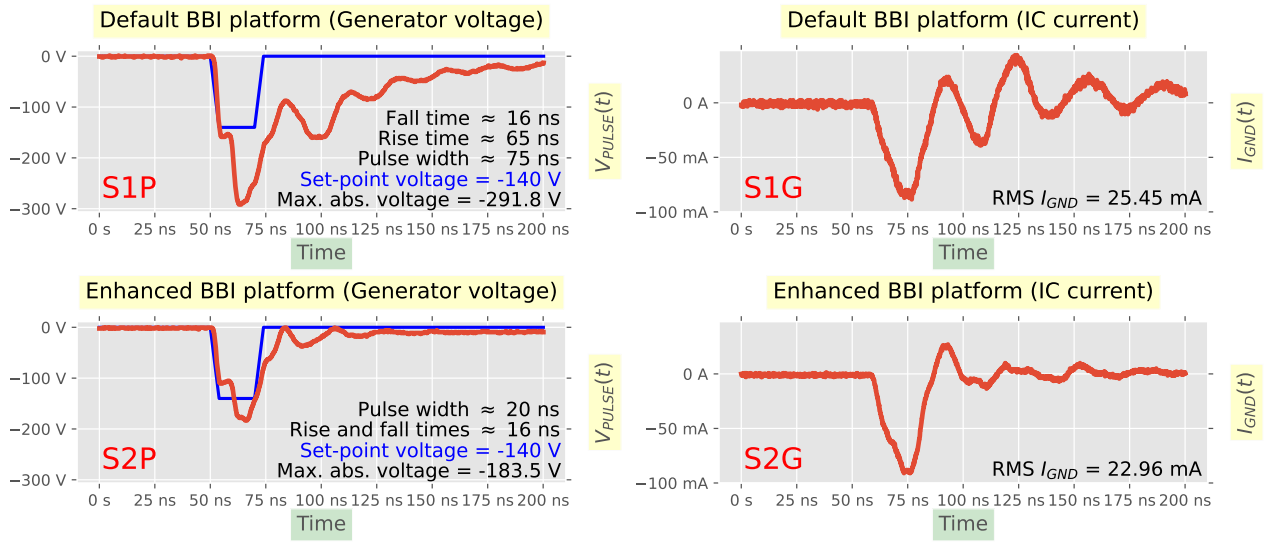


Figure 2.13: *Actual BBI platforms comparison: state-of-the-art (S1P and S1G) versus the proposed enhanced platform (S2P and S2G).*

Now that I presented how to practically set up the enhancements, let us analyze actual measurements on the platform. We will compare before and after results, allowing us to analyze each evaluation criterion. As it was done for the simulations, we will observe the voltage pulse and the IC ground current.

Fig. 2.13 presents the various waveform results. The voltage pulse was measured at the IC backside during the injection, and the IC ground current was measured using a current probe thanks to the IC PCB ground interconnection. Therefore, the measured current is precisely the IC ground current, excluding any other equipment. The four waveforms displayed in Fig. 2.13 are code named using three characters for clarity. The first character is common to all waveforms, denoted "S" for "setup". Then, the number indicates which platform is concerned, "1" being the default platform, "2" being the enhanced one. Eventually, the last letter indicates which waveform is observed, "P" being the voltage pulse, "G" being the IC ground current. Therefore, the default platform contains S1P and S1G waveforms, while the enhanced one contains S2P and S2G signals. Fig. 2.13 also displays the waveforms characteristics for more clarity. The ideal voltage pulse applied has a maximum negative amplitude of 140 V, a pulse width of 20 ns, and 4 ns rise and fall times.

S1P shows a clear undershoot of -108 % under the set point. It is a clearly non-negligible value, which is far from desirable when performing fault injection, as most of the time , the voltage value has a great importance concerning efficiency and repeatability. In addition to this, the pulse width is 275 % higher than its set point. It is an additional undesirable behavior, especially when one wants to inject precise disturbances into an IC under test. Then, fall time is four times higher than requested, and rise time is more than fifteen times higher. Put with the longer pulse width, it worsens the pulse accuracy.

S1G brings to light the obvious ringing issue, also observable to a lesser extent on S1P, which leads

to longer than expected disturbance inside the IC. Considering that the ringing is mainly caused by impedance mismatch between the generator and the IC, it will drastically change from one location to another, further reducing repeatability.

S2P, on the other hand, shows a better voltage amplitude, with a -31 % undershoot. It is far from perfect, but given the approximate nature of the proposed impedance matching, it was to be expected. Concerning the pulse width, the set point value is perfectly respected, which is very important for precise disturbance duration. However, rise and fall times are now consistent, but still four times higher than requested. It can easily be explained by the fact that the transmission line, the probe, the IC and the power installation are not a purely resistive load. Therefore, any capacitive element in the chain will inevitably reduce the system response time, thus elongating rise and fall times, leading to a shorter pulse plateau.

Concerning S2G, the approximate impedance matching shows a clear ringing reduction, with a steep current pulse, leading to a precise disturbance.

2.8 Enhanced BBI platform in a fault attack context

Now that we have seen with simple actual experiments the benefits of the proposed enhanced BBI platform, let us linger on further experiments to verify more thoroughly the soundness of these enhancements. To that end, I performed a differential fault attack on our IC target. More specifically, a constraining fault attack requiring single bit faults on one or more bytes working on an AES cryptographic core, introduced by C. Giraud [27] in 2002, submitted in April 2002 to CHES'02. In the first place, we are going to discuss in details the core of the attack. Afterward, I will describe the IC target, its characteristics, and its operating conditions for the experiments. Then, I will introduce experiments we developed to perform preliminary measurements to the attack, accelerating the search of points of interests on the IC. Next, we will discuss the practical attack results. Eventually, we will draw conclusions on the various observations.

2.8.1 Giraud's DFA detailed description

When Giraud's paper [27] was published in 2002, no existing DFA was capable of attacking an AES algorithm. In this context, they proposed two types of DFA on AES, in order to cover various fault types one can induce on secured ICs. In this thesis, I focused on the first fault model, consisting in inducing single bit faults, therefore, this is the one we are going to discuss and describe in details in this section. It is interesting to note that Piret and Quisquater published another DFA one year after Giraud to CHES'03 [37], but I focused on Giraud's DFA for my thesis.

As we said before, the attack requires single bit faults on AES computation. More specifically, the fault has to appear at the beginning of the final AES round. Because we are using an AES-128, we will describe everything with this in mind. In addition to this, the various notations we will be using

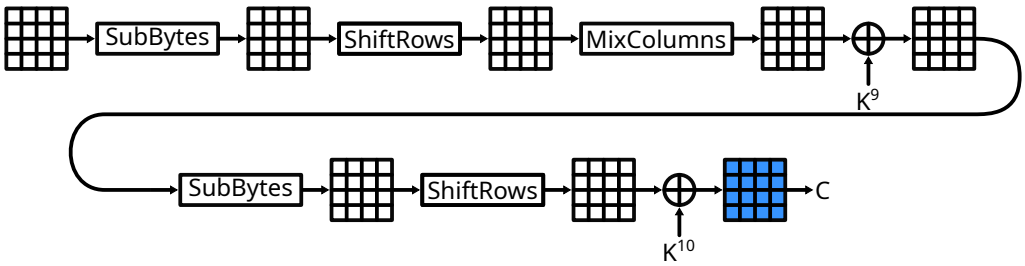


Figure 2.14: *The two last rounds of an AES-128*

are the following:

- P is the AES plaintext and K the AES secret key
- P^i stands for the intermediate cipher result after the i^{th} AES round
- P^i_j is the j^{th} byte of P^i
- K^i represents the i^{th} AES round key
- As for P, K^i_j is the j^{th} byte of K^i
- C is the correct ciphertext, C_j is the j^{th} byte of C
- Eventually, CF stands for the faulty ciphertext, CF_j is the j^{th} byte of CF

Although the attack requires single bit faults on the final round, the attack is fairly simple and quick to perform with the right data at hand. I will not describe how AES operates, as it is well described in [27, 5]. The final ciphertext is given thanks to the following equation:

$$C = ShiftRows(SubBytes(P^9)) \oplus K^{10} \tag{2.1}$$

With $SubBytes(P^i_j)$ being the substitution table (S-box) result calculated on M^i_j byte, and $ShiftRow(j)$ being the j^{th} byte position of the temporary result of the $ShiftRows$ transform. Thanks to eqn. 2.1, we can then deduce:

$$C_{ShiftRow(i)} = SubByte(P^9_i) \oplus K^{10}_{ShiftRows(i)}, \forall \in [\![0, 15]\!] \tag{2.2}$$

If an attacker manages to induce a fault e_j on a singe bit of the j^{th} byte of the intermediate cipher P^9 before the AES final round, we have the following faulty ciphertext CF:

$$CF_{ShiftRow(j)} = SubByte(P^9_j \oplus e_j) \oplus K^{10}_{ShiftRow(j)} \tag{2.3}$$

Which then gives us as before:

$$CF_{ShiftRow(i)} = SubByte(P^9_i) \oplus K^{10}_{ShiftRow(i)}, \forall \in [\![0, 15]\!] \tag{2.4}$$

If there is no fault on the i^{th} byte of P^9, thanks to eqns. 2.2 and 2.4, we have the following relation:

$$C_{ShiftRow(i)} \oplus CF_{ShiftRow(i)} = 0 \tag{2.5}$$

If there is a fault on P_j^9, we have, thanks to eqns. 2.2 and 2.3:

$$C_{ShiftRow(j)} \oplus CF_{ShiftRow(j)} = SubByte(P_j^9) \oplus SubByte(P_j^9 \oplus e_j) \tag{2.6}$$

Eventually, we have to first calculate ShiftRow(j), which gives us the location of the only non-zero byte of $C \oplus CF$, which in return gives us j. We then need to find P_j^9: we look for the single bit fault e_j, and identify an ensemble of values of P_j^9 satisfying eqn. 2.6. For each correct value, we increase a counter by 1. Then, by taking another faulty ciphertext CF, and the correct value for P_j^9 should be counter more often than another incorrect value. Therefore, we can identify the correct value thanks to that affirmation. This process shall be repeated as much as needed to find every bytes of P^9.

Thanks to eqn. 2.1, it is possible to retrieve the last round key K^{10}, which can be then converted to the AES secret key thanks to the inverse Key Scheduling applied on K^{10}. To finish.

2.8.2 Integrated circuits target characteristics

For the purpose of understanding clearly how we set up the previous attack, it is required to describe thoroughly the integrated circuit targeted. The model is an STM32F439VIT6 32-bits ARM Cortex-M4 microcontroller from STMicroelectronics, available in a LQFP100 package. The IC is manufactured using a 90 nm bulk technology. Its main characteristics are the following:

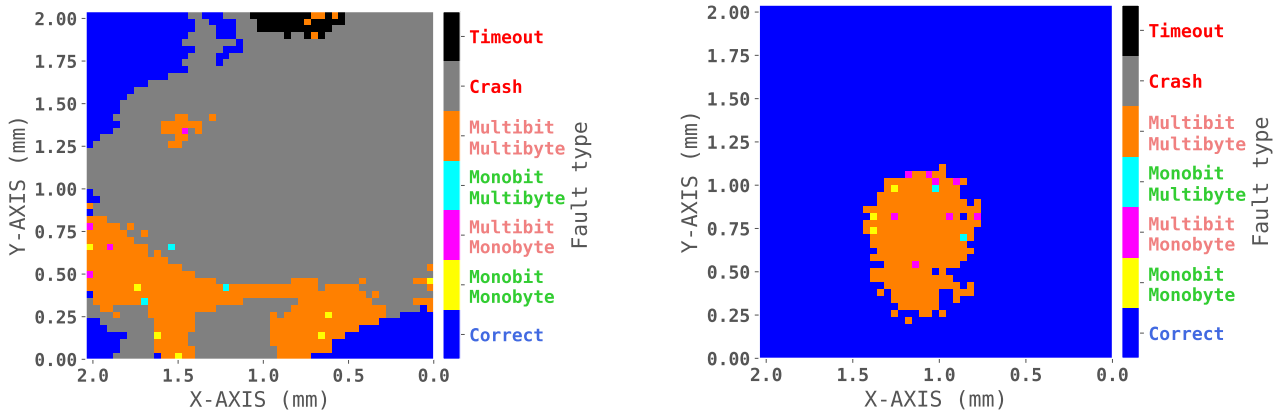
- A core clock up to 180 MHz;
- Two 1 MB FLASH memory banks;
- 256 kB of RAM;
- Voltage supply allowed from 1.7 V to 3.6 V;
- A True Random Number Generator (TRNG);
- A dedicated hardware cryptographic coprocessor, embedding AES (128, 192 and 256 bits), triple DES, and various HASH algorithms;
- A 700 µm substrate thickness.

For the purpose of every other experiment, the IC is clocked at 40 MHz thanks to an external 8 MHz crystal oscillator.

2.8.3 Preliminary attack experiments

2.8.3.1 Fault analysis mapping description

For the purpose of accelerating and simplifying the attack process, especially because creating single bit faults is a troublesome process, I designed experiments to be conducted on the IC target, allowing



(a) *FAM in a state-of-the-art BBI platform. 70 % of the tested locations exhibits an IC crash, and 15 % exhibits multibytes-mutibits faults. No Giraud's criterion compatible fault is observed.*

(b) *FAM in an enhanced BBI platform. 89.9 % of the tested area shows a correct behavior, 9.9 % are incompatible with Giraud's criterion. Five locations show single bit faults, potentially useful for the Giraud attack.*

Figure 2.15: *Fault analysis mapping*

me to spot interesting IC areas to perform the attack on. Because the attack targets the AES coprocessor, all the experiments described here are performed specifically on the AES core area.

These experiments are called "Fault Analysis Mapping" (FAM), and two results are shown in Fig.2.15. An FAM consists in performing BBI on the cryptographic core of the IC and trying to inject faults while identifying its behavior. We separated seven fault cases, described in Table 2.1.

Fault type	Description
Correct	The AES outputs a correct result
Monobit Monobyte	The fault is located on a single bit on a single byte
Multibit Monobyte	The faults are located multiple bits on a single byte
Monobit Multibyte	The faults are located multiple bytes and are single bit
Multibit Multibyte	The faults are located multiple bytes and multiple bits
Crash	The microcontroller did not respond correctly
Timeout	The microcontroller was unresponsive

Table 2.1: *FAM faults description.*

Over the seven outcomes, only two of them can lead to potential exploitable fault according to Giraud's criterion: Monobit Monobyte and Monobit Multibyte. We performed these experiments on a state-of-the-art (default) platform and on our enhanced platform, with the exact same equipment on both platforms.

2.8.3.2 Fault analysis mapping comparison

The FAMs I performed have the following parameters:

- A voltage pulse amplitude ranging from -150 V to -400 V with -5 V steps;

- A fixed pulse width of 4.5 ns;
- A fixed pulse delay of 150 ns + 553 ns allowing to target the penultimate AES round;
- The mapping measures 2 mm by 2 mm, with an isotropic displacement step of 40 µm.

These experiments take from 16 hours at best, up to 36 hours at worst to perform. It is quite long, however, compared to blindly looking for the correct location to perform the Giraud's attack, it is statistically much faster, especially when the AES approximate location is known to the user. I performed two FAMs, on the same IC target, for a state-of-the-art platform and for the proposed enhanced platform. FAM results are shown for the default platform in Fig. 2.15a and for the enhanced platform in Fig. 2.15b.

On the one hand, concerning the state-of-the-art platform, where the FAM is shown in Fig. 2.15a, we can spot numerous locations where a microcontroller crash was observed, more specifically 70 % of the tested locations. It is problematic as this behavior cannot lead to any meaningful data to perform a fault attack. Despite trying numerous experiment parameters, I was never able to obtain a single bit fault on any physical location on the AES core, even considering the ringing and reducing the voltage down to -20 V, to a point where the generator is not specified anymore to deliver consistent amplitudes.

On the other hand, without any tweaking, the FAM results show five single-bit faults. In addition to this, the IC did not crash at any given moment, and eight multi-bit faults can be spotted. It gives valuable information related to potentially interesting areas where the Giraud's DFA could be performed. It does not mean that the attack can be performed entirely on a single location, but it is a great way to guide the attack process, knowing that the set of parameters used is sound.

2.8.4 Attack results and analysis

Thanks to the previous FAM results, I decided to perform the attack on every candidate location using the enhanced platform as a target. For each selected location above the AES core, a parameter sweep was performed, consisting in finding, for each set of parameters, as much single bit faults as possible. The test settings are the following:

- The voltage pulse set point ranging from -300 V to -600 V;
- The pulse width ranging from 4.5 ns to 5.5 ns;
- The injection delay ranging from $\pm$ 10 ns around the penultimate AES round.

For each set of parameter, I set a limit of 100 single bit faults. However, this is an optimistic goal which, in some cases, can take a very long time to achieve, and in other cases, cannot be achieved at all. Therefore, I decided to limit the number of trials to 10000, allowing the test algorithm to be finite and not too long to perform. Then, with these results, I was able to perform the attack, where the result are shown in Table 2.2.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
K10	0xFF	0x1F	0x42	0xE8	0xEF	0x44	0xA5	0x6A	0xCA	0xE7	0x55	0x3C	0xFD	0x65	0x39	0x26
KEY	0x01	0x23	0x45	0x67	0x89	0xAB	0xCD	0xEF	0xDE	0xAD	0xBE	0xEF	0x12	0x34	0x43	0x21

Table 2.2: *Giraud's DFA results. In yellow are indicated the bytes retrieved with a brute-force method instead of the Giraud's bit fault attack.*

What is obvious at first glance is that I was not able to retrieve the 16 bytes of K^{10}. Indeed, without further search for valid locations above the AES core, I was able to retrieve 14 out of 16 bytes. It is a great result, as the attack took about 20 hours to perform, including the FAM, which was the longest experiment to set up the attack. However, to retrieve the secret key, it is required to find all 16 bytes of K^{10}. Because there are 16 bits left, it is not particularly relevant to try to find interesting locations to perform Giraud's DFA, as there are only 65536 combinations to blindly test at worst to find the correct last round key. Considering the platform computer is able to perform approximately $188 \cdot 10^3$ encryptions per second, it would take $\frac{2^{16}}{188 \cdot 10^3} \approxeq 349 \cdot 10^{-3}$ seconds to perform the required calculation in the worst scenario. I then blindly tested every possibility to find the last two bytes, and I was able to retrieve them, thus allowing me to retrieve the AES secret key. These brute-forced bytes are shown in yellow in Table 2.2.

2.9 Conclusion

In this chapter, I first introduced existing BBI platforms both in the state-of-the-art and commercially available. We have seen that multiples solutions ranging from low prices to very high prices exist, each one having its own advantages and disadvantages concerning their characteristics. Thanks to this platform overview, I was able to enumerate the fundamental building block of a typical BBI platform. After that, I presented the platform used during my thesis experiments, from the custom probe to the generator. Afterward, I introduced electrical models I designed to quickly compare and evaluate BBI platforms. We studied the simulation results of such models, which allowed me to introduce enhancements to existing BBI platforms, allowing for better accuracy and reproducibility. Thereafter, I presented experiments performed to verify the soundness of such models, comparing state-of-the-art platforms to the enhanced platform I propose. Eventually, to go further in the model validation, I described and performed a constraining differential fault attack on a hardware AES coprocessor, sustaining the usefulness of the proposed enhancements.

III

Integrated circuits modeling

Contents

3.1 Summary

This chapter is dedicated to introducing the work I carried out concerning the modeling and simulation of integrated circuits subject to BBI. It begins with a discussion of the various aspects involving how ICs are designed and manufactured. It includes a thorough description of their power delivery network and silicon substrate. The main aspects of their structure, being inherited from the standard design flow provided by CAD vendors, is also described. Afterward, it introduces electrical models allowing to simulate integrated circuits under BBI. Then, it lingers on how to properly model the voltage pulse generator and the electrical probe, which are the main tools for performing BBI. Eventually, it shows the study of how actual logic gates react to BBI disturbances and the implications of such results. Parts of this work have been published both in [3] and [35].

3.2 Introduction

When evaluating and studying ICs under BBI, it is important to be able to fully predict and understand the underlying mechanisms at work in order to set up reproducible and reliable experiments, as well as being able to set up efficient countermeasures. However, modeling and simulating integrated circuit behavior subject to fault injection is not an easy task. More specifically, simulating an entire IC at a transistor level under fault injection is unrealistic with current resources and technology. It is especially true when considering time cost, as most modern digital ICs are composed of billions of transistors.

