Novel Fault Injection Attacks on Logic Locking using ATPG

by

Ayush Jain

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 8, 2020

Keywords: Logic locking, IP Piracy, IC Overproduction, Differential Fault Analysis, Fault Injection, Automatic Test pattern Generation (ATPG)

Approved by

Ujjwal Guin, Chair, Assistant Professor of Electrical and Computer Engineering
Adit D. Singh, James B. Davis Professor of Electrical and Computer Engineering
Mehdi Sadi, Assistant Professor of Electrical and Computer Engineering

Abstract

The outsourcing of the design and manufacturing of integrated circuits (ICs) in the current horizontal semiconductor integration flow includes untrusted entities that have posed various security threats, such as overproduction of ICs, sale of out-of-specification/rejected ICs, and piracy of Intellectual Properties (IPs). Logic Locking is a well-accepted protection technique against the aforementioned threats, where the original design is modified by incorporating additional key gates in the netlist, resulting in a key-dependent functional circuit. The original functionality of the chip is recovered once it is programmed with the secret key, otherwise, it produces incorrect results for some input patterns. Over the past decade, different attacks have been proposed to break logic locking, simultaneously motivating researchers to develop more secure countermeasures. This thesis presents novel fault injection attacks based on stuck-at fault analysis, which can be used to break a secure logic locking technique. The proposed attacks are based on self-referencing, where the secret key is determined by injecting faults in the key lines to perform either differential fault analysis (DFA) with its fault-free counterpart or direct key extraction at the primary output through sensitization. A commercial ATPG tool is used to generate test patterns that detect stuck-at faults on the key lines, which will be used to determine the secret key from the external fault-induced functional IC. One test pattern is sufficient to determine one key bit, which results in at most $|\boldsymbol{K}|$ test patterns to determine the entire secret key of size $|\boldsymbol{K}|$. However, The number of test patterns decreases when stuck-at faults on different key wires can be targeted simultaneously. The laser fault injection tool is used during the experimentation to induce external faults on the circuit implemented in the FPGA to demonstrate the effectiveness of the attack methodology. The proposed attack is generic to break any logic locked circuits.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Prof. Ujjwal Guin for his support and guidance during my graduate study at the Auburn University. His support and encouragement paved the way for my successful research projects and completion of the thesis.

I would like to thank Prof. Navid Asadi Zanjani, University of Florida for helping with his insight and expertise in fault injection. Also, a special thanks to Mir Tanjidur Rahman (Ph.D candidate) at University of Florida for his significant contributions towards this research work.

I want to extend my gratitude to the committee members of the thesis: Prof. Adit Singh and Prof. Mehdi Sadi for their time, support, and advice towards my research and thesis.

My sincere thanks to all my labmates and colleagues for the knowledgeable experience that I gained during my course and research work. The lab activities and brainstorming sessions provided mw with valuable insights about my field of study.

In the end, I would like to express my gratitude to my parents and friends for their consistent support throughout my academic endeavours.

Table of Contents

List of Figures

List of Abbreviations

ASIC Application Specific Integrated Circuit

ATPG Automatic Test Pattern Generation

DFA Differential Fault Analysis

FPGA Field Programmable Gate Array

IP Intellectual Property

LFI Laser Fault Injection

SAF Stuck-at Fault

SoC System on-Chip

UART Universal Asynchronous Receiver/Transmitter

Chapter 1

Introduction

Over the last few decades, the impact of globalization has transformed the semiconductor manufacturing and testing industry from vertical to horizontal integration. The continuous trend of device scaling has enabled the designer to incorporate more functionality in a system-on-chip (SoC) by adopting lower technology nodes to increase performance and reduce the overall area and cost of an SoC. At present, majority of the SoC design companies or design houses no longer manufacture chips and maintain a foundry (fab) of their own due to cost for building and maintaining such foundries [2] and the increased complexity in the fabrication process as new technology is adopted. As a result, the semiconductor industry has moved towards horizontal integration, where an SoC designer acquires intellectual properties (IPs) from many different vendors and sends the design to a foundry for manufacturing, which is generally located offshore.