On the other hand, when considering microcontrollers, the transistor count is sensibly lower, in the order of the million of transistor. As no software nor algorithm is currently dedicated to simulate the functional, electrical behavior of millions of transistors at the same time while some of them are disrupted by strong and transient disturbances, I had to overcome this limitation. In addition to that, to be able to set up a reliable model, it is required to have knowledge relative to the detailed architecture of the considered IC, which is impossible in most cases, as the vast majority of digital IC architectures are proprietary and closed-source. Therefore, it is required to find alternative workarounds in order to be able to study IC behavior and their various responses to fault injection techniques.

One of the first technique allowing to simulate entire ICs behavior under fault injection has been first proposed in 2019 and 2020 concerning Electromagnetic Fault Injection (EMFI) [38] and Laser Fault Injection (LFI) [39]. It was further extended for EMFI in [2]. More specifically in the latest work, the proposed solution consists in establishing an equivalent non-logical model of an elementary section of the considered IC. Instead of modeling each logic gate with as many transistors as required to form a logic function, it was chosen to represent a hundred of logic gates in an average way, solely with a few resistors and capacitors, in addition to the power delivery network and the silicon substrate. It results in a transistor-less model, achieved using manufacturing data for the studied IC. The authors assumed that the first half of the transistors are conducting while the other half

are blocking. Then, by repeating the model and interconnecting the various instances to one another, the authors managed to evaluate the IC power delivery network behavior under EMFI. This clever solution allows to drastically reduce the computing work required to analyze and predict behaviors of ICs subject to EMFI. Indeed, simulating the average behavior of a hundred of logic gates only with four resistors and four capacitors is immensely lighter and faster than simulating the equivalent with BSIM (Berkeley Short-channel IGFET Model) transistor models. However, the main shortcoming being the lack of functionality of the IC models, it is impossible to evaluate the consequences of EMFI concerning their functional behavior.

Body biasing injection being less documented than EMFI, no distributed model has yet been proposed to simulate ICs under BBI. In this context, my motivations were to set up and evaluate electrical models being able to reliably predict both in time and space IC behavior in order to understand how BBI induced disturbances propagate and create faults inside ICs. To that end, I chose to use the previous paper [2] model as a strong basis, while completing the model and providing improvements to properly consider the unique aspects of BBI. Then, and because such model cannot allow me to evaluate the consequences of BBI disturbances on the logic gates behavior, I completed the simulation flow to consider this aspect.

This chapter begins with a presentation of typical integrated circuits structure, including its power delivery network and the logic gates layout, arranged in elementary blocks called Standard-Cell Segments (SCS). Afterward, I introduce the electrical models allowing me to simulate ICs under BBI, while verifying their soundness. Then, I present the importance of properly modeling the voltage generator when working with BBI simulations. Eventually, I introduce the new workflow I designed, expected to be used in conjunction with the previous models, allowing me to evaluate the functional consequences of BBI on integrated circuits.

3.3 Integrated circuits structure

For the purpose of properly introducing the electrical models I developed for BBI, it is required, in the first place, to linger on how integrated circuits are structured. It involves analyzing the main structures composing an IC, such as:

- Its power supply network, consisting in various metal levels stacked one on top of the others;
- The standard-cells: pre-characterized logic cells used as elementary building blocks;
- The various substrate types, such as Dual-Well and Triple-Well that I considered in my work, not to cite them all.

3.3.1 Power supply rails

In complex digital integrated circuits, power distribution is typically realized thanks to a grid structure on multiple levels, as illustrated in Fig. 3.1 for two metal levels. The upper layer forms a ring

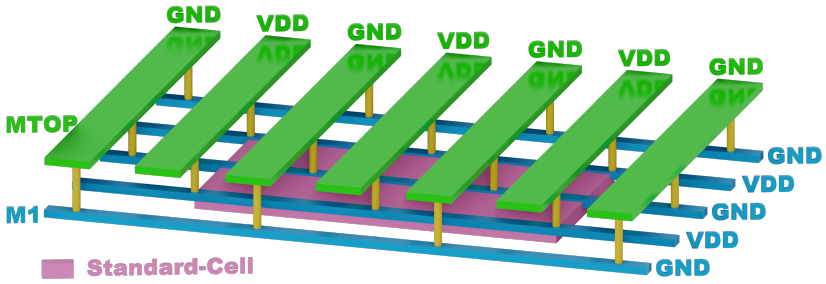


Figure 3.1: *Typical integrated circuit power delivery network, with two metal levels, showing standard-cell rows and standard-cell segments (in yellow) sandwiched between GND and VDD power rails.*

around the IC core. Each layer of the grid is a set of constant width metal lines equally spaced. The lines direction is orthogonal between layers. Then, vias are used to electrically connect the layers together at each overlap location. Commonly, the lower the layer the thinner the lines are. Thus, the lower grid brings the power close to each standard-cell segment, one of them being highlighted in desaturated yellow in Fig. 3.1. This topology has the advantage to bring a robust power delivery network inside the IC silicon. Indeed, there are multiple paths between power connections of each standard-cell segment, which leads to less power change sensitivity of the standard-cell segments. However, the shortcoming of such architecture lies in the high amount of metal resources required to create the power grids.

3.3.2 Standard-Cell Segments

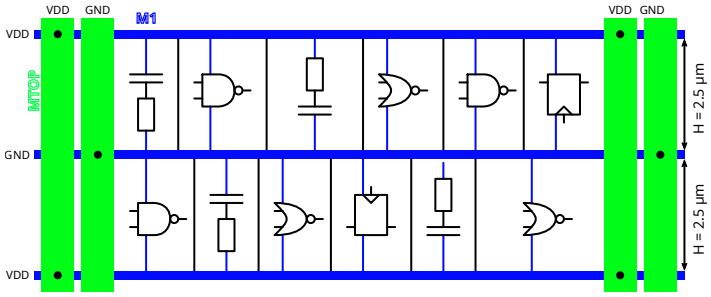
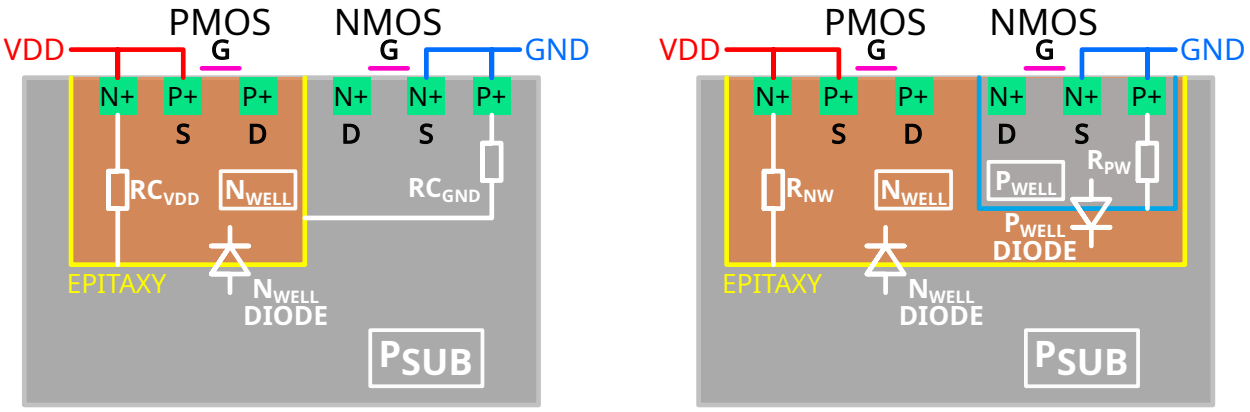


Figure 3.2: *Symbolic view of a Standard Cell Segment, surrounded by its local power delivery network.*

As I described above, the power planning of ICs creates standard-cell segments (SCS), delimited by power and ground rails. They typically contain standard logic gates and power decoupling capacitors. The standard logic gates are pre-defined and pre-characterized concerning their performance (both for timing and power), for each silicon technology node, by IC manufacturers. They are standard in the way that their height is fixed, while their width vary according to their complexity. It allows placing and routing them easily in rows, delimited by the power lines around them. Fig. 3.2 shows how a SCS is surrounded by VDD and GND power lines.

3.3.3 Various substrate types

During my work, I focused on bulk silicon substrate technologies. In most cases, the base silicon is P-doped, and called the substrate, on which are stacked the transistors and the power delivery net-



(a) *CMOS inverter in a dual-well silicon substrate sectional view. The epitaxy is the junction between the P-substrate and the N-well. RC_{GND} is the access resistance from the epitaxy to the NMOS through the P-substrate. RC_{VDD} is the access resistance from the epitaxy to the PMOS through the N-well.*

(b) *CMOS inverter in a triple-well silicon substrate sectional view. The epitaxy is the junction between the P-substrate and the N-well. R_{NW} is the access resistance from the epitaxy to the PMOS through the N-well. Inside the N-well is created the P-well. R_{PW} is the access resistance from the N-well/P-well junction to the NMOS.*

Figure 3.3: *Dual-well (3.3a) and triple-well (3.3b) inverter silicon sectional view.*

work. In these structures, there are two typical ways of lithographing the transistors inside the silicon substrate. Cross-sectional views of a logic inverter manufactured with both of these techniques are shown in Fig. 3.3, and are the following:

- A Dual-Well substrate, where the NMOS transistors are lithographed into the P-doped silicon substrate, while a N-doped area called the N-well is created inside the substrate to lithograph the PMOS transistors;
- A Triple-Well substrate, where an additional buried P-doped area is manufactured inside the N-well, called the P-well, allowing to lithograph the NMOS transistors inside it instead of the substrate.

3.3.3.1 Dual-Well substrates

To begin with, let us focus on Dual-Well substrates. A cross-sectional view of a CMOS inverter manufactured in a Dual-Well substrate is shown in Fig. 3.3a. Among moderately old ICs, it was common to find Dual-Well substrates. As I have stated before, in Dual-Well substrates, NMOS transistors are lithographed directly into the P-doped silicon substrate, as it is shown in Fig. 3.3a. In addition to this, a N-doped silicon area is created inside the P-substrate, called the N-well, to lithograph the PMOS transistors. This results in a silicon junction, electrically represented by the N-well diode on the schematic, highlighted in saturated yellow. Because doped silicon does have a non-zero resistivity, electrical resistances are represented to demonstrate this:

- RC_{VDD} represents the access resistance measured between the substrate and the PMOS transistor through the N-well;
- RC_{GND} is the access resistance measured between the epitaxy and the NMOS transistor through the P-substrate.

3.3.3.2 Triple-Well substrates

Let us now focus on Triple-Well substrates, which are nowadays commonly used in modern ICs because such substrates offer interesting noise properties. Indeed, thanks to an additional silicon well, it provides an electrical isolation between areas of the IC, thus reducing substrate crosstalk noise.

As for Dual-Well substrates, the cross-sectional view of an inverter manufactured using a Triple-Well substrate is shown in Fig. 3.3b. Similar to Dual-Well substrates, the PMOS transistor is lithographed in a dedicated N-well, located in the P-substrate, which creates a first diode: the N-well diode. Within the N-well, a P-well is created to lithograph the N-well transistor. Due to the appearance of a second silicon junction, it creates a second diode, the P-well diode. These two diodes, connected backwards relative to each other, isolate the PMOS and NMOS transistors from the rest of the substrate. The access resistances of these areas are:

- R_{NW} is the access resistance between the substrate and the PMOS transistor, measured through the N-well;
- R_{PW} is the access resistance between the N-well and the NMOS transistor, measured through the P-well.

It allows avoiding the propagation of switching noise from one part of the IC to another.

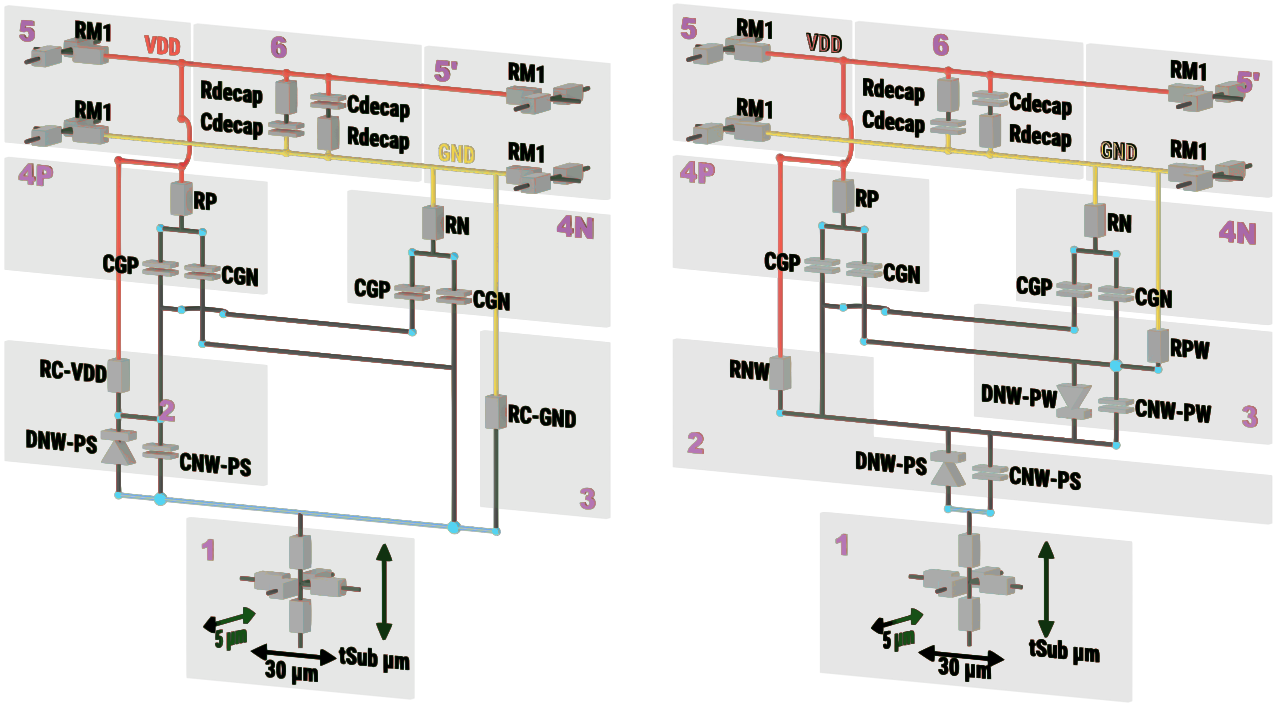
Generally, Dual-Well and Triple-Well substrates are used in conjunction on an IC die, therefore being able to manufacture transistors both on Dual-Well and Triple-Well substrates while taking advantages of the isolating properties of Triple-Well.

3.4 Standard-Cell Segment (SCS) and their models

Thanks to what I have introduced in the previous section, that is, the Standard-Cell arrangement used to create IC architectures, alongside the two identified substrate types of interest: Dual-Well and Triple-Well, it is now possible to elaborate an electrical model for such integrated circuits. Because I am differentiating Dual-Well and Triple-Well substrates, I am introducing two separate models, even though they show some similarities. The models I developed are an improvement over the electrical models proposed by M. Dumont for EMFI [2].

3.4.1 Former models

The original Standard-Cell Segment models for Dual-Well and Triple-Well substrates proposed by M. Dumont [2] are shown in Fig. 3.4. Let us now analyze their structure.



(a) *Dual-Well SCS model.* **(b)** *Triple-Well SCS model.*

Figure 3.4: *Original 3D Dual-Well and Triple-Well IC comprehensive Standard-Cell Segment electrical models.*

3.4.1.1 Dual-Well SCS

Fig. 3.4a shows the electrical model of an SCS for Dual-Well substrates. Each SCS is delimited by the P-substrate at its bottom (1) and by the power rails at its top (5, 5′). The dotted wires at the bottom of the substrate indicates that each substrate layer can be repeated as required to form the correct t_{Sub} thickness. Dual-Well SCS are composed of six main regions, each one describing a part of an IC core sampling:

- Region 1 is the SCS substrate resistive network, composed of six resistors;
- Region 2 is the P-substrate/N-well junction, modeled by a diode (DNW-PS) and its capacitance (CNW-PS), and is called the epitaxy, represented thanks to the blue net on the schematic. In addition to this, there is an access resistance from the epitaxy to the top of the N-well (VDD), representing the non-zero electrical resistance of N-doped silicon;
- Region 3 is the access resistance from the epitaxy up to the GND power rail, representing the non-zero electrical resistance of the P-doped substrate;
- Region 4P is the model of the PMOS transistors, half of them being conducting. It consists in an access resistance RP from the VDD power-rail to the PMOS, and two capacitances, CGP being the load formed by the PMOS input capacitances, CGN the load formed by the NMOS input capacitances;
- Region 4N is similar to 4P: it is the model of the NMOS transistors, half of them being conducting. It consists in an access resistance RN from the GND power-rail to the NMOS, and two capacitances, CGP being the load formed by the PMOS input capacitances, CGN the load formed by the NMOS input capacitances;

- Region [5] and [5′] are the power network interconnections, represented on two metal levels, the colored ones being the first metal levels, the gray being the top metal level.
- Eventually, region [6] is simply the decoupling existing between both GND and VDD power rails.

3.4.1.2 Triple-Well SCS

Fig. 3.4b illustrates the SCS model for Triple-Well substrates. The SCS are delimited, as before, by the P-substrate at their bottom and the power delivery network at their top. As for the Dual-Well model, the bottom dotted wires indicate that each substrate layer is repeated as needed to create the correct thickness required. The main difference between Dual-Well and Triple-Well SCS models lies in the region [3]. First, let me describe every region:

- As for the previous model, region [1] is the silicon P-doped substrate;
- Region [2] is the N-well, created inside the P-substrate, with the junction (epitaxy) represented by a blue wire. It is composed of the diode DNW-PS, and its capacitance CNW-PS, alongside the N-well access resistance RNW from the epitaxy up to the VDD network;
- Region [3] is where the Triple-Well model drastically differ from the Dual-Well one. As I have explained before, in Triple-Well substrates, the NMOS transistors are lithographed inside an isolated P-doped region called the P-well. Therefore, region [3] describes the P-well, with an additional silicon junction modeled with the diode DNW-PW connected backward compared to DNW-PS, its capacitance CNW-PW, and the P-well access resistance RPW to GND;
- The three other regions are identical to the previous Dual-Well model.

3.4.2 Enhancing the substrate model

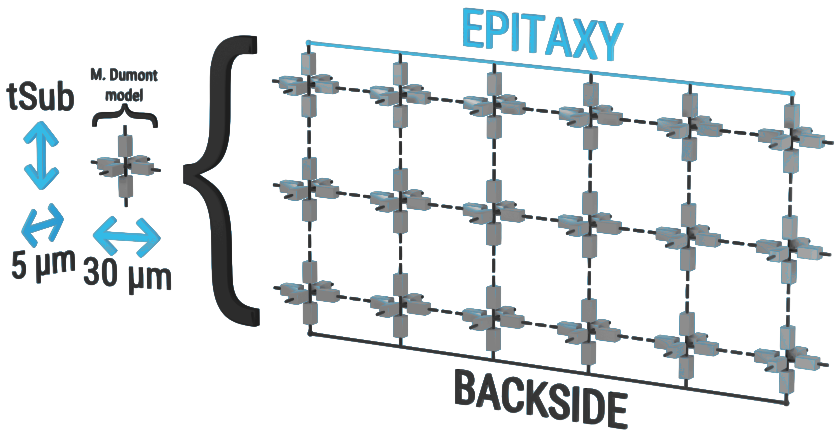


Figure 3.5: *Standard-Cell Segment substrate sub model subdivision. It represents an improvement in geometric resolution over M. Dumont model for EMFI [2]. The backside is the accessible substrate, and the epitaxy is the highest substrate level.*

In the model designed by M. Dumont, the substrate block at the SCS bottom is only modeled

thanks to six resistors. If it was enough to fully appreciate and simulate EMFI disturbances, it is not sufficient to model and simulate BBI disturbances, for many reasons.

Among them, we can identify the coarse nature of the substrate model. Indeed, the six resistors composing the model are sufficient in an EMFI context as the substrate acts as an almost transparent environment and is here only to create the substrate electrical interconnect between SCS. However, in a BBI context, the substrate is the physical environment used to convey the energy from the metal probe to the transistors. In addition to this, the typical substrate thickness of ICs ranges from 40 µm to 1 mm, and using only two resistors to represent 700 µm of material is a too wide geometric step to fully appreciate the mechanisms at work during BBI. Furthermore, as the modeled SCS is 5 µm wide and 30 µm long, the spatial resolution of the substrate sub model is not the same in each direction.

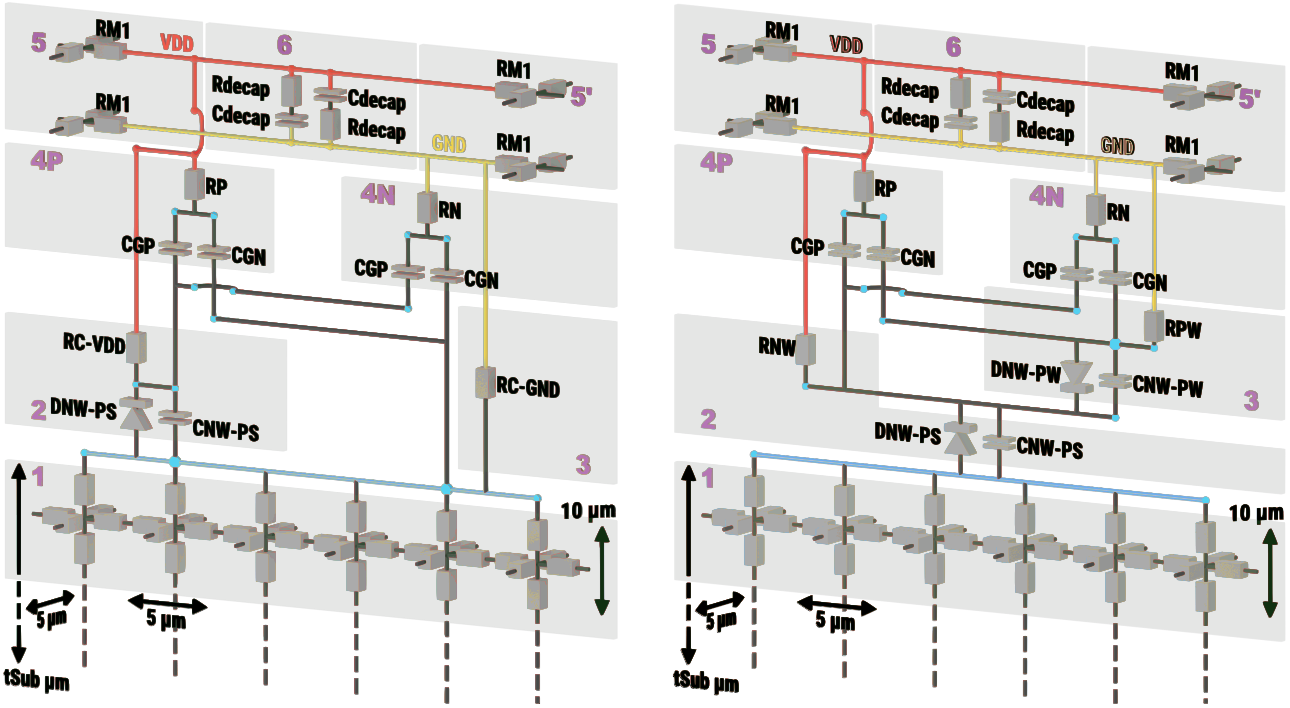
Therefore, I decided to improve the former model to allow for finer geometry resolution. My proposition consists in fixing the width and length of an elementary substrate sub model to a value of 5 µm, while allowing the user to have a certain degree of freedom concerning the elementary thickness value. However, I chose a 10 µm default value for the elementary thickness as it is a good value, computationally speaking. To achieve this, I split the former six-resistors sub model into many six-resistors blocks connected to each other, as it is shown in Fig. 3.5. These new substrate blocks then measure 5 µm by 5 µm by 10 µm. As one can see, to preserve the higher part of the SCS model, I kept the SCS length of 30 µm, which results in a 6 by 1 array of elementary substrate blocks. Eventually, these enhancements also allow me to analyze the effects of substrate thinning on the models, similar to what is done with actual ICs.

3.4.3 The considered SCS for the rest of my work

Considering that the IC target I used for my experiments is the STM32F439VIT6, and because we have technical information concerning the inner structure of the target IC technology, I considered this information to calculate the various SCS model parameters. Therefore, every simulation presented in this work relies on a model that is as close as possible from the actual microcontroller. Fig. 3.6 presents the enhanced model used during my simulations, and Table 3.1 shows the components values calculated in accordance to the 90 nm technology.

Component	RM1	Rmtop	Cdecap	Rdecap	CGP	CGN	p-sub
Description	Metal 1 resistance	Metal top resistance	Power decoupling capacitance	Power decoupling resistance	Equivalent PMOS gate capacitance	Equivalent NMOS gate capacitance	Substrate resistivity
Dual-well	*26* Ω	*5* Ω	*2.25 fF*	*2* Ω	*35.2 fF*	*25.2 fF*	*0.01* Ω.*m*
Triple-well	*26* Ω	*5* Ω	*2.25 fF*	*2* Ω	*35.2 fF*	*25.2 fF*	*0.01* Ω.*m*
Component	**RP**	**RN**	**CNW**	**RC-GND**	**RC-VDD**	**RNW**	**RPW**
Description	PMOS equivalent resistance	NMOS equivalent resistance	Diode capacitance	GND to substrate access resistance	access resistance	N-well access resistance	access resistance
Dual-well	*9.57* Ω	*5.3* Ω	*20 fF*	*3.1* kΩ	*3.1* kΩ	n/a	n/a
Triple-well	*9.57* Ω	*5.3* Ω	*20 fF*	n/a	n/a	*3.1* kΩ	*3.1* kΩ

Table 3.1: *SCS model numeric values.*



(a) *Dual-Well SCS model.* **(b)** *Triple-Well SCS model.*

Figure 3.6: *3D Dual-Well and Triple-Well IC comprehensive Standard-Cell Segment electrical models.*

3.4.4 Interconnecting Standard-Cell Segments together

The models I previously presented only describe a small portion of an integrated circuit, this is to say an SCS. Therefore, to be able to model an entire IC, it is required to instantiate several SCS and to connect them with each other in a mesh arrangement. Fig. 3.7 shows coarsely how it is

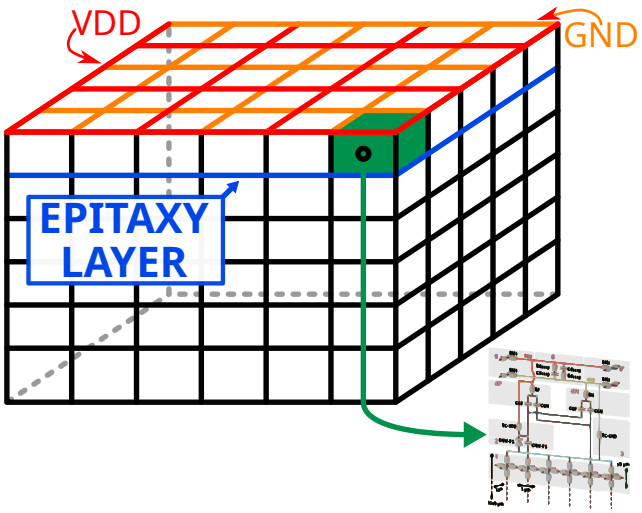


Figure 3.7: *Three-dimensional Standard-Cell Segments interconnection example.*

achieved. I have chosen to work with the SPICE simulator to perform the simulations as it is a widely used software in electronics and is available in various flavors. Therefore, SCS are written in SPICE language. As I have said before, the very elementary building substrate blocks are written manually and verified before any usage. Then, the SCS elementary models are automatically generated instead of being written manually. It allows avoiding human errors and enabling fast modifications to the

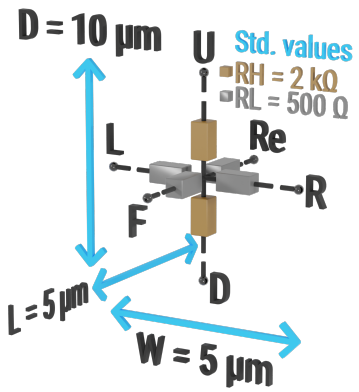
models when required.

Then comes the interconnection step. To that end, it is needed to create a top SPICE file which instantiates every SCS required to form the IC and defines constant parameters in addition to simulation conditions. I chose to use the Python language to do so as I use it for every tool I write my work, therefore enabling fast support and update, while ensuring cross-script compatibility. By doing so, the interconnection process is automated and free from human errors. The actual generator Python script is annexed at the end of the manuscript (Listing **??**). Because I have designed two separate models, one for Dual-Well and another for Triple-Well substrate types, in addition to the various dynamic parameters which are useful when modeling an IC, the algorithm offers a certain degree of flexibility, allowing me to cover multiple use cases. The actual generation script uses procedural generation according to the input parameters. Various parameters are user-accessible and can be modified to fit the needs. Next is the list of these settings:

- The resulting IC size;
- The BBI probe location;
- The IC global substrate thickness;
- The substrate sub-model thickness;
- The substrate type: Dual-Well, Triple-Well, or a mix of both, allowing to replicate actual IC architectures;
- The voltage pulse amplitude, width, and rise and fall times;
- Various SPICE simulation settings.

Eventually, the program incorporates a visual inspection tool in order to provide a quick verification of the generated IC structure to the end-user.

3.4.5 Writing the elementary models



(a) *Elementary substrate block 3D schematic with the values used for our technology and the SPICE in-out names.*

```
.subckt elementary_bloc D F L R Re U
R1 U N001 RH
R2 N001 D RH
R3 Re N001 RL
R4 N001 F RL
R5 N001 L RL
R6 R N001 RL
.ends elementary_bloc
```

(b) *Elementary 6-resistors substrate SPICE netlist description, with the in-out names in accordance with Fig. 3.8a.*

Figure 3.8: *Elementary substrate building block 3D schematic and its SPICE netlist.*

If the SCS models are automatically generated, I have written manually the substrate sub models. Indeed, they are very simple and straightforward, and it allowed me to test them independently before incorporating them into the SCS models. In addition to writing them, I calculated, thanks to the technology values of our IC targets, the effective substrate resistor values. Considering the substrate resistivity being $\rho = 0.01\ \Omega \cdot m$, and the elementary block size being $W = \ \mu m$; $L = 5\ \mu m$; $D = 10\ \mu m$, it is trivial to calculate each resistance value R_i thanks to the following equation:

$$R_i = \frac{\rho \cdot l}{S} \tag{3.1}$$

Knowing that $L = W \equiv LW$ and that $D = LW$, we can first write that the vertical resistances will be four times the horizontal ones. Therefore, calculating one value of them gives the six values. Let us calculate the vertical resistances, which I will call RH:

$$RH = \frac{\rho \cdot \frac{D}{2}}{L \cdot W} = 2000\ \Omega \tag{3.2}$$

Therefore, the horizontal resistances are equal to $RL = 500\ \Omega$. These calculations can be adapted as needed depending on the technology used. In addition to that, a substrate with a given thickness t_{Sub} can be represented with virtually any number of layers. For example, one can reduce the number of layer by ten by adjusting the resistor values. It has the advantage to provide a lighter simulation, which will be faster to perform, at the cost of less accuracy. Eventually, this model gives us an electrically isotropic environment, as it should be.

Previously, I have stated that an SCS is 30 µm wide. However, the elementary block I described is 5 µm wide. To achieve a 30 µm wide SCS, it is simply required to connect six of these blocks together to form a $W = 30\ \mu m \cdot L = 5\ \mu m \cdot D = 10\ \mu m$ substrate, as it can be seen in Fig. 3.6 models. The

```
.subckt elementary_blocx6 D1 D2 D3 D4 D5 D6
+F1 F2 F3 F4 F5 F6 L R RE1 RE2 RE3 RE4 RE5 RE6
+U1 U2 U3 U4 U5 U6 VSUBCintC
XX1 D1 F1 L VSUBCintL2 RE1 U1 elementary_bloc
XX2 D2 F2 VSUBCintL2 VSUBCintL1 RE2 U2 elementary_bloc
XX3 D3 F3 VSUBCintL1 VSUBCintC RE3 U3 elementary_bloc
XX4 D4 F4 VSUBCintC VSUBCintR1 RE4 U4 elementary_bloc
XX5 D5 F5 VSUBCintR1 VSUBCintR2 RE5 U5 elementary_bloc
XX6 D6 F6 VSUBCintR2 R RE6 U6 elementary_bloc
.ends elementary_blocx6
```

Figure 3.9: *SCS substrate layer SPICE netlist*

resulting netlist is shown in Fig. 3.9, and naturally instantiate six times the previous netlist shown in Fig. 3.8b.

3.4.6 Preliminary models validation: IC operating point

Designing and creating the models I previously described is the first step in practically implementing them. Afterward comes various validation to verify the soundness of such models. Among

these validations, studying the resulting IC operating point is the first step to verify inconsistencies concerning idle power draw, allowing me to spot undesirable short-circuits and bad interconnects, coming from the SCS generation algorithm or from the global generation algorithm. To that end, using the generation algorithm, I created various ICs (a Dual-Well, a Triple-Well, and a mixed substrate) with the following measurements: a width of $550\ \mu m$, a depth of $450\ \mu m$, and a substrate thickness of $140\ \mu m$. It is, according to our platform computational power, an IC with a reasonable size/calculation time ratio. I generated three ICs:

- An exclusive Dual-Well circuit, to isolate the Dual-Well specific BBI effects and potential Dual-Well generation errors;
- An exclusive Triple-Well circuit, to isolate the Triple-Well specific BBI effects and potential Triple-Well generation errors;
- Eventually, a mixed substrate IC, comprising at the same time Dual-Well and Triple-Well SCS, to mimic a real IC, more specifically our platform microcontroller.

For each generated IC, I simulated the operating point and identified the key values, which are presented in Table 3.2. We can observe very low steady current, which, according to the model, is

Table 3.2: *Dual-well, triple-well and mixed substrates SCS operating point.*

Measurement	Description	Dual-well	Triple-well	Mixed substrates
I_{GND}	IC global ground current	1.92 nA	1.94 nA	3.4 nA
I_{VDD}	IC global VDD current	-1.96 nA	-5.8 nA	-3.5 nA
GND_{AVG}	Max. GND voltage	1 nV	1 nV	1.75 nV
VDD_{AVG}	Min. VDD Voltage	1.2 V	1.2 V	1.2 V

consistent and should be expected. Indeed, as the transistors are modeled thanks to resistors and capacitor in parallel, and because the diodes are not conducting, there is no path for DC current without further circuit bias. Therefore, the power supply drop is negligible, with a uniform power delivery of 1.2 V everywhere.

3.5 Modeling the voltage pulse generator and the probe

In this section, I describe the various steps I went through to properly model the voltage pulse generator and the problem it arises. In the first place, I present a very simple and naive way of modeling the generator, using an ideal voltage source. Then, after having analyzed the shortcomings of such model, I introduce a better model which better suits an actual generator. Eventually, I analyze preliminary validation results including the new generator model.

3.5.1 Various generator architectures

There are several existing types of pulse generator. Some of them are designed as voltage sources, while others are current sources. In addition to this, the way their output stage is designed is also

the origin of important behavioral differences. It is then important to take a quick look at these architectural differences. We can distinguish pulse generator into two categories depending on their output stage coupling: DC-coupled and AC-coupled. In my work, I use a voltage source DC-coupled generator, but it is not the case of all other works. Therefore, I am going to cite various documented generators which have been used for the study of BBI.

For instance, in [40] and [41], the authors used an ESD gun to create the pulses on their IC substrate. These devices typically have a DC-coupled output, commonly feature an RCR output filter, and are designed to generate single pulses, very short and of very high voltage, without having a fine control over rise and fall times or pulse width. Their output impedance is not finely controller when compared to high precision high voltage pulse generators. Therefore, it is important to consider these parameters when elaborating an electrical model.

On the other hand, there are other works, such as [1], in which the generator output stage consists in a transformer. Therefore, it is AC-coupled to the IC, which changes drastically the electrical behavior of such generator compared to a DC-coupled one. For instance, it is not possible to inject energy into the IC outside of pulse edges, while on DC-coupled generators it is.

Eventually, like in my work, there are very high voltage high precision generators, which are DC-coupled and allow for precise pulse width and voltage set point. Their output stage may vary unit to unit, but their output impedance is precisely controller. In my case, the generator is specified to deliver its specifications into a 50 Ω load.

With this in mind, let us jump into the next section.

3.5.2 Voltage pulse generator naive model

First, let us consider a very simple voltage generator, an ideal variable voltage source, and a very simple probe, a perfect wire. In that scenario, as the generator is DC-coupled, it is biasing the substrate at 0 V at rest. However, under normal conditions, an IC substrate should not be biased externally. Indeed, whether in my model or in an actual IC, it can create unexpected behavior. In addition to this, our AVTECH generator does not bias the IC at rest, even when the probe is connected. Instead, it is operating in a high-impedance mode where its output presents a very high impedance to the IC backside, thus preventing any undesirable biasing.

3.5.3 More elaborated generator model

In this context, I decided to mimic our actual generator structure in my model to properly appreciate its behavior in the simulations.

To do so, I chose to use a time-controlled resistor, which is a component available in SPICE. It is very straightforward, and consists in dynamically defining the resistor value. To that end, before

the pulse, its value is high-impedance (100 GΩ), during the pulse it is close to zero, and after the pulse it goes back to a high-impedance value. By doing so, I am avoiding any bias which could cause unexpected behavior of the model, which then could invalidate the simulation conclusions.

3.6 Modeling BBI disturbances: further model validation

With correct SCS and generator models, it is now possible to simulate BBI disturbances. To that end, we distinguish Dual-Well and Triple-Well substrates, to better analyze their differences, both in behavior and in structure. As for the previous section, the ICs have the following measurements: a width of $550\ \mu m$, a depth of $450\ \mu m$, and a substrate thickness of $140\ \mu m$, representing 1620 SCS connected to each other in a mesh array. There are two external power supply rails located at the top and at the bottom of each IC. The probe is a 30 µm square probe, placed at the IC center, and the generator settings are the following:

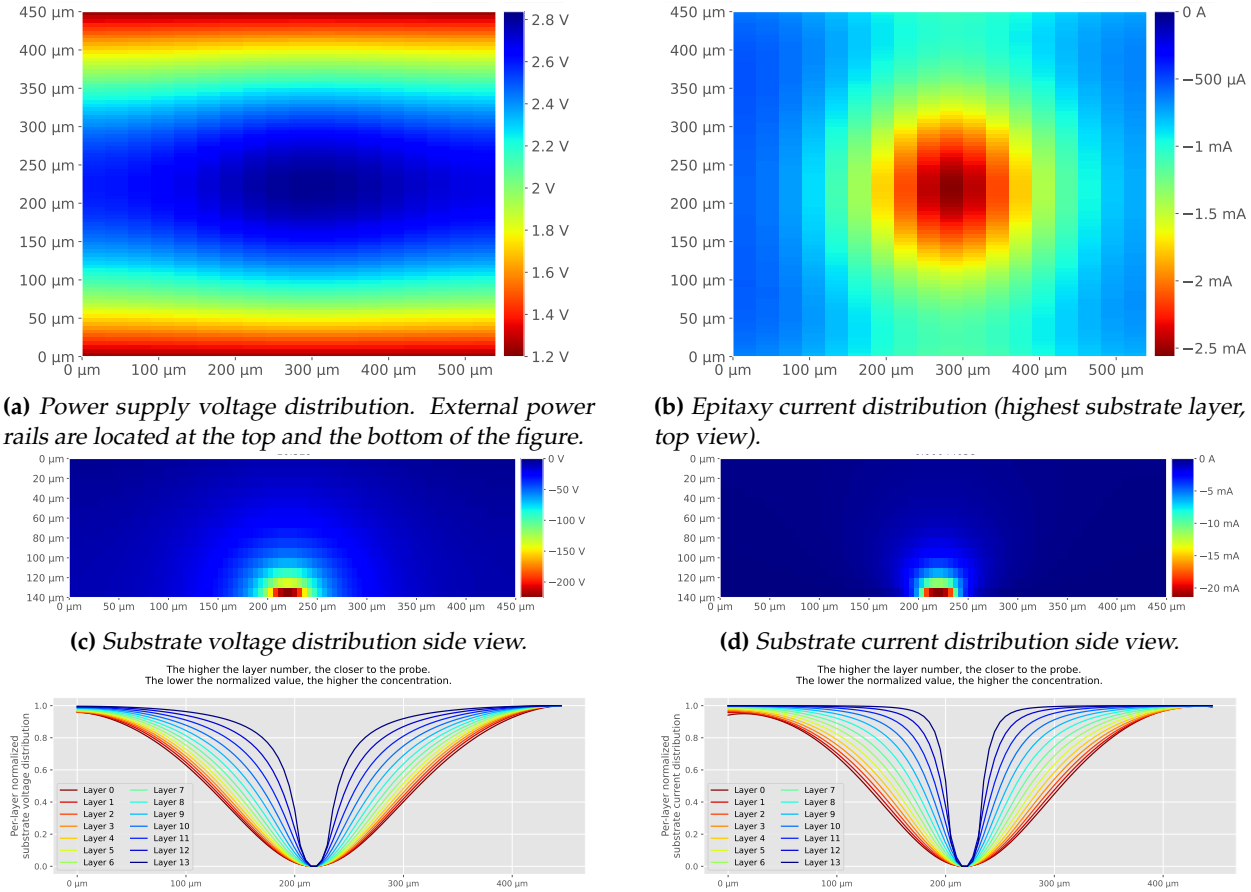
- Voltage pulse maximum amplitude: -300 V;
- Voltage pulse width: 20 ns;
- Rise and fall times: 8 ns;
- Approximate impedance matching realized thanks to a 50 Ω load, as described in the Chapter 2.