In order to reduce cost and development time for the integrated circuit (IC), the most effective way is the reuse of designs or intellectual property (IP). The development and verification of these IPs require time and effort but copying or modifying the IP for illegal re-distribution or re-use leads to security risk and economic loss. Intellectual property (IP) infringement has emerged as a serious threat where restricting an adversary to gain the design information has become very difficult. The threat of IP piracy or theft arises mainly from physical reverse engineering of the IC or the outsourcing of the design for manufacturing. The layer-by-layer reverse engineering of the IC by an untrusted entity can lead to the extraction of the gate-level netlist exposing the complete or targeted part of the design. Also, vulnerability arises when the design is outsourced to a potential untrusted foundry for fabrication. In this case, the layout is possessed by the untrusted foundry which can

create the IC netlist that can be overproduced leading to IP infringement. In general, this leaves the designer with no control over the possibility of IP piracy carried out by different untrusted entities.

The hardware layers that were assumed to be trusted, are no longer true with the outsourcing of IC fabrication in a globalized and distributed design flow including multiple entities. Third-party IPs, fabrication, and test facilities of chips represent security threats to the current horizontal integration of the production. The security threats posed by these entities include – (*i*) overproduction of ICs [3, 4, 5, 6, 7, 8, 9], where an untrusted foundry fabricates more chips without the consent of the SoC designer in order to generate revenue by selling them in the market, (*ii*) sale of out-of-specification/rejected ICs [9, 10], and (*iii*) IP piracy [11, 12, 13, 14], where an entity in the supply chain can use, modify and/or sell functional IPs illegally, (*iv*) Recycling of ICs [15, 16, 13], where an used/old chip is sold in the market as new and (*v*) tampering with a hardware Trojan [17, 5, 12, 18]. Over the years, researchers have proposed different techniques to prevent the aforementioned attacks and they are IC metering [3, 4, 6, 19], logic locking [3, 20, 9], hardware watermarking [21, 22, 23], recycled IC [24, 25, 26, 27] and hardware Trojan detection [28, 29, 30, 31, 32, 33].

1.1 Motivation

Logic locking has emerged as the most prominent method to address the threats incurred from untrusted manufacturing. In logic locking, the design of a circuit is locked so that the circuit produces incorrect results in normal operation unless a correct secret key is programmed into the chip. Figure 1.1 shows an abstract view of logic locking where the key is stored in a tamper-proof non-volatile memory. Subramanyan et al. [34] first showed that a locked circuit can efficiently be broken using key-pruning oracle-guided Boolean Satisfiability (SAT) analysis. Since then, many different versions of SAT-based attacks have been launched on logic locking [35], and the solutions have been proposed to mitigate these attacks as well [36, 37, 9, 10, 38, 39, 40, 41, 42]. *Can we safely state that a logic locking technique*

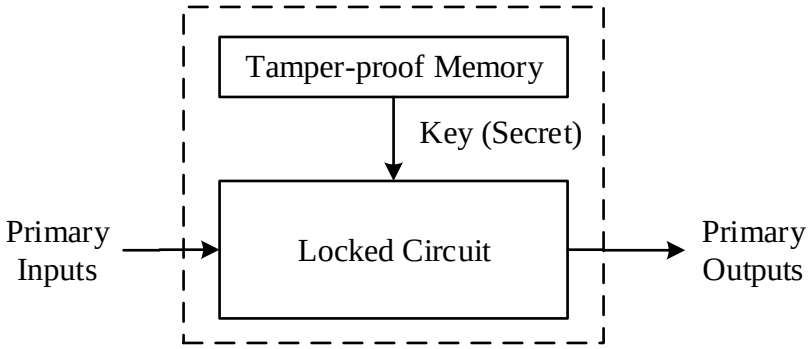

Figure 1.1: An abstract view of the logic locking technique.

is completely secure even if we achieve complete SAT resistivity? Note that an untrusted foundry has many more effective means to determine the secret key without performing SAT analysis [43, 44, 45, 46].