For each considered IC, we observe various signals in a two-dimensional view to fully appreciate the spatial behavior of the ICs under BBI, at the apex of the disturbance:

- The power supply voltage distribution (top view);
- The epitaxy current distribution (top view);
- The substrate voltage distribution (side view from Y-axis);
- The substrate current distribution (side view from Y-axis);
- The per-layer normalized substrate voltage focusing (side view from Y-axis);
- The per-layer normalized substrate current focusing (side view from Y-axis);

3.6.1 Dual-Well integrated circuits under BBI

Fig. 3.10 presents the six simulation results for a Dual-Well substrate integrated circuit. Sub-fig. 3.10a shows us the power supply differential voltage distribution across the entire IC (top-view). As mentioned before, the power rails are located at the top and bottom of the IC. Quite naturally, we can observe less deviation from the nominal 1.2 V value at these spots. However, at the IC center, just below the probe, the power supply differential voltage rises up to 2.8 V. This represents a 133 % increase over the nominal value. Fortunately, it lasts only a few dozen of nanoseconds.



(a) *Power supply voltage distribution. External power rails are located at the top and the bottom of the figure.*

(b) *Epitaxy current distribution (highest substrate layer, top view).*

(c) *Substrate voltage distribution side view.*

(d) *Substrate current distribution side view.*

(e) *Per-layer normalized substrate voltage concentration.*

(f) *Per-layer normalized substrate current concentration.*

Figure 3.10: *Dual-Well SCS simulation results at the apex of the pulse disturbance.*

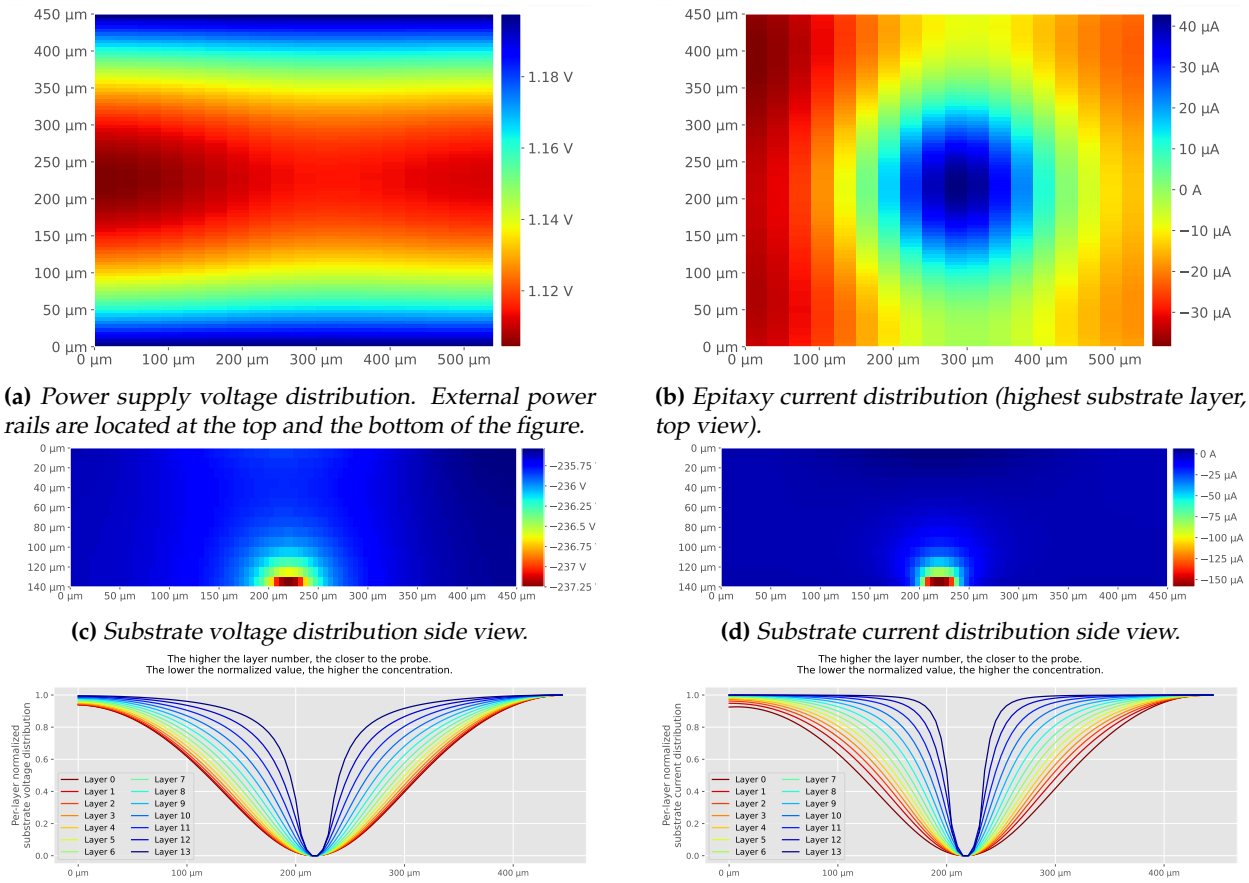
Sub-fig. 3.10b shows us the current distribution at the epitaxy level (the highest substrate layer), which represents the charges going from the substrate to the top of the SCS. What is worth noting here is that most of the charges are flowing through the center of the IC (a.k.a below the probe). It is to be expected with an isotropic environment such as the substrate. It is also natural to observe when comparing with sub-fig. 3.10a, where the power delivery deviation were stronger below the probe.

Sub-fig. 3.10c and 3.10d respectively illustrate the voltage distribution and the current distribution respectively across the substrate from a cross-sectional view at the center of the IC along the Y-axis. These results are interesting as they show the homogeneous propagation of the energy through the substrate, with equipotentials forming half-circles. In addition to this, these results show a clear example of a DC-coupled probe with the IC. Indeed, the substrate acts as a voltage divider, and the further we are from the probe, the lower the voltage is. However, these results are not practical enough to fully appreciate the concentration of charges inside the substrate. To that end, and to better show these results, let us analyze sub-fig. 3.10e and 3.10f.

As for sub-fig. 3.10c and 3.10d, they represent respectively the voltage concentration and the current concentration. For each substrate layer (14 layers in that case as we simulated a 140 µm substrate), the voltage was normalized along the 3D Y-axis, thus allowing to grasp and compare each layer to the other. The higher the layer number, the closer to the probe, and the lower the normalized

value, the higher the charge concentration. What is very interesting to remark here is that on both sub-figures and for each layer, most charges are focused in the center of the layer. Then, the deeper we go inside the substrate, the wider the focusing. This result is to be expected. Indeed, because the charges diffuse homogeneously inside the substrate, the further they are from the probe, the more they have diffused. However, even in the deepest layer (layer 0), most of them are still highly focused below the probe. Therefore, it allows us to see that most of the energy is transferred directly under the probe.

3.6.2 Triple-Well integrated circuits under BBI



(a) *Power supply voltage distribution. External power rails are located at the top and the bottom of the figure.*

(b) *Epitaxy current distribution (highest substrate layer, top view).*

(c) *Substrate voltage distribution side view.*

(d) *Substrate current distribution side view.*

(e) *Per-layer normalized substrate voltage concentration.*

(f) *Per-layer normalized substrate current concentration.*

Figure 3.11: *Triple-Well SCS simulation results at the apex of the pulse disturbance.*

Fig. 3.11 presents the simulation results for a Triple-Well substrate integrated circuit. Sub-fig. 3.11a shows the power supply differential voltage distribution (top-view). As for the Dual-Well IC, the power rails are located at the top and bottom. What should be observed here is that there are less spatial variations compared to the Dual-Well substrate, and the power supply voltage is lower at the center of the IC instead of being higher. In addition to this, the deviation is a lot less important on that case, with a voltage only down to 1.1 V from the nominal 1.2 V.

Sub-fig. 3.11b shows us some interesting results. At the peak of the disturbance, there is almost no current at the epitaxy level. At first glance, it can seem odd, but we will explain why this is logical

thanks to the substrate distribution results which follow.

Therefore, sub-fig. 3.11c and 3.11d show the voltage and current distribution inside the substrate from a cross-sectional view. The results might seem similar to the ones of the Dual-Well substrate, but they are not. Indeed, when looking at the voltage and current scales, we can remark two things:

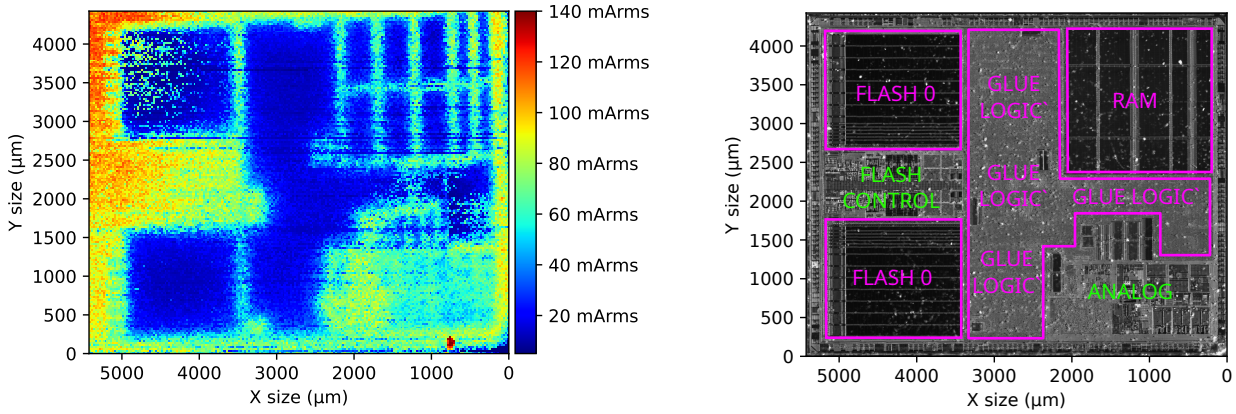
- The voltage distribution is extremely homogeneous, with an average value of -236 V;
- The current distribution is very low when compared to the Dual-Well substrate, with still a slight amount of current at the probe location.

These results are explained by the fact that with negative pulses in a Triple-Well substrate, both silicon junction diodes are not conducting. Therefore, the Triple-Well simulated IC acts as a big capacitor. The system probe/substrate is then AC-coupled to the top of the SCS. Then, similar to one side of a capacitor, the substrate charges up to the set-point value, if it has enough time to do so, or to a lower value (-236 V here) if it does not have enough time. Eventually, because only AC signals can pass through a capacitor, the current tends to be zero on the pulse plateau.

Sub-fig. 3.11e and 3.11f show, as before, the normalized current concentration for each substrate layer. Because the substrate network is identical in both simulated ICs, we observe the same result as before, with a strong concentration of charge below the probe, and a charge diffusion when going deeper into the substrate.

3.6.3 Major differences between Dual-Well and Triple-Well circuits

The main behavioral differences between Dual-Well and Triple-Well substrate types lie in the substrate structure, more precisely in the way transistors are lithographed. Indeed, as I have stated previously, in Dual-Well substrates, NMOS are lithographed directly into the P-substrate, which allows a DC coupling between the backside of the IC and the NMOS. On the other hand, in Triple-Well substrates, as NMOS and PMOS are electrically isolated from the substrate thanks to the P-well and the N-well. Therefore, no DC coupling is possible between neither the PMOS nor the NMOS. This is because silicon diodes are blocked, and thus act as capacitors. Therefore, the substrate is AC-coupled with the logic gates in a Triple-Well substrate, while it is DC-coupled with the NMOS and AC-coupled with the PMOS in a Dual-Well substrate. These structural differences result in behavioral differences when performing BBI, as we have observed thanks to the simulations. Therefore, it changes the way one can create faults in an IC, or an IC region, depending on the substrate type. In addition to this, because energy can only flow during voltage changes in an AC-coupled scenario, the effective current over a given injection with fixed parameters will always be lower on a Triple-Well substrate compared to a Dual-Well substrate, where the DC-coupling allows for energy transfer during the pulse plateau.



(a) *IC ground current mapping of the entire STM32F439 IC target, allowing to spot Dual-Well and Triple-Well areas depending on the amount of measured current.*

(b) *IR photograph of the STM32F439 IC target.*

Figure 3.12: *IC ground Current mapping and IR photograph of our IC target.*

3.6.4 Dual-Well and Triple-Well circuits in practice

Thanks to the previous observations and conclusions, we can expect to observe such substrate structural differences on an actual IC. Therefore, we can expect Dual-Well regions to exhibit a higher amount of measured RMS current over Triple-Well for a fixed set of experimental parameters. To that end, we chose to use our STM32F439 IC target and perform straightforward experiments.

We called these experiments IC ground current mapping, and consists in choosing an arbitrary set of fixed pulse parameters, and performing a spatial sweep of the entire IC backside while measuring the current going out of the target ground connector at each location. For clarity purposes, only the RMS current is displayed in the mapping results. It is important to remark that in these experiments, the IC is powered-off, but connected to its main power supply. Fig. 3.12a shows experimental results for a 50 µm thick IC, and Fig. 3.12b shows an IR photograph of the exact same target for comparison purposes. The experimental parameters are the following:

- Voltage pulse amplitude: -70 V;
- Voltage pulse width: 20 ns;

When analyzing the results, we can notice strong differences in the measured RMS current between various areas. Therefore, we could suppose that the areas showing a bigger RMS current are composed of Dual-Well substrates, while the areas showing a lower current are composed of Triple-Well substrates. It is indeed true. At the bottom-right are located the analog blocks, such as the voltage regulators and the PLLs. On the one hand, concerning our target technology, analog areas are typically manufactured using Dual-Well substrate. On the other hand, glue logic, such as the CPU core and the cryptographic coprocessor are manufactured using Triple-Well substrates, to take advantage of the isolation it provides. As expected, the measured current above the glue logic is lower compared to the analog or the FLASH control area.

3.7 Conclusion

In this chapter, I have introduced a modeling and simulation flow for Body Biasing Injection. The models are based on a typical integrated circuit structure, including the power delivery network, the logic gates and the silicon substrate. They allowed me to simulate various IC structures under BBI. The most important things they taught me are how energy is flowing inside the substrate to reach the logic gates. Thanks to an analysis of charges concentration in the substrate, I was able to observe that most of the energy flows below the BBI probe. Therefore, it shows that BBI seems to be a local fault injection method, such as LFI, even if their precision is on different levels. Then, thanks to a deep study of two substrate types: Dual-Well and Triple-Well, I was able to draw conclusions on how BBI pulses impact the energy propagation inside ICs. I observed that for negative pulses, in Dual-Well substrates, ICs are DC-coupled with the probe, thus allowing for energy to flow during the entire duration of the pulse. On the other hand, in Triple-Well substrates and negative pulses, ICs are entirely AC-coupled to the probe, thus allowing energy to only flow on the edges of the pulse. These differences mean that for a given disturbance, a lower amount of energy can be transferred into a Triple-Well IC compared to a Dual-Well IC. Therefore, it is important to keep this in mind concerning the potential hazards BBI practice on Dual-Well ICs could involve.

IV

From more complete simulation models to fault model

Contents

4.1 Summary

This chapter presents an extended simulation flow, which adds up to the one presented in the previous chapter. This extension is designed to overcome the limitations of the previous SCS models. Indeed, as I have mentioned, they do not provide the logical behavior evaluation of the simulated ICs. However, they allow simulating large ICs in a reasonable amount of time. While this limitation is a design choice, which is to approximate the transistors with average equivalent models, it is not an insurmountable boundary.

When studying fault injection methods, it is fundamental to understand the mechanisms at work in fault creation. It allows creating a fault model or associating an existing one to the fault injection technique. With this knowledge, it is then possible to set up countermeasures. In short, the extended simulation flow consists in extracting signals from the SCS models and injection them in functioning logic gates.

4.2 Introduction

As I have described in the first chapter for various fault injection techniques, various fault models exist and can be associated to these methods depending on their impact on IC targets. Among them, one can identify single-bit fault models, bit-flip fault models, or bit set/reset fault models. Depending on the model, the faults created are data-independent or data-dependent. While data-independent faults are easier to create, data-dependent one leak information about the data being processed by the target IC. To identify the correct one or develop a new one for BBI, it is then mandatory to simulate the logic behavior of transistors subjected to BBI disturbances in an IC context.

To that end, I decided to extend the SCS simulation flow for it to consider the logical function of the simulated ICs. The one proposed in the previous chapter allowed me to finely analyze the effects of BBI disturbances in IC substrates and power delivery networks. In addition to this, I was able to study the electrical differences between Dual-Well and Triple-Well substrate types ICs under BBI. However, this approach alone does not give any insights on how fault occur inside logic gates. Indeed, the elementary SCS model represents a hundred of logic gates solely with two resistors and four capacitors, which is their average electrical behavior.

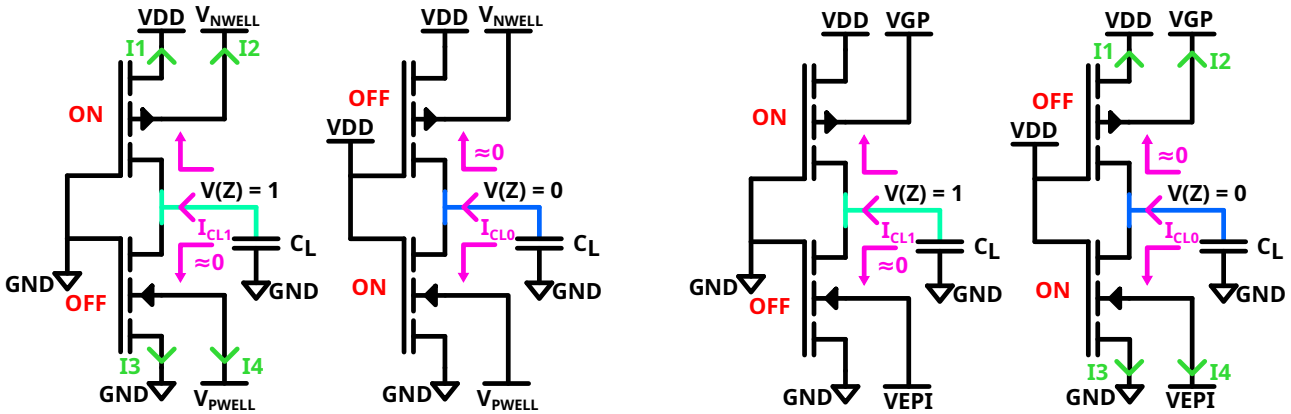
The simulation flow follow-up consists in adding a few extra steps to the original flow, allowing to actually measure the impact of BBI on logic gates. In the first place, I introduce these new simulation steps. Then, I present the simulation results of such an addition. Afterward, and thanks to the simulation results, I introduce a fault model which could explain the origin of faults in ICs subjected to BBI. Eventually, I take a step back on how this improves BBI modeling and simulation, while bringing opportunities of improvement.

4.3 Simulation flow follow-up

Due to the shortcomings of the simulation flow presented in chapter 3, which mainly lie in the fact that a hundred of transistors is approximated with two resistors and four capacitors, the simulation flow follow-up aims at considering the logical function of some of these transistors. To that end, one has to first choose a target SCS where they want to conduct the further analysis. Then, the SCS internal signals are extracted from the SCS simulation, and then they can be injected into simulated logic gates. The main signals of use are the following:

- VDD: the positive power supply net;
- VSS: the negative power supply net;
- VEPI: the voltage at the epitaxy level, also known as the junction between the substrate and the top of the SCS;
- VNW (VGP): the N-well voltage, required in both Dual-Well and Triple-Well substrates;
- VPW: the P-well voltage, only required in a Triple-Well substrate.

The signals VEPI, VGP, VPW and VNW, depending on the substrate type, act as the bulk voltage of the simulated transistors. In line with VDD and VSS, the shape of these signals dynamically change the bias of the transistors, and therefore their temporal behavior. To have a comprehensive understanding of the various possible scenarios, in addition to keeping clarity, I present the results for logic inverters, composed of two transistors in the technology of my thesis target microcontroller: 90 nm bulk. Triple-Well and Dual-Well substrates are presented in this order, and for each substrate type, two inverters are simulated: a normally low and a normally high. The simulated schematic and their significant signals are shown in Fig. 4.1.



(a) *Simulated inverters schematic: Triple-Well* **(b)** *Simulated inverters schematic: Dual-Well*

Figure 4.1: *Simulated inverters under BBI schematics.*

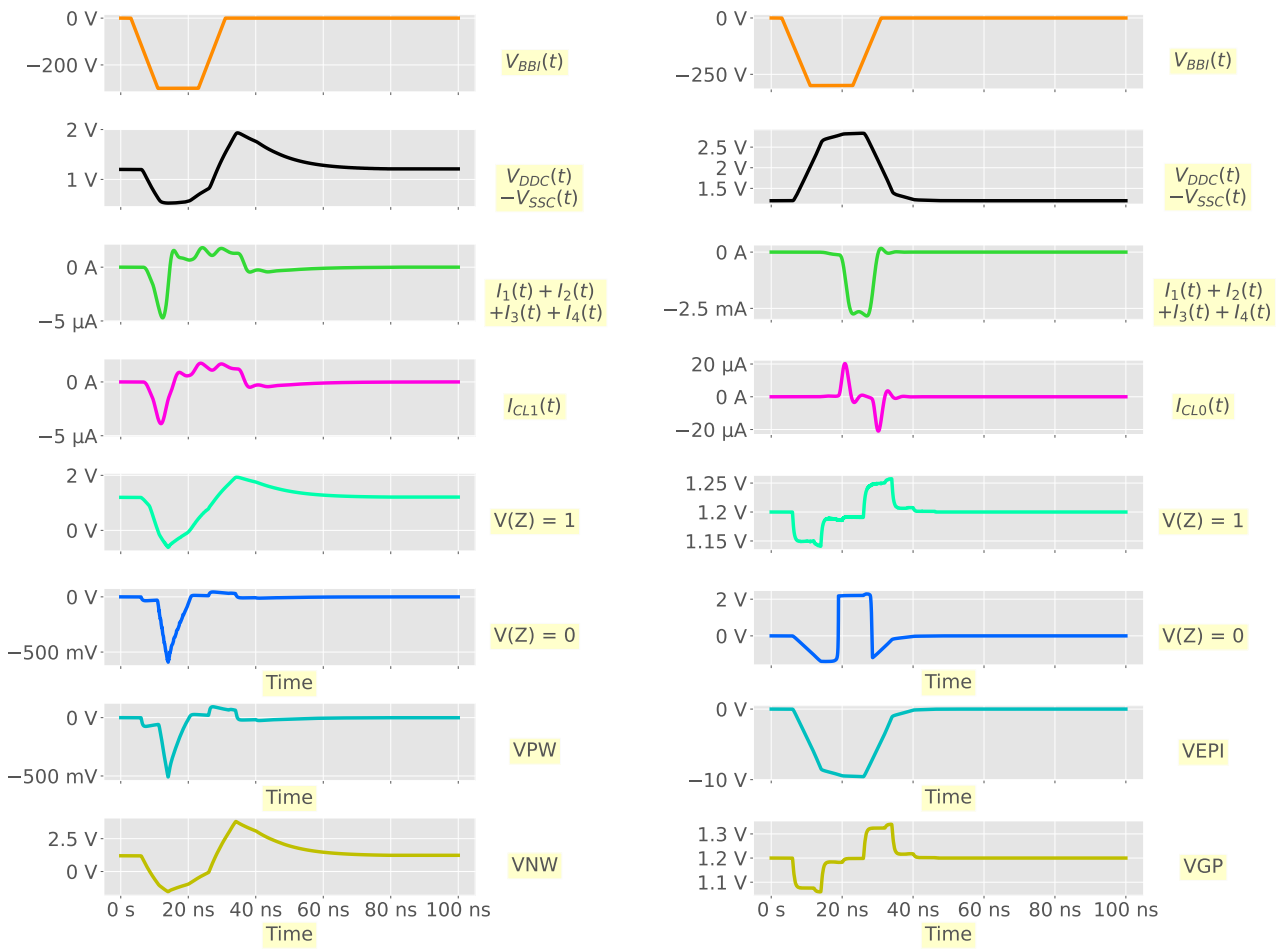
For each scenario, I present five signals of interest:

- The voltage pulse at the IC backside, for reference purposes (in orange);
- The local power supply voltage seen by the transistors (in black);

- The current sum of the inverter of interest (in medium green);
- The load current of the inverter of interest (in magenta);
- The output of the normally high inverter (in light cyan);
- The output of the normally low inverter (in dark blue);
- The P-well (otherwise epitaxy) voltage (in dark cyan);
- The N-well (otherwise VGP) voltage (in dark yellow).

Concerning the Triple-Well substrate, the inverter of interest is the normally high one, while for the Dual-Well substrate, it is the normally low one.

4.4 Simulation follow-up results and analysis



(a) *Triple-Well simulation results: analysis of the normally high inverter.*

(b) *Dual-Well simulation results: analysis of the normally low inverter.*

Figure 4.2: *Inverters under BBI simulation results.*

The disturbances injected into the inverters are extracted from the previous chapter SCS simulations. The chosen SCS is the one located directly under the BBI probe, in other words, at the center of the simulated IC. Therefore, according to the simulation flow follow-up and the accuracy of the SCS models, the hundred of logic gates modeled in that SCS are subject to the same disturbances.

4.4.1 Triple-Well substrate results and analysis

In the first place, let me introduce the Triple-Well results. The corresponding schematic is the one displayed in Fig. 4.1a, and its simulation results are shown in Fig. 4.2a. Let us remember that the coupling between the probe and the modeled IC is purely capacitive in that scenario. Therefore, energy is only transferred at the edges of the voltage pulse. It can be observed in the normally high inverter current sum waveform (medium green). Concerning the inverters, on the one hand, the normally high inverter sees a change in its output logical value during 12 ns, and follows, with a slight amplitude reduction, the N-well voltage waveform (dark yellow). This is because the normally high inverter has its capacitive load C_{L_1} drained out, which explains the voltage drop at its output. On the other hand, the normally low inverter sees a 500 mV voltage drop, which, in that case, does not change its output value. Therefore, the type of fault in that scenario is closer to a bit set/reset, which is data dependent. Thus, in some cases, it could result in no fault with these specific experimental conditions.

4.4.2 Dual-Well substrate results and analysis

Let us now focus on the Dual-Well results. The corresponding schematic is the one displayed in Fig. 4.1b, and its simulation results are shown in Fig. 4.2b. In that scenario, as I have stated before, the inverter of interest is the normally low one. Here, the coupling is both resistive and capacitive depending on the power supply side. It is resistive with the VSS network, and capacitive with the VDD network, due to the silicon junction between the P-substrate and the N-well. Overall, it results in a resistive coupling between the probe and the IC. In that case, the normally high inverter sees very little disturbance, therefore its output is unchanged. However, when analyzing the total current waveform (medium green) of the normally low inverter, we can see that the charges are injected into the inverter, which results in charging the output capacitor C_{L_0}, therefore setting to one the inverter output during about 10 ns. Once again, the faults are related to bit set/reset, which are data dependent.

4.5 A final word on the fault model for BBI

Thanks to the previous observations, drawn with the extended simulation flow results, I can now discuss a fault model for body biasing injection, similar to what has been made in the past concerning EMFI or LFI. According to my observations concerning the simulation results, I have got evidence that the faults induces thanks to BBI are data dependent. This is due to the fact that bit set/reset fault models are data dependent. Indeed, lowering the voltage of a logic gate outputting a low logical value, or increasing the voltage of a logic gate outputting a high logical value, has theoretically no effect on the logic being targeted. As I have shown in the last section, we can observe bit set/reset faults in logic gates subject to BBI. In addition to this, and thanks to the results of the previous chapter concerning the voltage concentration, BBI effects seems to be very local, depending on the spreading

of the charges inside the substrate. As Fig. 3.10f and Fig. 3.11f have shown concerning both Dual-Well and Triple-Well substrates, charges spread in majority in the direction the probe targets.

4.6 Conclusion

Eventually, my work can be put in perspective to what has been observed concerning laser fault injection in [42]. This work has shown that LFI can induce data dependent faults and single bit faults. Data dependent faults can help attackers by providing additional information on the nature of the data manipulated by the IC target. LFI is also a very precise fault injection method which does not spread out very much of the injection point. To put it all in perspective with my thesis, in the second chapter, I presented a DFA relying on a single-bit fault model and have proven to successfully perform it using BBI. Then, thanks to the modeling of BBI I have set up, I have shown that BBI induce bit set/reset faults which are data dependent. In addition to this, thanks to the SCS simulations, I have shown that most of the electric charges concentrate under the injection point, which can be put in parallel to the local nature of laser fault injection.

V

Substrate thinning analysis

Contents

5.1 Summary

In this chapter, I study the interests of thinning the substrate of integrated circuits in a BBI context. Although this topic had been studied for LFI in the past [43], it was not the case for BBI at the beginning of my thesis.

To that end, I divided the work into three parts. First, I present a geometric approach to IC substrate thinning, allowing a high degree of abstraction from electronics. Second, thanks to the models I introduced in the Chapter 3, I introduce simulation results for various substrate thicknesses, allowing to verify the soundness of the geometric approach. Eventually, I present actual experiments performed on three identical IC targets where their substrate have been thinned to different amount.

5.2 Introduction

When working with integrated circuits in a fault injection context, several physical parameters of the device under test are of great importance. For example, as I have stated in the previous chapter, the type of substrate used to manufacture the IC has a significant impact on BBI efficiency and behavior, more specifically due to the coupling difference between the IC and the probe. In addition to this, the size of the transistors, the power supply voltage, the IC package or the IC substrate thickness can drastically change fault injections results. Among these examples, one of great interest for body biasing injection is the substrate thickness, as the silicon substrate is the environment carrying the charges allowing the input of energy into the target.

According to the manufacturing process, and depending on the purpose of the manufactured IC, it is common to find various substrate thicknesses. On the one hand, it is not rare to find 700 µm thick wafers with 300 mm diameters for generic applications such as microcontrollers or microprocessors. On the other hand, in specific applications like SoCs, where vertical stacking is commonly used, or in Smart-cards and ID cards, the typical substrate thickness value is lower, around 150 µm.

In addition to these differences one can find in commercial products, the practice of thinning the substrate of ICs is not uncommon in a fault injection context. More specifically, substrate thinning has been thoroughly studied concerning Laser Fault Injection (LFI) [43, 44], and has proven to greatly enhance LFI efficiency, in addition to drastically reducing the power required to create faults. However, as it had not been studied for Body Biasing Injection at the beginning of this work, I decided to look into the matter.

In this context, this work was first done in order to assess whether substrate thinning has similar effects on BBI as it has on LFI. Second, because thin ICs commonly found in smart-cards have unavoidable security constraints, and third because BBI is performed using the silicon substrate as the physical environment to carry energy through electric charges. Therefore, this chapter evaluates the interests of substrate thinning on BBI efficiency. In other words, I am presenting the analysis of the electrical and behavioral differences between identical ICs with different substrate thicknesses. This

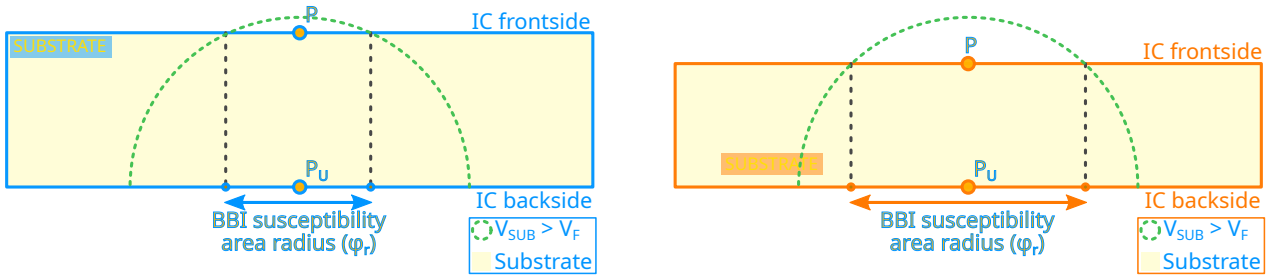
analysis will take place using multiple approaches.

In the first place, I address the question using a geometric approach to appreciate the effects of substrate thinning on voltage propagation inside the substrate while taking a step back from electrical modeling. Then, the geometric approach is completed with an electrical simulation analysis of two identical ICs with different substrate thicknesses, created thanks to the models I proposed in Chapter 3. Eventually, experimental results are analyzed in order to verify the correctness of the previous approaches, in addition to studying the actual effects of substrate thinning concerning faults creation.

5.3 Theoretical modeling: the effects of substrate thickness on BBI

In this section, I first present the geometric approach, allowing an abstraction from electronics, enabling faster and easier modeling at first. Then, and because this modeling alone is insufficient, I perform and analyze electric IC simulations to strengthen the geometric approach conclusions.

5.3.1 Geometric modeling



(a) *Normal IC* **(b)** *Thinned IC*

Figure 5.1: *BBI susceptibility area cross-sectional 2D view*

For the purpose of the geometric modeling, let us consider two identical ICs. A commercial one, with an arbitrary standard substrate thickness, and another one with its substrate thinned by a certain amount in order to perform fault injection more easily. Fig. 5.1 illustrates the two-dimensional cross-sectional views of the considered ICs substrates during an arbitrary BBI voltage pulse of amplitude V_{P_U}. The probe is located at the center of the ICs at point P_U, and the targeted spot in the logic is at point P. The silicon substrate being an isotropic resistive environment, it is quite natural to expect the electrical charges to flow and spread evenly when injected into it at any given time. Therefore, equipotentials form half-sphere surfaces inside the substrate volume. These surfaces are highlighted in two-dimensions as green half-circles in Fig. 5.1, for an arbitrary voltage V_F.

In this scenario, an attacker wants to induce a fault in the logic gates, located at the top of each IC. To that end, they need to change the voltage enough at point P, called V_P, in order to disturb the transistors and change the logic gates behavior for a short amount of time. In addition to that, and for the sake of simplicity, let us assume that P is the only location in the considered IC where

faults can be injected. However, in order to observe faults at point P, V_P needs to reach a minimal threshold voltage, called V_F. Because the attacker is working with BBI, a metallic probe is connected onto the backside of the IC, at point P_U, in order to inject energy into the IC. Depending on the amount of injected energy, in other words, the maximum amplitude of the voltage pulse because the substrate effective electric resistance is static over time, the voltage at P might never reach V_F, therefore, no faults will be observed. Let us consider that the attacker chose an amplitude V_{PU} big enough such that at a given moment in the injection, V_P reaches V_F or more in each considered IC. In that scenario, the area on the IC front side where $V > V_F$ is a disk of radius ϕ, centered in P, called the BBI susceptibility area radius. It means that the attacker can position the probe anywhere on the backside within this disk to reach V_F at P, and therefore induce a fault at P.

Let me introduce the various equations enabling the calculation of the various parameters I previously mentioned. The half-sphere equipotential radius relative to time can be determined thanks to the following formula:

$$r(t) = \frac{\rho_{SUB}}{\sqrt{2}} \cdot \frac{|I_G(t)|}{|V_{PU}(t) + V_F|} \tag{5.1}$$

with ρ_{SUB} the resistivity of the silicon substrate, $I_G(t)$ the instantaneous sum of the current distribution contained in the half-sphere, and $V_{PU}(t)$ the instantaneous voltage pulse applied on the backside of the IC.

Then, logically, the BBI susceptibility area radius, denoted ϕ_r, is given by:

$$\phi_r(t) = 2 \cdot \sqrt{r(t)^2 - t_{SUB}^2} \tag{5.2}$$

with t_{SUB} being the IC substrate thickness.

As illustrated in Fig. 5.1, thinning the substrate inevitably increases the size of the susceptibility area if the experimental conditions are constant. It means that the susceptibility evolution ratio is always greater than 1 when thinning the substrate for a fixed disturbance:

$$\frac{\phi_r^{THIN}}{\phi_r^{THICK}} = \sqrt{\frac{r^2 - t_{THIN}^2}{r^2 - t_{THICK}^2}} > 1 \tag{5.3}$$

Therefore, in order to obtain the same susceptibility area with a thinner IC, it is required to reduce the voltage pulse amplitude, thanks to the following relation:

$$V_{PU}^* = \frac{t_{THIN}}{t_{THICK}} \cdot V_{PU} + V_F \cdot (1 - \frac{t_{THIN}}{t_{THICK}}) \tag{5.4}$$

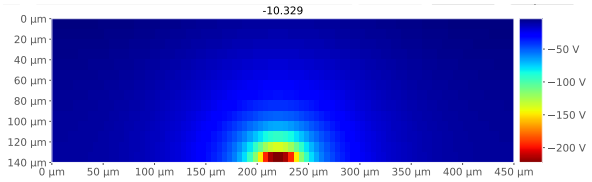
Eventually, this geometric approach allows deducing three related outcomes:

1. Thinning the substrate allows reducing the minimal voltage pulse amplitude required to induce a fault while keeping a constant susceptibility area.
2. The BBI susceptibility area increases while the substrate thickness decreases while working at

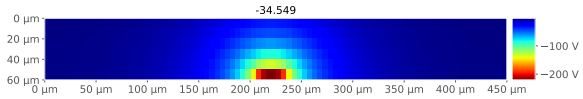
a constant voltage pulse V_{PU}.

3. Thinning the substrate alone does not have an influence on BBI spatial resolution, as the susceptibility area depends on the couple (t_{SUB}, V_{PU}). Thus, similar spatial resolution could be obtained with different substrate thicknesses by changing V_{PU}.

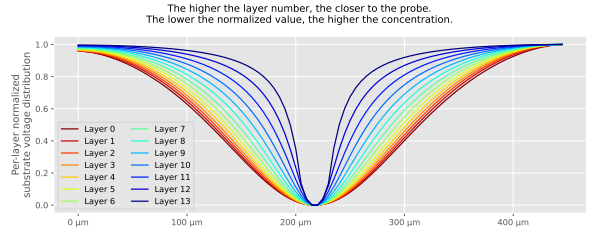
5.3.2 Electric approach



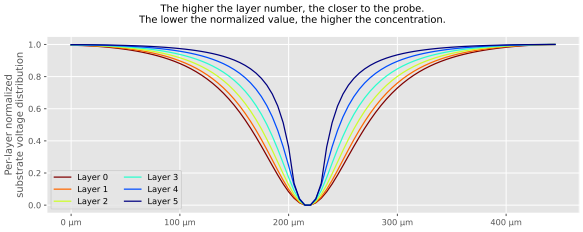
(a) *140 µm IC substrate voltage distribution cross-sectional view (Y-axis).*



(b) *60 µm IC substrate voltage distribution cross-sectional view (Y-axis).*



(c) *140 µm IC substrate per-layer normalized voltage concentration.*



(d) *60 µm IC substrate per-layer normalized voltage concentration.*

Figure 5.2: *Simulated non-thinned IC (140 µm) and thinned IC (60 µm) substrate voltage distribution cross-sectional view and normalized voltage concentration at the peak amplitude of the first voltage pulse edge.*

As I stated previously, in order to verify the meaningfulness of the geometrical approach, I complete it with an electrical modeling approach. This approach takes advantage of the models I previously introduced in chapter 3. Then, thanks to these models, I am able to generate identical ICs with different substrate thicknesses. After that, I simulate the circuits under BBI and I am able to plot the voltage distribution inside the substrate for each simulated IC.

I chose to study two distinct substrate thicknesses: 60 µm and 140 µm. The simulation parameters are the following:

- Dual-Well exclusive substrate
- Required voltage pulse: -300 V
- Required pulse width: 20 ns
- Required rise and fall times: 8 ns

Fig. 5.2 shows, for each simulated IC, the voltage distribution across the substrate through a cross-sectional view at the apex of the voltage pulse first edge, in addition to the per-layer voltage concentration. These last graphs allow me to spot more easily the way the voltage propagates and is distributed inside the substrate during BBI. For simplicity, the results are shown in two dimensions, and as the substrate is an isotropic environment, the propagation is uniform in every direction.

The first interesting thing to note is that, as predicted thanks to the geometric model and as shown in Fig. 5.2a and 5.2b, equipotentials effectively form half-circles into the substrate (which are effectively half-spheres in 3D). However, these figures alone do not say much, at least not very obviously, about the voltage spreading inside the substrate. Then, thanks to Fig. 5.2c and 5.2d, which represent, for each substrate layer (14 in number concerning the 140 µm IC, 6 for the 60 µm IC), the normalized voltage distribution. The lower the layer number (or the closer to the red color the line is), the deeper into the IC it is. Concerning the values in the X-axis, there are normalized voltage values calculated per-layer, and the closer to zero, the higher the concentration. What these figures show us is that for identical experimental parameters, when the substrate is thinned, the voltage concentration is higher in the highest substrate layer. It means that for a fixed V_F voltage threshold, the BBI susceptibility area is more spread out on a thinner IC. It is quite intuitive when looking at the simulation results. Indeed, if we simply remove the additional layers in Fig. 5.2a and 5.2b between the 60 µm and the 140 µm IC, the results look very similar. The same observation can be made concerning Fig. 5.2c and 5.2d.