An adversary such as an untrusted foundry who has access to most advanced equipment, such as a micro-probing station, scanning electron microscope (SEM), etc. is capable of attacking any chip with physical attacks. The concept of fault injection attacks have been studied extensively on the security of cryptographic primitives such as AES, RSA, etc. [47, 48, 49, 50, 51, 52]. The attack aims towards intentionally disturbing the computation of the crypto-systems through invasive or non-invasive techniques, in order to extract information regarding the secret key from the erroneous output. The security of logic locking has never been exposed to such attacks before in the literature. However, it is important to study the security of a SoC design against all the existing attack to categorize it as a robust countermeasure against all the threats in the IC supply chain. With this motivation and discussion, we present the vulnerability of logic locking techniques against the fault injection attacks to undermine its security by extracting the secret key.

1.2 Contributions

In this thesis, we show how an adversary can extract the secret key from a locked netlist, even if all the existing countermeasures are in place. We propose novel attacks to break any

key-based logic locking technique using the fault injection attack. The primary contributions of this thesis are as follows:

- Firstly, we demonstrate how an adversary can perform differential fault analysis using the functional ICs (programmed with correct key). The differential fault analysis (DFA) attack on logic locking is motivated by the test pattern generation for VLSI circuits. For DFA, an input pattern that produces incorrect response for only one key bit while keeping the other key bits at the faulty states is required. To generate such a test pattern, we propose to use constrained automatic test pattern generation (ATPG) algorithm, which is widely popular for testing of VLSI circuits. When we apply a constrained *sa1* pattern to a key line, the hypothesis key bit becomes 1 if the responses of the fault-free and faulty circuits are the same, otherwise, the key value is 0 [53]. The proposed attack is self-referencing and does not require any complex analysis (i.e., SAT).

- We propose a novel fault injection based key sensitization attack. It requires a test pattern that sensitizes the key bit to the primary output irrespective of other keys in the circuit. However, it is highly likely to observe the inter-dependency of the key gates in locking techniques to mask/block the key propagation. We formulate the fault injection approach to assist in key propagation and it also limits the overall number of constraints and corresponding external faults required. It is an iterative method where we perform logic cone analysis through our proposed algorithm on the netlist to determine the constraints required through the identification of key dependency. The key bits lying in the same logic cone and blocking the propagation of targeted key bits are induced with an external fault on the functional IC. When we apply the ATPG generated test pattern to detect stuck-at fault (saf) on the key lines to the fault induced functional chip, the key information is observed at the output.

- We demonstrate and validate our proposed attack by performing the laser fault injection on a Kintex-7 FPGA. The technology-dependent gate-level netlist for locked ISCAS'85 [54, 55] is generated from Synopsys Design Compiler [56], which is used in Synopsys Tetramax [57] for test pattern generation. The netlist is then converted to a technology-independent netlist and implemented in Xilinx Vivado without any optimization so that the *saf* patterns can be applied to the FPGA.

1.3 Organization of the thesis

The rest of the thesis is organized as follows: an overview of different logic locking techniques and existing attacks along with fault injection techniques is provided in Section 2. The proposed attack based on differential fault analysis and its methodology to extract the secret key from any locked circuit are described in Section 3. Next, the proposed key sensitization attack using fault injection is presented in section 4. We present the results for the implementation of the proposed attack on different logic locked benchmark circuits in section 5. Finally, we provide the future research directions and conclude the thesis in Section 6.

Chapter 2

Background and Related work

In this chapter, we discuss a few fundamentals that are essential for understanding the core concept of this thesis. Then, we provide a comprehensive survey of the relevant contributions across the research community.