5.4 Models validation

5.4.1 A few words about IC substrate thinning

As substrate thinning is quite widespread when performing fault injection, especially concerning LFI, let me explain how it is commonly performed. The first step to thin the substrate of an IC is to get direct access of it. To do so, it is required to remove part of the plastic package of the IC. There are multiple ways of doing it, sometimes using chemical solutions do dissolve the material, but most commonly using milling processes. More specifically using Selected Area Preparation (SAP) or Focused Ion Beams (FIB) milling.

On the one hand, SAP milling consists in a very precise mechanical milling tool, generally able to remove material with a precision down to a few micrometers. It is used to remove the package and at the same time thin the IC substrate. However, it can often lead to uneven surfaces, which then requires polishing.

On the other hand, FIB milling consists in a physical milling which does not imply a mechanical contact with the material to be removed, and allows nanometer-level precision. For that purpose, FIB is commonly used in combination with SAP [45] to produce even substrate surfaces.

In addition to substrate thinning and plastic package removal , SAP milling machines allow removing eventual internal metallic heat-sinks of ICs prior to substrate thinning. It has the advantage of providing low damage to thinned ICs, thanks to low spindle speed and low temperature rise compared to traditional high speed milling.

5.4.2 Experiments with thinned circuits

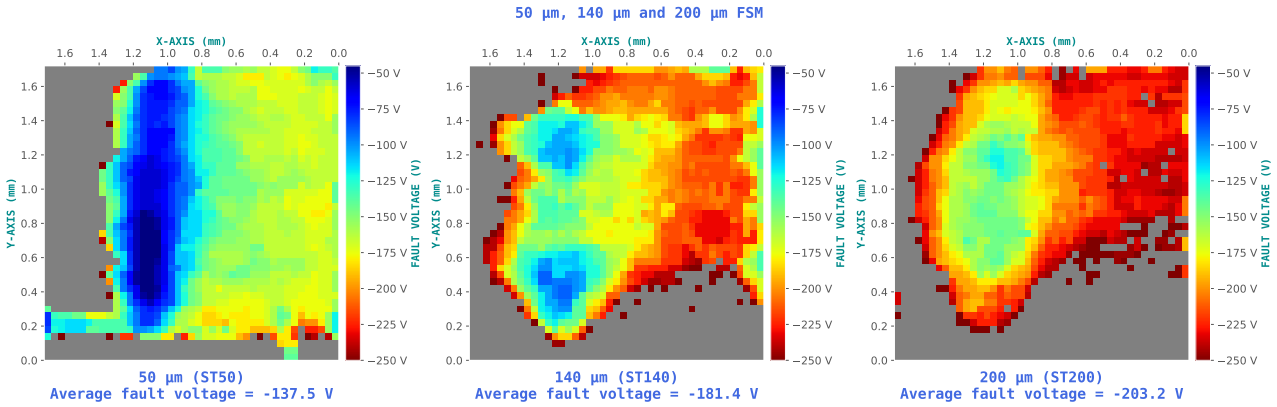


Figure 5.3: *Fault susceptibility maps*

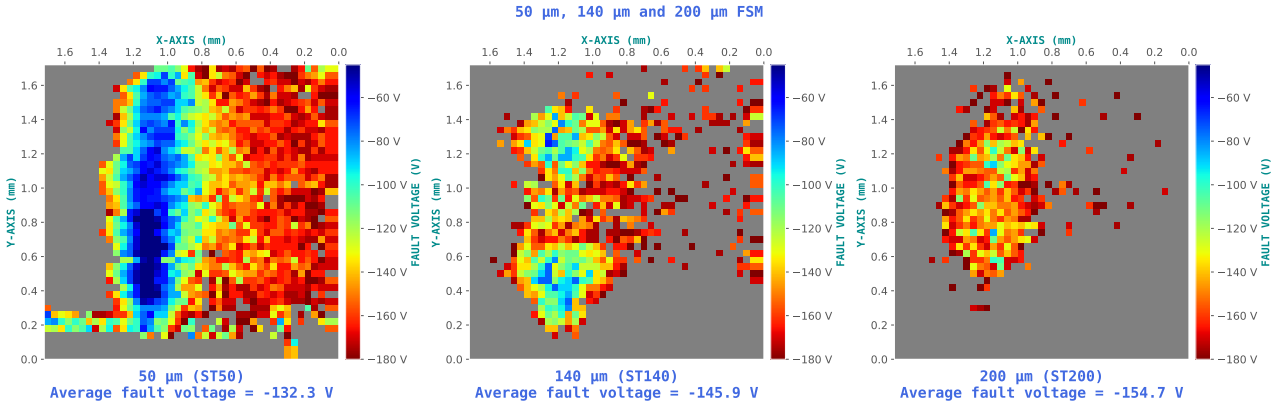


Figure 5.4: *Susceptibility area spreading*

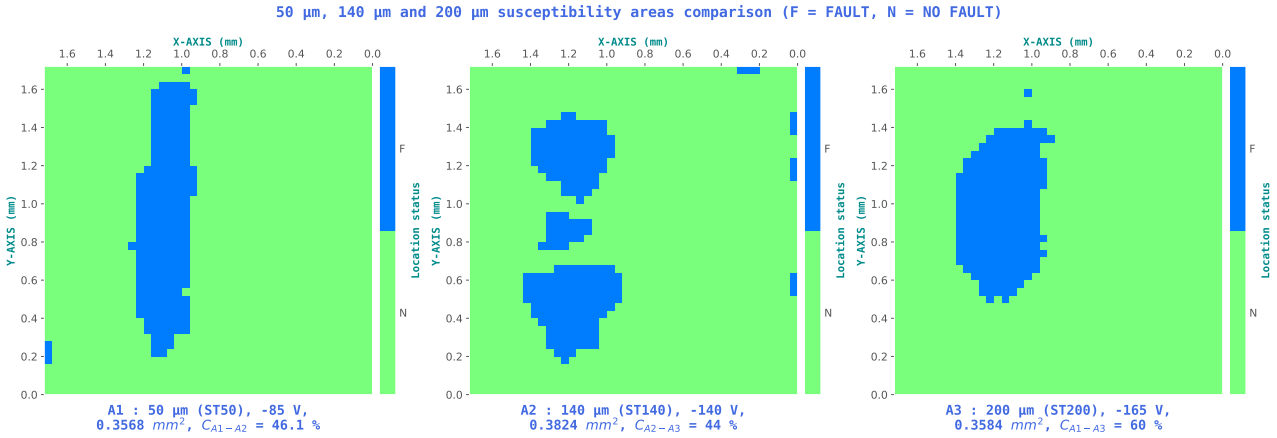


Figure 5.5: *Fault susceptibility maps couples*

With geometric and electrical modeling complete, it is now possible to conduct actual experiments in order to verify the meaningfulness of the previous approaches. In this context, three an STM32F439VIT6 LQFP100 identical targets were thinned to three different levels, from 750 µm to respectively 200 µm, 140 µm and 50 µm, respectively named ST200, ST140 and ST50 for the rest of this Chapter. In order to verify the three conclusions extracted from the modeling section, three experiments are conducted for each target.

The first experiment aims at measuring the minimal voltage pulse amplitude V_{PU}^{MIN} required to induce a faulty behavior on an IC performing computations. These experiments are called Fault Susceptibility Maps (FSM). They allow spotting the region where the IC is sensitive to BBI, no matter which type of induced fault. Therefore, when mapping an entire IC, it is common to spot various areas not directly involved in the targeted calculation, like the analog voltage regulator or the FLASH memory logic control logic not to cite them all. As a result, and because in a fault injection context the cryptographic core is very often targeted, it was decided to focus the maps above the STM32 AES core only. Fig. 5.3 presents the three performed FSM. From left to right, t_{SUB} goes from 50 µm, then to 140 µm, finally to 200 µm. As stated before, the maps are performed above the hardware AES core of the IC, temporally aiming the penultimate AES round. The scanned area measures 1.7 mm by 1.7 mm, with a displacement step of 40 µm between each point. V_{PU} was limited to the following range: [30 V ; 280 V], with 5 V steps and a negative polarity. The pulse width was fixed at 6 ns. The first important thing to note here is that, as predicted with the geometric and electrical modelings, a thinner substrate allows a lower fault induction threshold. It is mainly shown thanks to the measurement of the average voltage required to induce a fault across the entire map, annotated at the bottom of each map. All of this sustains the first conclusion made in section 5.3.

Then, the second experiment, whose results are shown in Fig. 5.4, consist in analyzing the spreading of the BBI susceptibility area. The core of the experiment is identical as before. However, in order to highlight the spreading effect, it was required to set a lower maximum voltage amplitude (in absolute value). The value of 180 V was chosen as it is the average voltage of the medium-thinned IC. What is interesting here is that, for the ST200 target, because the voltage at the epitaxy level cannot reach the threshold value V_F in most cases, the fault area is tiny compared to the other targets, and focused on the AES core. Then, concerning the ST140 target, thanks to the thinner substrate, the voltage at the epitaxy level can reach a higher value, and thus can cause more logic gates or further logic gates from the probe to have a faulty behavior. Eventually, the ST50 target shows the largest fault area. These experiments help to sustain the second conclusion of section 5.3.

Eventually, the last experiment, where the results are displayed in Fig. 5.5, consisted in finding, whenever possible, (t_{SUB}, V_{PU}) couples for which the susceptibility area is identical across all three targets. To do so, we chose the couple $(200\ \mu m, -165\ V)$ as the base search value. Afterward, we swept across every possible couple for ST140 and chose the one with the closest area and correlation to the base ST200 couple. Then, we did the same thing between ST140 and ST50, giving us the three following couples for each IC:

- ST50: $(50\ \mu m, -85\ V)$, with a fault susceptibility area of $0.3568 mm^2$;
- ST140: $(140\ \mu m, -140\ V)$, with a fault susceptibility area of $0.3824 mm^2$;
- ST200: $(200\ \mu m, -165\ V)$, with a fault susceptibility area of $0.3584 mm^2$.

5.5 Conclusion

In this chapter, I introduced the interest of thinning the substrate of integrated circuits in a body biasing injection context. Various works have studied this topic concerning laser fault injection [43], and because the substrate is the physical environment where the energy is carried to the IC when performing BBI, I decided to take a deep look on substrate thinning in this context.

In the first place, I ideally analyzed theoretical BBI effects thanks to a geometric model, which allowed me to conclude that the resolution of BBI not only depends on the substrate thickness but on the couple thickness/voltage. On the one hand, thinning the substrate and reducing the voltage pulse amplitude should lead to identical results on a fixed target. On the other hand, thinning the substrate while keeping the same voltage leads to a spread out BBI susceptibility area. Afterward, I conducted the same protocol using the electrical models presented in the chapter 3. Using an analogous analysis as in that chapter, I observed that the thinner the IC, for a fixed set of parameters, the higher the voltage is focused under the probe. Therefore, the susceptibility area increases, which means that less voltage, therefore a less powerful generator, is required to create a very similar effect. These outcomes were verified thanks to actual experiments on three identical ICs with their substrate thinned to various levels. Eventually, the main thing to remember concerning substrate thinning and BBI is that, as opposed to LFI, thinning the substrate is not necessary while the attacker has a powerful enough generator. However, sometimes, thinning the substrate can financially be more interesting than purchasing a more powerful pulse generator.

VI

Conclusion and perspectives

Contents

6.1 Contributions

6.2 Outlooks

6.3 Final remarks

Appendix

Standard-cell segment and integrated circuit generation algorithm

```

# -*- coding: utf-8 -*-
"""
Created on Fri Jun  9 12:42:15 2023.

This is the version 11.1 of a HSPICE netlists generator which creates simulable ICs based on
standard cells segments models from Geoffrey CHANCEL researches about BBI.
This generator automatically connects together a set of elementary netlists
to form a complete IC, with explorable silicon substrate and power distribution network.
The power distribution networks is connected to the IC from the outside through
Metal TOP layer external pins, laid out as an external ring, which are equipotentials.
The IC generated uses a mix of Triple-Well and Dual-Well elementary blocks. They are
connected to each other in a similar way of a real STM32F439VIT6 die. However, as it is
extremely heavy to simulate a real sized IC, which equals approximately 4.5 mm x 5.5 mm,
it is proposed here to reduce its total size, according to <mul> value, which is to be
set by the user.

@author: Geoffrey Chancel alias geoff2.
"""


import os
import sys
import gzip
import time
import argparse
# import colorama
import numpy as np
# import defLibGenV11 as const
import HSPICE_gen_test_twordw_v11p1 as pgen

from progress.bar import Bar
from defLibGenV11 import (ProbeError, SUBCKT_BEGIN_TW, BLOC_TW, SUBCKT_BEGIN_DW,
                          BLOC_DW, GEN_OK, ELEM_BLOCKS_BEGIN, ELEM_BLOCKS_END, GEN_ERROR,
                          WSEG, WSEG6, HSEG, BBI_END_V11_mixed, BBI_END_V11_TW, BBI_END_V11_DW)

VER = "V11.1"
VINT = 11.1


def dual_or_triple(x_l, y_l, mul_ic, ext = False, s_x = None, s_y = None):
    """
    Determine if a dual well or triple well block is to be used.

    Parameters
```

```
    ----------
    X : list
        X coordinates of block.
    Y : list
        Y coordinates of block.
    m: float
        Ratio of circuit size.
    ext : boolean
        Use the extended function.

    Returns
    -------
    elem_bloc: str
        The corresponding elementary block string.

    """
    global tex
    global tey

    ext = True
    mul_ic = 1.0

    x_l = [float(i.replace('p', '.')) for i in x_l]
    y_l = [float(i.replace('p', '.')) for i in y_l]
    if not ext:
        if (x_l[0] >= 0 and x_l[0] < int(60 * mul_ic)):  # LEFT VERTICAL BAND
            return "elem_blocDW"
        elif (x_l[0] >= int(840 * mul_ic) and x_l[0] <= int(900 * mul_ic)):  # RIGHT VBAND
            return "elem_blocDW"
        elif (y_l[0] >= int(0 * mul_ic) and y_l[0] < int(60 * mul_ic)):  # BOTTOM V BAND
            return "elem_blocDW"
        elif (y_l[0] >= int(1040 * mul_ic) and y_l[0] <= int(1100 * mul_ic)):  # BOTTOM VBAND
            return "elem_blocDW"
        elif (y_l[0] >= 0 and y_l[0] <= int(340 * mul_ic)
              and x_l[0] >= int(330 * mul_ic) and x_l[0] < int(570 * mul_ic)):
            return "elem_blocDW"
        elif (y_l[0] > int(340 * mul_ic) and y_l[0] < int(405 * mul_ic)):
            return "elem_blocDW"
        elif (y_l[0] >= int(700 * mul_ic) and y_l[0] < int(735 * mul_ic)
              and x_l[0] >= int(60 * mul_ic) and x_l[0] < int(360 * mul_ic)):
            return "elem_blocDW"
        elif (y_l[0] >= int(700 * mul_ic) and y_l[0] < int(1100 * mul_ic)
              and x_l[0] >= int(330 * mul_ic) and x_l[0] < int(360 * mul_ic)):
            return "elem_blocDW"
        else:
            return "elem_blocTW"

    else:
        if s_x is None or s_y is None:
            raise ValueError("If ext is True, sx and sy must be float or integers")

        mulx = tex / 900
        muly = tey / 1100

        m60, m840, m900, m1040 = 60 * mulx, 840 * mulx, 900 * mulx, 1040 * muly
        m1100, m700 = 1100 * muly, 700 * muly
        m330, m540, m340, m405 = 330 * mulx, 570 * mulx, 340 * muly, 405 * muly
        m735, m700, m1100, m360 = 735 * muly, 700 * muly, 1100 * muly, 360 * mulx
        if m60 < 30:
            m60 = 30
```

```
106         if tex - m840 < 60:
107             m840 = m840 - 60 + (m900 - m840)
108         if tey - m1040 < 35:
109             m1040 = m1040 - 35 + (m1100 - m1040)
110         # if m*(m840 - m570 )
111         # if m
112 
113         if (x_l[0] >= 0 and x_l[0] < int(m60)):  # LEFT VERTICAL BAND
114             return "elem_blocDW"
115         elif (x_l[0] >= int(m840) and x_l[0] <= int(m900)):  # RIGHT VERTICAL BAND
116             return "elem_blocDW"
117         elif (y_l[0] >= int(0 * mul_ic) and y_l[0] < int(m60)):  # BOTTOM VERTICAL BAND
118             return "elem_blocDW"
119         elif (y_l[0] >= int(m1040) and y_l[0] <= int(m1100)):  # BOTTOM VERTICAL BAND
120             return "elem_blocDW"
121         elif (y_l[0] >= 0 and y_l[0] <= int(m340)
122               and x_l[0] >= int(m330) and x_l[-1] < int(m540)):
123             return "elem_blocDW"
124         elif (y_l[0] > int(m340) and y_l[0] < int(m405)):
125             return "elem_blocDW"
126         elif (y_l[0] >= int(m700) and y_l[0] < int(m735)
127               and x_l[0] >= int(m60) and x_l[0] < int(m360)):
128             return "elem_blocDW"
129         elif (y_l[0] >= int(m700) and y_l[0] < int(m1100)
130               and x_l[0] >= int(m330) and x_l[0] < int(m360)):
131             return "elem_blocDW"
132         else:
133             return "elem_blocTW"
134 
135 
136 class SubstrateGeneration(object):
137     """
138     Class to generate the substrate netlist.
139 
140     ...
141 
142     Attributes
143     ----------
144     subType : str
145         The desired substrate type, either 'TW', 'DW' or 'mixed'
146 
147     Methods
148     -------
149     get_status():
150         Return the status of substrate generation.
151     """
152 
153     def __init__(self, sub_type):
154         global bbi_gen
155         global nH
156         global tSub
157         if sub_type == "mixed":
158             bbi_gen.write(f"\n*elementary bloc {tSub}umTW")
159             bbi_gen.write(SUBCKT_BEGIN_TW)
160             self.__output__(sub_type)
161             bbi_gen.write(BLOC_TW)
162             bbi_gen.write(f"\n*elementary bloc {tSub}umDW")
163             bbi_gen.write(SUBCKT_BEGIN_DW)
164             self.__output__(sub_type)
165             bbi_gen.write(BLOC_DW)
```

```
166             bbi_gen.write("\n")
167             self.__return__ = GEN_OK
168         elif sub_type == "TW":
169             bbi_gen.write(f"\n*elementary bloc {tSub}umTW")
170             bbi_gen.write(SUBCKT_BEGIN_TW)
171             self.__output__(sub_type)
172             bbi_gen.write(BLOC_TW)
173             bbi_gen.write("\n")
174             self.__return__ = GEN_OK
175         elif sub_type == "DW":
176             bbi_gen.write(f"\n*elementary bloc {tSub}umDW")
177             bbi_gen.write(SUBCKT_BEGIN_DW)
178             self.__output__(sub_type)
179             bbi_gen.write(BLOC_DW)
180             bbi_gen.write("\n")
181             self.__return__ = GEN_OK
182         else:
183             self.__return__ = GEN_ERROR
184 
185     def get_status(self):
186         """
187         Return the status of substrate generation.
188 
189         Parameters
190         ----------
191         None.
192 
193         Returns
194         -------
195         int
196             The value of the generation status.
197 
198         """
199         return self.__return__
200 
201     def __output__(self, sub_type):
202         """
203         ss.
204 
205         Parameters
206         ----------
207         sub_type : str
208             The desired substrate type (TW, DW or mixed).
209 
210         Returns
211         -------
212         None.
213 
214         """
215         global bbi_gen
216         global nH
217         bbi_gen.write("+")
218         for prof in range(1, nH + 1, 1):
219             if prof < nH + 1 - 1:
220                 bbi_gen.write(f"V_L_{prof} ")
221             else:
222                 bbi_gen.write(f"V_L_{prof}\n")
223         bbi_gen.write("+")
224         for prof in range(1, nH + 1, 1):
225             if prof < nH + 1 - 1:
```

```
                bbi_gen.write(f"V_R_{prof} ")
            else:
                bbi_gen.write(f"V_R_{prof}\n")
        for h in range(1, nH + 1, 1):
            for i in range(1, 7, 1):
                if i == 1:
                    bbi_gen.write("+")
                bbi_gen.write(f"V_F{i}_{h} ")
            for i in range(1, 7, 1):
                bbi_gen.write(f"V_RE{i}_{h} ")
            bbi_gen.write("\n")
        bbi_gen.write("+VDOWN_1 VDOWN_2 VDOWN_3 VDOWN_4 VDOWN_5 VDOWN_6\n\n")
        # Internal signals generation
        NG = []
        for h in range(1, nH + 1, 1):
            NGloc = []
            if h < nH:
                for u in range(0, 6, 1):
                    NGloc.append((u + 1) + 6 * (h - 1))
                NG.append(NGloc)
            bbi_gen.write(f"XX{h} ")
            if h == nH:
                bbi_gen.write("VDOWN_1 VDOWN_2 VDOWN_3 VDOWN_4 VDOWN_5 VDOWN_6 ")
            else:
                for ng in NGloc:
                    bbi_gen.write(f"NG{ng} ")
            for i in range(1, 7, 1):
                bbi_gen.write(f"V_F{i}_{h} ")
            bbi_gen.write(f"V_L_{h} V_R_{h} ")
            for i in range(1, 7, 1):
                bbi_gen.write(f"V_RE{i}_{h} ")
            if h == 1:
                for i in range(6):
                    bbi_gen.write("vepi ")
            else:
                for ng in NG[h - 1 - 1]:
                    bbi_gen.write(f"NG{ng} ")
            bbi_gen.write(f"VSUBC{h} elementary_blocx6\n")


def main():
    """
    Execute the main code.

    Raises
    ------
    ValueError
        DESCRIPTION.
    ProbeError
        DESCRIPTION.

    Returns
    -------
    None.

    """
    # ============================================================================
    # Global variables
    # ============================================================================
    global bbi_gen
```

```
286     global nH
287     global tSub
288     global nC
289     global nL
290     global tex
291     global tey
292
293     # ================================================================================
294     # CLI INTERFACE
295     # ================================================================================
296     desc = '''Generate a netlists of standard-cells according to user settings.
297     One can specify the required substrate type, the model type (bad gnd, good gnd, good gnd
298                                                                     + impedance matching).
299     The user has to specify the IC size (see the parameters). When choosing a size,
300     the program might resize the IC according to elemetary models size, as they are
301     unbreakable. Please also note that 1:1 ratios might often produce impossible results.
302     If the generator does not manage to generate what you request, it will notify you and abort
303     the generation process, asking you to change the IC size.
304     In addition to this,, <mul> can be specified to change the global size of the IC.
305     Note that some configurations may not work properly.
306     '''
307
308     parser = argparse.ArgumentParser(prog = "python ./hspiceGeneratorV11cli.py",
309                                      description = desc,
310                                      epilog = "gchancel2023")
311
312     parser.add_argument("-s", dest = "subType",
313                         help = "Substrate type, can take 3 values: <mixed>, <tw> or <dw>",
314                         required = True)
315
316     parser.add_argument("-m", dest = "model",
317                         help = "Model type, can take 3 values: <0>, <1>, or <2>",
318                         default = "2",
319                         type = int,
320                         required = False)
321
322     parser.add_argument("-t", dest = "tSub",
323                         help = '''Substrate thickness, integer between 10 and infinity,
324                                 multiple of 10''',
325                         required = True, type = int)
326
327     parser.add_argument("-eT", dest = "elemThickness",
328                         help = '''Thickness of the elementary substrate block to choose.
329                                 A lower value brings more precision, a higher value decreases
330                                 precision but increases simulation speed for a constat tSub.
331                                 Can take any floating point value
332                                 between 10 and 1000 (in Âţm).''',
333                         required = True, type = int)
334
335     parser.add_argument("--mul",
336                         help = '''Resize factor (float)''',
337                         default = "1.0",
338                         type = float,
339                         required = False)
340
341     parser.add_argument("-v", dest = "vpUU",
342                         help = '''Amplitude of the applied pulse (in V)''',
343                         type = float,
344                         required = True)
345
```

```
346     parser.add_argument("-p", dest = "pW",
347                         help = '''Pulse width (in ns)''',
348                         type = float,
349                         required = True)
350
351     parser.add_argument("-tr", dest = "tFR",
352                         help = '''Rise/fall time (ns)''',
353                         type = float,
354                         required = True)
355
356     parser.add_argument("--step", dest = "sim_step",
357                         help = '''Simulation time step (specifiy the unit without space):
358                             example: --step 50ps''',
359                         required = True)
360
361     parser.add_argument("--time", dest = "sim_time",
362                         help = '''Simulation duration (specify the unit without space):
363                             example: --time 40ns''',
364                         required = True)
365
366     parser.add_argument("--auto", dest = "auto",
367                         help = '''Execute the generator without user prompt.''',
368                         required = False,
369                         default = "False",
370                         type = str)
371
372     parser.add_argument("--res", dest = "res",
373                         help = '''Display the result graphically for visual inspection.''',
374                         required = False,
375                         default = "True",
376                         type = str)
377
378     parser.add_argument("-tex", dest = "tex",
379                         help = '''X size''',
380                         type = float,
381                         required = True)
382
383     parser.add_argument("-tey", dest = "tey",
384                         help = '''Y size''',
385                         type = float,
386                         required = True)
387
388     parser.add_argument("--psInt", dest = "plotSubInt",
389                         help = '''Output all substrate interior signals into output file''',
390                         type = str,
391                         default = "False",
392                         required = False)
393
394     parser.add_argument("--compress", dest = "compress",
395                         help = '''Compress output file and remove non compressed file.''',
396                         type = str,
397                         default = "False",
398                         required = False)
399
400     parser.add_argument("--af", dest = "appendFname",
401                         help = '''Append custom string to the end of filename.''',
402                         type = str,
403                         default = "",
404                         required = False)
405
```

```
406     args = parser.parse_args()
407     print("\nThe settings you entered are the following:")
408     print(f"SubType = {args.subType}")
409     print(f"Model = {args.model}")
410     print(f"Thickness = {args.tSub} Âţm")
411     print(f"Elementary thickness = {args.elemThickness} Âţm")
412     print(f"Mult = {args.mul}")
413     print(f"Vpulse = {args.vpUU} V")
414     print(f"Pulse width = {args.pW} ns")
415     print(f"Rise/fall time = {args.tFR} ns")
416     print(f"Simulation time step = {args.sim_step}")
417     print(f"Simulation duration = {args.sim_time}")
418     print(f"Auto mode = {args.auto}")
419     print(f"Display mode = {args.res}")
420     print(f"X size = {args.tex}")
421     print(f"Y size = {args.tey}")
422     print(f"Plot Substrate Interior = {args.plotSubInt}")
423 
424     if not args.auto == "True":
425         input("\n!!! PRESS ENTER TO CONTINUE !!!\n")
426         locked = True
427         msg = '''Are you sure to continue? Please answer with 'y'
428         to continue or with 'n' to abort.'''
429         while locked:
430             print(msg)
431             choice = input().lower()
432             if choice == 'n':
433                 sys.exit(0)
434             elif choice == 'y':
435                 locked = False
436 
437     print("Netlist generation in progess")
438 
439     # ============================================================================
440     # IC SIZE SECTION (USER CUSTOMIZATION)
441     # ============================================================================
442     # mul = 0.35  # Set the scale used to reduce or increase original IC size.
443     mul = args.mul  # Set the scale used to reduce or increase original IC size.
444     # tex = mul * 900  # Final x size of the IC. DO NOT CHANGE ON ANY SITUATION !!!
445     # tey = mul * 1100  # Final y size of the IC. DO NOT CHANGE ON ANY SITUATION !!!
446     tex = args.tex * mul  # Final x size of the IC. DO NOT CHANGE ON ANY SITUATION !!!
447     tey = args.tey * mul  # Final y size of the IC. DO NOT CHANGE ON ANY SITUATION !!!
448 
449     nC = int(tex / 30.0)  # According to the previously computed size, final number of columns.
450     nL = int(tey / 5.0)  # According to the previously computed size, final number of lines.
451     tex = int(nC * 30.0)
452     tey = int(nL * 5.0)
453     tSub = args.tSub  # Substrate thickness in Âţm.
454     eT = args.elemThickness  # Thickness of an elementary substrate block.
455     nH = int(tSub / eT)  # (Number of silicon substrate layers).
456     print(f"\033[1;32mNumber of substrate layers = {nH}")
457     resultingTSUB = nH * eT
458     print(f"Resulting Substrate chosen = {resultingTSUB} Âţm\033[1;0m")
459 
460     if args.subType == "mixed":
461         mixedDWTW = True
462         TWorDW = True
463     elif args.subType == "tw":
464         mixedDWTW = False
465         TWorDW = True
```

```
    elif args.subType == "dw":
        mixedDWTW = False
        TWorDW = False

    bar = Bar("Generation", max = nL * nC)

    model = args.model  # Can take 3 different values : 0, 1 or 2.
    '''
    Model 0 corresponds to the first BBI attempts, with bad grounding and impedance mismatch.
    Model 1 corresponds to Good grounding and impedance mismatch.
    Model 2 corresponds to Good Grounding + Impedance matching.
    '''

    vpUU = args.vpUU  # Amplitude of the voltage pulse.
    pW = args.pW  # Pulse width (in ns)
    tFR = args.tFR  # Fall/Rise time (in ns)
    sim_step = args.sim_step  # Time step of the simulation.
    sim_time = args.sim_time  # Duration of the simulation.

    Rm1 = '26'
    Rmup = '5'
    Cdecp = '2.25f'
    Ra = '2'
    Cgp = '35.2f'
    Cgn = '25.2f'
    Rp = '9.57'
    Rn = '5.3'
    Cnw = '20f'
    RcontactTW = '3.1k'
    RcontactDW = '3.1k'

    probe_count = 0

    # ============================================================================
    # BBI PROBE SECTION: !!! DO NOT TOUCH, NOT FINALIZED AND HIGHLY PRONE TO PRODUCE
    # UNEXPECTED RESULTS. !!! (BETA TEST, INCONSISTENT)
    # ============================================================================
    probe_center = True
    c_probe = True  # BBI PROBE CONNECTION (ONLY PARAMETER ALLOWED TO BE MODIFIED WITH CARE).
    if probe_center:
        # prbX = tex / 2.0
        # prbY = tey / 2.0
        prbX = int((nC * 30.0) / 2.0)
        prbY = int((nL * 5.0) / 2.0)
        # print(f"PROBE X = {prbX}")
        # print(f"PROBE Y = {prbY}")
    else:  # Custom probe position (may leads to inconsistencies, use with care).
        prbX = 210 * mul
        prbY = 890 * mul
    # DEFAULT PROBE IS 30 Âţm * 30 Âţm, not anything else. Other sizes will be added in
    # future versions of this script. Default location is the center of the IC, no choice.

    # ============================================================================
    # Development test variables, do not touch. (ALPHA TEST, unstable)
    # ============================================================================
    # cgnd = 0
    # cvdd = 0

    # xvdd = []
    # xgnd = []
```

```
526
527     x_trace = []
528     y_trace = []
529
530     if mul == 1:
531         ext_loc = False
532     else:
533         ext_loc = True
534
535     ring = False
536     if ring:
537         print('RING POWER SUPPLY')
538     else:
539         print('TOP/BOTTOM POWER SUPPLIES (VDD AND GND TOGETHER)')
540     print('X size =', tex, 'Âţm')
541     print('Y size =', tey, 'Âţm')
542     print('Nb of columns =', nC)
543     print('Nb of lines =', nL)
544     print('Depth of the IC (FIXED VALUE) =', int(nH * 10), 'Âţm')
545     print('Probe connected =', c_probe)
546     if c_probe:
547         print('!CENTRAL PROBE, square 30 Âţm * 30 Âţm, not editable! (for now at least)')
548     print('Mixed TW and DW cells =', mixedDWTW)
549     if not mixedDWTW:
550         if TWorDW:
551             print('Exclusive TRIPLE-WELL')
552         else:
553             print('Exclusive DUAL-WELL')
554
555     # ============================================================================
556     # BEGINNING OF THE SCRIPT
557     # ============================================================================
558     filename = "./GENERATED_NETLISTS/"
559     if not c_probe:
560         if mixedDWTW:
561             filename += f'./bbiGenVer{VINT}_T{int(tSub)}eT{eT}x'
562             filename += f'{int(tex)}y{int(tey)}mixDwTwOp{args.appendFname}.sp'
563         else:
564             if TWorDW:
565                 filename += f'./bbiGenVer{VINT}_T{int(tSub)}eT{eT}x'
566                 filename += f'{int(tex)}y{int(tey)}excTwOp{args.appendFname}.sp'
567             else:
568                 filename += f'./bbiGenVer{VINT}_T{int(tSub)}eT{eT}x'
569                 filename += f'{int(tex)}y{int(tey)}excDwOp{args.appendFname}.sp'
570     else:
571         filename += f'./bbiGenVer{VINT}_T{int(tSub)}eT{eT}'
572         filename += f'x{int(tex)}y{int(tey)}Px{prbX}Py{prbY}'
573         if mixedDWTW:
574             filename += f"mixDwTwTranVp{int(vpUU)}Pw{pW}tFR{tFR}M{model}{args.appendFname}.sp"
575         else:
576             if TWorDW:
577                 filename += f"ExcTwVp{int(vpUU)}Pw{pW}tFR{tFR}M{model}{args.appendFname}.sp"
578             else:
579                 filename += f"ExcDwVp{int(vpUU)}Pw{pW}tFR{tFR}M{model}{args.appendFname}.sp"
580
581     bbi_gen = open(filename, 'w', encoding = 'utf-8')
582
583     bbi_gen.write(f'BBI GENERATOR {VER}\n')
584     bbi_gen.write('\n.param vpUU=' + str(vpUU) + '\n')
585     bbi_gen.write('\n.param RM1=' + Rm1 + ' Rmup=' + Rmup + ' Cdecp=' + Cdecp + ' Ra=' + Ra)
```

```
586     bbi_gen.write(' Cgp=' + Cgp + ' Cgn=' + Cgn + ' RP=' + Rp + ' RN=' + Rn + ' CNW=' + Cnw)
587     bbi_gen.write(' RcontactTW=' + RcontactTW + ' RcontactDW=' + RcontactDW)
588     bbi_gen.write('\n\n')
589 
590     bbi_gen.write('vBBI_source VDOWN_BBI_source GND pwl(')
591     bbi_gen.write('0 0 3n 0 ' + str(3 + tFR) + 'n vpUU ' + str(3 + pW) + 'n vpUU ')
592     bbi_gen.write(str(3 + pW + tFR) + 'n 0)\n\n')
593 
594     bbi_gen.write("Rvbbi VDOWN_BBI_source VDOWN_BBI_trans '100e9*(TIME < 3n && TIME > ")
595     bbi_gen.write(str(3 + pW + tFR) + "n) + 0.00001'\n\n")
596 
597     bbi_gen.write("Tx1 VDOWN_BBI_trans GND VDOWN_BBI GND z0=50 td=3n\n\n")
598 
599     if model == 2:
600         bbi_gen.write("* Impedance Matching Enabled\n")
601         bbi_gen.write("R50 VDOWN_BBI GND 50\n\n")
602 
603     bbi_gen.write("vBBI VDOWN_BBI VDOWN_BBI_STM32 dc=0\n\n")
604 
605     bbi_gen.write("V_ALIM VDD GND dc=1.2\n")
606     bbi_gen.write("VmeasGND GNDmeasSTM 0 dc=0\n\n")
607     if model == 0:
608         bbi_gen.write("R150_0 GNDm GNDmeasSTM 150\n")
609         bbi_gen.write("R150_1 GNDg 0 150\n\n")
610     elif model == 1 or model == 2:
611         bbi_gen.write("R150_0 GNDm GNDmeasSTM 0.01\n")
612         bbi_gen.write("R150_1 GNDg 0 0.01\n\n")
613 
614     bbi_gen.write(f'* xSize {tex} Âţm\n')
615     bbi_gen.write(f'* ySize {tey} Âţm\n')
616     bbi_gen.write(f'* tSize {tSub} Âţm\n\n')
617 
618     if mixedDWTW:
619         bbi_gen.write('* MIXED TRIPLE-WELL AND DUAL-WELL\n')
620     else:
621         if TWorDW:
622             bbi_gen.write('* EXCLUSIVE TRIPLE-WELL\n')
623         else:
624             bbi_gen.write('* EXCLUSIVE DUAL-WELL\n')
625 
626     bbi_gen.write(f"* MODEL {model}\n")
627 
628     # subElemResDepth10u = 2000
629     # subElemResOther10u =
630 
631     rho = 0.01  # Ohm*meter
632     ResVert = (rho * ((eT * 1e-6) / 2)) / (5e-6 * 5e-6)
633     ResOther = (rho * 2.5e-6) / (5e-6 * (eT * 1e-6))
634 
635     bbi_gen.write(ELEM_BLOCKS_BEGIN)
636     bbi_gen.write(f"""R1 U N001 {ResVert}
637 R2 N001 D {ResVert}
638 R3 Re N001 {ResOther}
639 R4 N001 F {ResOther}
640 R5 N001 L {ResOther}
641 R6 R N001 {ResOther}""")
642     bbi_gen.write(ELEM_BLOCKS_END)
643 
644 #     bbi_gen.write(ELEM_BLOCKS_BEGIN)
645 #     bbi_gen.write(f"""R1 U N001 {subElemResDepth10u * (eT / 10)}
```

```
# R2 N001 D {subElemResDepth10u * (eT / 10)}
# R3 Re N001 {subElemResOther10u * (eT / 10)}
# R4 N001 F {subElemResOther10u * (eT / 10)}
# R5 N001 L {subElemResOther10u * (eT / 10)}
# R6 R N001 {subElemResOther10u * (eT / 10)}""")
#     bbi_gen.write(ELEM_BLOCKS_END)

    # rho = 0.01  # Ohm*meter
    # subRT = 2750
    # subRB = 2750
    # subRL = 2750
    # subRR = 2750
    # subRRe = 2750
    # subRF = 2750

#     bbi_gen.write(ELEM_BLOCKS_BEGIN)
#     bbi_gen.write(f"""R1 U N001 {subRT}
# R2 N001 D {subRT}
# R3 Re N001 {subRT}
# R4 N001 F {subRT}
# R5 N001 L {subRT}
# R6 R N001 {subRT}""")
#     bbi_gen.write(ELEM_BLOCKS_END)

    if mixedDWTW:
        gen_status = SubstrateGeneration("mixed").get_status()
    elif not mixedDWTW and TWorDW:
        gen_status = SubstrateGeneration("TW").get_status()
    elif not mixedDWTW and not TWorDW:
        gen_status = SubstrateGeneration("DW").get_status()
    else:
        gen_status = SubstrateGeneration("error").get_status()

    if gen_status == GEN_ERROR:
        raise ValueError

    print("Main generation beginning")
    for le in range(nL):
        for co in range(nC):
            # ================================================================================
            # Generating first the local X and Y coordinates lists needed in nets names.
            # ================================================================================
            X = [co * WSEG + 0 * (WSEG6 / 2),
                 co * WSEG + 1 * (WSEG6 / 2),
                 co * WSEG + 3 * (WSEG6 / 2),
                 co * WSEG + 5 * (WSEG6 / 2),
                 co * WSEG + 7 * (WSEG6 / 2),
                 co * WSEG + 9 * (WSEG6 / 2),
                 co * WSEG + 11 * (WSEG6 / 2),
                 co * WSEG + 12 * (WSEG6 / 2)]
            X = [float(i) for i in X]  # To be sure they are all floats.
            Xf = X.copy()
            X = [str(i).replace('.', 'p') for i in X]

            Y = [le * HSEG,
                 le * HSEG + 1 / 2 * HSEG,
                 (le + 1) * HSEG]
            Y = [float(i) for i in Y]  # To be sure they are all floats.
            Yf = Y.copy()
            Y = [str(i).replace('.', 'p') for i in Y]
```

```
706
707             x_trace.append(Xf)
708             y_trace.append(Yf)
709
710             # ================================================================================
711             # VDD nets (Metal TOP and Metal 1)
712             # ================================================================================
713             vic = 'V_' + X[1] + '_' + Y[2] + '_0'
714             vic1 = 'V_' + X[6] + '_' + Y[2] + '_0'
715             vi1c = 'V_' + X[1] + '_' + Y[0] + '_0'
716             vi1c1 = 'V_' + X[6] + '_' + Y[0] + '_0'
717             vlt = 'VM_' + X[0] + '_' + Y[1] + '_0'
718             vrt = 'VM_' + X[7] + '_' + Y[1] + '_0'
719
720             # ================================================================================
721             # GND nets (Metal TOP and Metal 1)
722             # ================================================================================
723             gic = 'G_' + X[1] + '_' + Y[2] + '_0'
724             gic1 = 'G_' + X[6] + '_' + Y[2] + '_0'
725             gi1c = 'G_' + X[1] + '_' + Y[0] + '_0'
726             gi1c1 = 'G_' + X[6] + '_' + Y[0] + '_0'
727             glb = 'GM_' + X[0] + '_' + Y[1] + '_0'
728             grb = 'GM_' + X[7] + '_' + Y[1] + '_0'
729
730             # ================================================================================
731             # LEFT NETS
732             # ================================================================================
733             VL = []
734             for left in range(1, nH + 1, 1):
735                 VL.append('VLR_' + X[0] + '_' + Y[1] + '_' + str(left))
736
737             # ================================================================================
738             # RIGHT NETS
739             # ================================================================================
740             VR = []
741             for right in range(1, nH + 1, 1):
742                 VR.append('VLR_' + X[7] + '_' + Y[1] + '_' + str(right))
743
744             # ================================================================================
745             # REAR NETS
746             # Will be a 3D numpy array,
747             # ================================================================================
748             VRE = np.empty((nH, 6), dtype = object)
749             for rear in range(1, nH + 1, 1):
750                 for floc in range(6):
751                     VRE[rear - 1, floc] = 'VFRE' + str(floc + 1) + '_'
752                     VRE[rear - 1, floc] += X[floc + 1] + '_'
753                     VRE[rear - 1, floc] += Y[2] + '_' + str(rear)
754
755             # ================================================================================
756             # FRONT NETS
757             # ================================================================================
758             VFR = np.empty((nH, 6), dtype = object)
759             for front in range(1, nH + 1, 1):
760                 for floc in range(6):
761                     VFR[front - 1, floc] = 'VFRE' + str(floc + 1) + '_'
762                     VFR[front - 1, floc] += X[floc + 1] + '_'
763                     VFR[front - 1, floc] += Y[0] + '_' + str(front)
764
765             # ================================================================================
```

```
            # EXTREME BOTTOM NETS
            # ================================================================================
            VDOWN = []
            for bot in range(6):
                VDOWN.append('VDOWN_' + str(bot + 1) + '_' + X[bot + 1] + '_' + Y[1])

            # ================================================================================
            # EVALUATING AND CONNECTINF EXTERNAL POWER SUPPLY, GROUND, AND BBI PROBE.
            # ================================================================================
            if ring:
                alimplace = False
                if np.isclose(Xf[0], 0.0):  # LEFT SIDE
                    vic = 'VDD'
                    vi1c = 'VDD'
                    gic = 'GNDm'
                    gi1c = 'GNDm'
                    alimplace = True
                elif np.isclose(Xf[-1], tex):  # RIGHT SIDE
                    vic1 = 'VDD'
                    vi1c1 = 'VDD'
                    gic1 = 'GNDm'
                    gi1c1 = 'GNDm'
                    alimplace = True
                elif np.isclose(Yf[0], 0.0):  # FRONT
                    vi1c = 'VDD'
                    vi1c1 = 'VDD'
                    gi1c = 'GNDm'
                    gi1c1 = 'GNDm'
                    alimplace = True
                elif np.isclose(Yf[-1], tey):  # REAR
                    vic = 'VDD'
                    vic1 = 'VDD'
                    gic = 'GNDm'
                    gic1 = 'GNDm'
                    alimplace = True
                else:
                    pass  # CAS GÃL'NÃL'RAL
            else:
                alimplace = False
                if np.isclose(Yf[0], 0.0):  # FRONT
                    vi1c = 'VDD'
                    vi1c1 = 'VDD'
                    gi1c = 'GNDm'
                    gi1c1 = 'GNDm'
                    alimplace = True
                    # print('alimplace')
                elif np.isclose(Yf[-1], tey):  # REAR
                    vic = 'VDD'
                    vic1 = 'VDD'
                    gic = 'GNDm'
                    gic1 = 'GNDm'
                    alimplace = True
                    # print('alimplace')
                else:
                    pass  # CAS GÃL'NÃL'RAL

            probe_here = False
            # print(Xf[0], tex / 2.0)
            # print(Yf[1], (tey / 2.0) + 15.0)
            # print(Yf[1], (tey / 2.0) - 15.0)
```

```
826             # print(c_probe
827             #       and np.isclose(Xf[0], tex / 2.0)
828             #       and Yf[1] <= (tey / 2.0) + 15.0
829             #       and Yf[1] >= (tey / 2.0) - 15.0)
830             if probe_center:
831                 if (c_probe
832                    and (np.isclose(Xf[0], prbX) or np.isclose(Xf[2], prbX))
833                    and (Yf[0] <= (prbY) + 15.0)
834                    and (Yf[0] > (prbY) - 15.0)):
835                     VDOWN = ['VDOWN_BBI_STM32' for i in VDOWN]
836                     # print(Xf[0], Yf[0], 'PROBE')
837                     probe_here = True
838                     probe_count += 1
839                 else:
840                     pass
841             else:
842                 if (c_probe
843                    and (np.isclose(Xf[0], prbX) or np.isclose(Xf[2], prbX))
844                    and Yf[0] <= prbY + 15.0
845                    and Yf[0] > prbY - 15.0):
846                     VDOWN = ['VDOWN_BBI_STM32' for i in VDOWN]
847                     # print(Xf[0], Yf[0], 'PROBE')
848                     probe_here = True
849                     probe_count += 1
850                 else:
851                     pass
852
853             # ============================================================================
854             # WRITING INTO FINAL FILE
855             # ============================================================================
856             if not probe_here:
857                 if not alimplace:
858                     if mixedDWTW:
859                         dualORtriple = dual_or_triple(X, Y, mul,
860                                                       ext = ext_loc, s_x = tex, s_y = tey)
861                         if dualORtriple == 'elem_blocTW':
862                             bbi_gen.write('XblocTW_x' + X[0] + '_y' + Y[0] + ' ')
863                         elif dualORtriple == 'elem_blocDW':
864                             bbi_gen.write('XblocDW_x' + X[0] + '_y' + Y[0] + ' ')
865                     else:
866                         if TWorDW:
867                             bbi_gen.write('XblocTW_x' + X[0] + '_y' + Y[0] + ' ')
868                         else:
869                             bbi_gen.write('XblocDW_x' + X[0] + '_y' + Y[0] + ' ')
870                     # bbi_gen.write('Xbloc_x' + X[0] + '_y' + Y[0] + ' ')
871                 else:
872                     if mixedDWTW:
873                         dualORtriple = dual_or_triple(X, Y, mul,
874                                                       ext = ext_loc, s_x = tex, s_y = tey)
875                         if dualORtriple == 'elem_blocTW':
876                             bbi_gen.write('XblocTWALM_x' + X[0] + '_y' + Y[0] + ' ')
877                         elif dualORtriple == 'elem_blocDW':
878                             bbi_gen.write('XblocDWALM_x' + X[0] + '_y' + Y[0] + ' ')
879                     else:
880                         if TWorDW:
881                             bbi_gen.write('XblocTWALM_x' + X[0] + '_y' + Y[0] + ' ')
882                         else:
883                             bbi_gen.write('XblocDWALM_x' + X[0] + '_y' + Y[0] + ' ')
884                     # bbi_gen.write('Xbloc_x' + X[0] + '_y' + Y[0] + ' ')
885             else:
```

```
            # print('xblocpro')
            if not alimplace:
                if mixedDWTW:
                    dualORtriple = dual_or_triple(X, Y, mul,
                                                  ext = ext_loc, s_x = tex, s_y = tey)
                    if dualORtriple == 'elem_blocTW':
                        bbi_gen.write('XblocTWPRB_x' + X[0] + '_y' + Y[0] + ' ')
                    elif dualORtriple == 'elem_blocDW':
                        bbi_gen.write('XblocDWPRB_x' + X[0] + '_y' + Y[0] + ' ')
                else:
                    if TWorDW:
                        bbi_gen.write('XblocTWPRB_x' + X[0] + '_y' + Y[0] + ' ')
                    else:
                        bbi_gen.write('XblocDWPRB_x' + X[0] + '_y' + Y[0] + ' ')
            else:
                if mixedDWTW:
                    dualORtriple = dual_or_triple(X, Y, mul,
                                                  ext = ext_loc, s_x = tex, s_y = tey)
                    if dualORtriple == 'elem_blocTW':
                        bbi_gen.write('XblocTWALMPRB_x' + X[0] + '_y' + Y[0] + ' ')
                    elif dualORtriple == 'elem_blocDW':
                        bbi_gen.write('XblocDWALMPRB_x' + X[0] + '_y' + Y[0] + ' ')
                else:
                    if TWorDW:
                        bbi_gen.write('XblocTWALMPRB_x' + X[0] + '_y' + Y[0] + ' ')
                    else:
                        bbi_gen.write('XblocDWALMPRB_x' + X[0] + '_y' + Y[0] + ' ')

        bbi_gen.write(vic + ' ')
        bbi_gen.write(vic1 + ' ')
        bbi_gen.write(vi1c + ' ')
        bbi_gen.write(vi1c1 + ' ')

        bbi_gen.write('\n+')

        bbi_gen.write(gic + ' ')
        bbi_gen.write(gic1 + ' ')
        bbi_gen.write(gi1c + ' ')
        bbi_gen.write(gi1c1 + ' ')

        bbi_gen.write('\n+')

        bbi_gen.write(vlt + ' ')
        bbi_gen.write(vrt + ' ')
        bbi_gen.write(glb + ' ')
        bbi_gen.write(grb + ' ')

        bbi_gen.write('\n+')

        for line in VL:
            bbi_gen.write(line + ' ')

        bbi_gen.write('\n+')

        for line in VR:
            bbi_gen.write(line + ' ')

        bbi_gen.write('\n+')

        for depth in range(nH):
```

```
946                 for line in VFR[depth]:
947                     bbi_gen.write(line + ' ')
948                 bbi_gen.write('\n+')
949                 for line in VRE[depth]:
950                     bbi_gen.write(line + ' ')
951 
952             bbi_gen.write('\n+')
953 
954             for line in VDOWN:
955                 bbi_gen.write(line + ' ')
956 
957             if mixedDWTW:
958                 bbi_gen.write(dual_or_triple(X, Y, mul, ext = ext_loc, s_x = tex, s_y = tey))
959             else:
960                 if TWorDW:
961                     bbi_gen.write("elem_blocTW")
962                 else:
963                     bbi_gen.write("elem_blocDW")
964 
965             bbi_gen.write('\n\n')
966             bar.next()
967 
968     # print(f"\nPROBE COUNT = {probe_count}")
969 
970     bbi_gen.write('\n.tran ' + sim_step + ' ' + sim_time)
971     if mixedDWTW:
972         bbi_gen.write(BBI_END_V11_mixed)
973     elif TWorDW:
974         bbi_gen.write(BBI_END_V11_TW)
975     else:
976         bbi_gen.write(BBI_END_V11_DW)
977 
978     if args.plotSubInt == "True":
979         bbi_gen.write('.print v(*.VSUBC*)\n')
980         bbi_gen.write('.print v(VDOWN*)\n')
981     bbi_gen.write("\n.end")
982     bbi_gen.close()
983 
984     try:
985         if probe_count != 6:
986             raise ProbeError
987         if args.res == "True":
988             print("\033[1;32m\nGeneration ended without errors.")
989             print("Drawing resulting netlist top-view, please wait...\033[1;0m")
990             bbi_gtest = open(filename, 'r')
991             bbi_results = pgen.plotGEN(bbi_gtest.read())
992             bbi_gtest.close()
993             if bbi_results:
994                 pass
995             else:
996                 print("Plot error\033[1;0m")
997         else:
998             print("\033[1;32m\nGeneration ended without errors, exiting generator.\033[1;0m")
999     except ProbeError:
1000         os.remove(filename)
1001         print("\033[1;31mProbe incorrectly placed.")
1002         print("Please choose another IC size.\033[1;0m")
1003 
1004     if args.compress == "True":
1005         print("\033[1;33mCompressing text file, please wait...\033[1;0m")
```

```
        with open(filename, 'r') as bbi_gen, gzip.open(f'{filename}.gz', 'wt') as bbi_gen_gz:
            bbi_gen_gz.write(bbi_gen.read())
        print("\033[1;34mFile compressed successfully\033[1;0m")
        print("\033[1;35mRemoving uncompressed .sp file!\033[1;0m")
        os.remove(filename)


if __name__ == '__main__':
    startTime = time.time()
    main()
    endTime = time.time()
    print(f"Elasped time for execution: {round(endTime - startTime, 2)} s")
```