2.1 Logic Locking

The challenges for protecting a circuit against hardware security threats have been the driving force for the development of different techniques to limit the amount of circuit information that can be recovered by an adversary. Logic locking has emerged as a field of significant interest from the researchers, as it can provide complete protection against IC overproduction and IP piracy. Different researchers have proposed to use logic locking to prevent hardware Trojans as well [58, 59, 60, 61, 62, 63].

The objective of logic locking is to obfuscate the inner details of the circuit and making it infeasible for an adversary to reconstruct the original netlist. Logic Locking hides the functionality of the circuit by inserting additional logic gates into the original design, which we termed as *key gates*. In addition to the original inputs, the locked circuit needs secret key inputs to key gates from on-chip tamper-proof memory (TM) (see Figure 2.1 for details). The correct functionality of the design is obtained when the key inputs receive the proper secret key value. Applying an invalid key to the key gates would result in incorrect functionality of the locked design that will produce incorrect output responses for the circuit. Note that for a secure locked circuit, the design details cannot be recovered using reverse engineering.

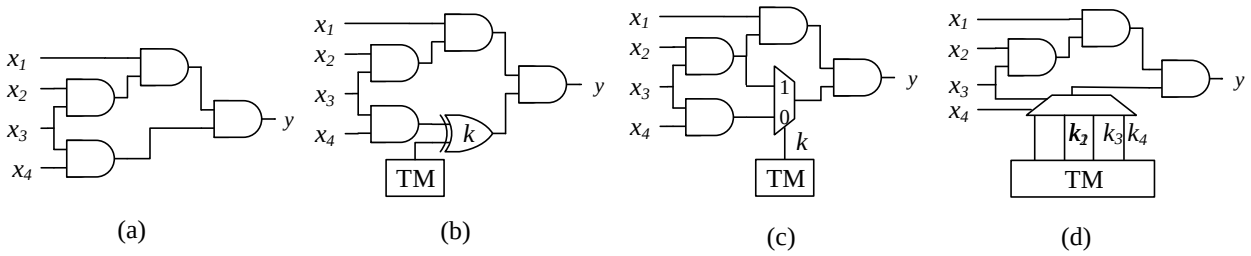

Figure 2.1: Different logic locking techniques, where the secret key (K) is programmed in a tamper-proof memory (TM). (a) Original circuit. (b) XOR-based locking. (c) MUX-based locking. (d) LUT-based locking.

2.2 Relevant Work

Different logic locking methods were devised over the years and can be categorized into three different categories. First, XOR-based logic locking, shown in Figure 2.1.(b), has received much attention due to its simplicity. In this technique, a set of XOR or XNOR gates are inserted as key gates [3, 20, 9, 64, 65, 10, 66, 38, 67]. The secret key is stored in tamper-proof memory (TM), and connections are made from TM to the key gates. Second, in the MUX-based logic locking technique [68, 69], multiplexers (MUX) are inserted so that one of its input is correct, which is the original net of the circuit. The other input of the MUX is incorrect, which is a dummy net randomly selected from the netlist. This technique is shown in Figure 2.1.(c). The select signal of the MUX is associated with the key bit from the tamper-proof memory. The correct signal goes through the MUX upon applying valid key value, otherwise incorrect signal propagates in the netlist. Third, in LUT-based logic locking, [8, 70, 60], shown in Figure 2.1.(d), a look-up table with several key inputs is used to lock the netlist. The LUTs replace a combinational logic in the design making it difficult to predict the output as it depends on several different key values

The research community has proposed several attacks to exploit the security vulnerability on a logic locked circuit. Subramanyan et al. [34] first showed that a locked circuit can be broken using Boolean Satisfiability (SAT) analysis. The SAT attack algorithm, attributed as an oracle-guided attack, requires a locked netlist, which can be recovered using reverse engineering and functional chip with a valid key stored/programmed in its tamper-proof memory.