Listing 1: *SCS generation algorithm Python implementation*

Bibliography

[1] Colin O'Flynn. Low-cost body biasing injection (BBI) attacks on WLCSP devices. In Pierre-Yvan Liardet and Nele Mentens, editors, *Smart Card Research and Advanced Applications*, pages 166–180, Cham, 2021. Springer International Publishing. ix, 9, 13, 14, 48

[2] M. Lisart M. Dumont and P. Maurine. Modeling and simulating electromagnetic fault injection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(4):680–693, 2021. ix, 8, 36, 37, 40, 42

[3] G. Chancel, J.-M. Galliere, and P. Maurine. Body biasing injection: To thin or not to thin the substrate? In Josep Balasch and Colin O'Flynn, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 125–139, Cham, 2022. Springer International Publishing. 36

[4] Prasanna Ravi, Zakaria Najm, Shivam Bhasin, Mustafa Khairallah, Sourav Sen Gupta, and Anupam Chattopadhyay. Security is an architectural design constraint. *Microprocessors and Microsystems*, 68:17–27, 2019. 2

[5] C. Sanchez-Avila and R. Sanchez-Reillol. The rijndael block cipher (aes proposal) : a comparison with des. In *Proceedings IEEE 35th Annual 2001 International Carnahan Conference on Security Technology (Cat. No.01CH37186)*, pages 229–234, 2001. 3, 29

[6] National Institute of Standards and Technology. Fips-46: Data encryption standard (des). revised as fips 46-1:1988, fips 46-2:1993, fips 46-3:1999. 3

[7] How-Shen Chang. International data encryption algorithm. *jmu. edu, googleusercontent. com, Fall*, 2004. 3

[8] D. Coppersmith, D. B. Johnson, and S. M. Matyas. A proposed mode for triple-des encryption. *IBM Journal of Research and Development*, 40(2):253–262, 1996. 3

[9] A. Shamir R.L. Rivest and L.Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Communications of the ACM*, volume 21, pages 120–126, 1978. 3, 5

[10] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985. 3

[11] Vivek Kapoor, Vivek Sonny Abraham, and Ramesh Singh. Elliptic curve cryptography. *Ubiquity*, 2008(May):1–8, 2008. 3

[12] Ronald Cramer and Victor Shoup. Signature schemes based on the strong rsa assumption. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):161–185, 2000. 3

[13] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, pages 104–113, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. 5

[14] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1:36–63, 2001. 5

[15] Boris KÃűpf and Markus DÃijrmuth. A provably secure and efficient countermeasure against timing attacks. In *2009 22nd IEEE Computer Security Foundations Symposium*, pages 324–335, 2009. 5

[16] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. 5

[17] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 142–159. Springer, 2013. 6

[18] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 16–29, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. 6

[19] Vincent Carlier, HervÃl' Chabanne, Emmanuelle Dottax, and HervÃl' Pelletier. Electromagnetic side channels of an fpga implementation of aes. Cryptology ePrint Archive, Paper 2004/145, 2004. https://eprint.iacr.org/2004/145. 6

[20] Thomas Ordas, Mathieu Lisart, Etienne Sicard, Philippe Maurine, and Lionel Torres. Near-field mapping system to scan in time domain the magnetic emissions of integrated circuits. In Lars Svensson and José Monteiro, editors, *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, pages 229–236, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. 6

[21] Aurélien Vasselle, Philippe Maurine, and Maxime Cozzi. Breaking mobile firmware encryption through near-field side-channel analysis. In *Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, ASHES'19, page 23âĂŞ32, New York, NY, USA, 2019. Association for Computing Machinery. 6

[22] Robert Schilling, Mario Werner, and Stefan Mangard. Securing conditional branches in the presence of fault attacks. In *2018 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1586–1591, 2018. 7

[23] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Annual International Cryptology Conference*, 1997. 7

[24] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 37–51, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. 7

[25] Mathieu Ciet and Marc Joye. Elliptic curve cryptosystems in the presence of permanent and transient faults. *Designs, Codes and Cryptography*, 36(1):33–43, July 2005. 7

[26] Ingrid Biehl, Bernd Meyer, and Volker Müller. Differential fault attacks on elliptic curve cryptosystems. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, pages 131–146, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. 7

[27] Christophe Giraud. Dfa on aes. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard – AES*, pages 27–41, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. 7, 28, 29

[28] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 320–334, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. 7

[29] Sergei Skorobogatov and Ross Anderson. Optical fault induction attacks. volume 2523, pages 2–12, 08 2002. 8

[30] David Samyde, Sergei P. Skorobogatov, Ross J. Anderson, and Jean-Jacques Quisquater. On a new way to read data from memory. *First International IEEE Security in Storage Workshop, 2002. Proceedings.*, pages 65–69, 2002. 8

[31] S. Ordas, L. Guillaume-Sage, and Philippe Maurine. Em injection: Fault model and locality. In *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 3–13, 2015. 9

[32] Philippe Maurine, Karim Tobich, Thomas Ordas, and Pierre-Yvan Liardet. Yet another fault injection technique : by forward body biasing injection. 09 2012. 9, 13

[33] K. Tobich, P. Maurine, P.-Y. Liardet, M. Lisart, and T. Ordas. Voltage spikes on the substrate to obtain timing faults. In *2013 Euromicro Conference on Digital System Design*, pages 483–486, 2013. 9, 13

[34] Noemie Beringuier-Boher, Marc Lacruche, David El-Baze, Jean-Max Dutertre, Jean-Baptiste Rigaud, and Philippe Maurine. Body biasing injection attacks in practice. In *Proceedings of the Third Workshop on Cryptography and Security in Computing Systems*, CS2 '16, page 49âĂŞ54, New York, NY, USA, 2016. Association for Computing Machinery. 9, 13

[35] G. Chancel, Jean-Marc GalliÃĺre, and P. Maurine. Body biasing injection: Impact of substrate types on the induced disturbancesÆŠ. In *2022 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*, pages 50–60, 2022. 12, 36

[36] Colin O'Flynn. Picoemp: A low-cost emfi platform compared to bbi and voltage fault injection using tdc and external vcc measurements. Cryptology ePrint Archive, Paper 2023/1195, 2023. https://eprint.iacr.org/2023/1195. 17

[37] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against spn structures, with application to the aes and khazad. In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, pages 77–88, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 28

[38] Mathieu Dumont, Philippe Maurine, and Mathieu Lisart. Modeling of electromagnetic fault injection. In *2019 12th International Workshop on the Electromagnetic Compatibility of Integrated Circuits (EMC Compo)*, pages 246–248, 2019. 36

[39] Raphael A. Camponogara Viera, Philippe Maurine, Jean-Max Dutertre, and Rodrigo Possamai Bastos. Simulation and experimental demonstration of the importance of ir-drops during laser fault injection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(6):1231–1244, 2020. 36

[40] Takuya Wadatsumi, Kohei Kawai, Rikuu Hasegawa, Takuji Miki, Makoto Nagata, Kikuo Muramatsu, Hiromu Hasegawa, Takuya Sawada, Takahito Fukushima, and Hisashi Kondo. Voltage surges by backside esd impacts on ic chip in flip chip packaging. In *2022 IEEE International Reliability Physics Symposium (IRPS)*, pages P14–1–P14–6, 2022. 48

[41] Takuya Wadatsumi, Kohei Kawai, Rikuu Hasegawa, Kazuki Monta, Takuji Miki, and Makoto Nagata. Characterization of backside esd impacts on integrated circuits. In *2023 IEEE International Reliability Physics Symposium 2IRPS)*, pages 1–6, 2023. 48

[42] Jean-Max Dutertre, Vincent Beroulle, Philippe Candelier, Stephan De Castro, Louis-Barthelemy Faber, Marie-Lise Flottes, Philippe Gendrier, David HÃl'ly, Regis Leveugle, Paolo Maistri, Giorgio Di Natale, Athanasios Papadimitriou, and Bruno Rouzeyre. Laser fault injection at the cmos 28 nm technology node: an analysis of the fault model. In *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 1–6, 2018. 60

[43] Breier et al. Extensive laser fault injection profiling of 65 nm fpga. *J Hardw Syst Secur 1*, pages 237–251, 2017. 62, 69

[44] Jakub Breier and Chien-Ning Chen. On determining optimal parameters for testing devices against laser fault attacks. In *2016 International Symposium on Integrated Circuits (ISIC)*, pages 1–4, 2016. 62

[45] C. Boit, R. Schlangen, A. Glowacki, U. Kindereit, T. Kiyan, U. Kerst, T. Lundquist, S. Kasapi, and H. Suzuki. Physical ic debug and - backside approach and nanoscale challenge. *Advances in Radio Science*, 6:265–272, 2008. 66