In this attack, an adversary can query an activated chip and observe the response. Note that the SAT attack requires access to the internal nodes of the circuit through the scan chains, which is common in today's netlist for implementing Design-for-Testability (DFT) [71]. The SAT attack works iteratively to eliminate incorrect key values from the key space using distinguishing input patters (DIPs). A DIP is defined as an input pattern for which two sets of hypothesis keys produce complementary results. By comparing these with the output of an unlocked chip, one set of hypothesis keys is discarded. The SAT attack works efficiently as it discards multiple hypothesis keys in one iteration. Thereafter, SAT resiliency became an important security metric to demonstrate the security of logic locking techniques.

As a result, researchers have focused on improving and developing locking techniques to be resilient against the SAT attack. Subsequent work in this direction suggested to includ SAT resilient blocks into the design itself such as Anti-SAT [64, 72], SARLock [65] or perform cyclic obfuscation [73]. These proposed techniques use one-point functions that corrupts internal node value in the original netlist to produce incorrect outputs when supplied with a wrong key value. However, the traditional SAT resiliency of such techniques could not prevent them against other attacks that included removal/bypass attack [74], double DIP attack [75], CLIC-A [76] attack, Approximate SAT attack [77], and Cyclic SAT [78].

Due to limitations of SARlock, Yasin et al. developed an improved version of this design and referred to as TTLock [66], where the original design itself is modified to produce corrupted/inverted results upon applying an incorrect key for a single input pattern. To provide more flexibility in the number of protected input patterns, stripped functionality based logic locking (SFLL) [67] was proposed. The design is no longer the same as the original design due to stripped parts of the functionality resulting in erroneous output. A separate restore unit is responsible for removing this error in the output supplying correct key values to it. However, Subramanyan et al. has shown recently that SFLL can be defeated through FALL attack [79]. The attack is built on 3 primary steps, namely, structural analysis, functional analysis, and key confirmation. The structural analysis is performed to identify the gates

that are the output of the cube stripping function in SFLL. After identification of these candidate gates, the functional analysis targets the property of cube stripping functions, which results in a set of potential key values. Finally, the key confirmation algorithm identifies the correct key from the set of potential key values.

Currently, secure solutions from the logic design point of view include modifications in the original circuit [37, 42, 80, 41]. The SFLL-rem is another variant of the parent stripped functionality based logic locking technique which relies on performing functionality stripping through the injected stuck-at fault [37]. A fault is injected in the netlist and the circuit logic gates stripped corresponding to the injected fault. All the failing patterns are now reported based on the comparison with the original netlist. Amongst all the patterns, a single pattern is chosen for which the circuit will produce erroneous output, while the circuit is reconstructed for the remaining failing patterns through the ECO tool. This result in the stripped circuit which produces erroneous output for particular input pattern. To correct the output for this pattern, a restore unit is implemented which is basically a comparator circuit. The same pattern is also stored as the key. Once the same pattern appears at the input and it matches with the key, the output of the comparator becomes logic 1 which XOR's with the erroneous output to correct it. This SFLL-rem method has been demonstrated resilient to SAT as well as FALL attacks.

As the SAT-attack is based on the availability of accessing the internal states of a circuit through the scan chains, Guin et al. proposed placing multiple flip-flops capturing signals controlled by different key bits at the same level of the parallel scan chains, which were used in current test compression methodologies [9]. However, a vulnerability existed in this design, when an adversary performs multi-cycle tests, such as delay tests (transition delay faults and path delay faults) [71]. This leads to the necessity for developing a new design-for-security (DFS) architecture to prevent leaking of the key during any manufacturing tests [10, 38]. This design prevents scanning out the internal states of design after a chip is being activated and the keys are programmed/stored in the circuit. Other research groups have also demonstrated

resiliency against the SAT attack by restricting the scan-access and preventing key leakage through scan-chain by performing sequential locking [81, 82, 83]. Rahman et al. proposed the dynamic obfuscation of the scan-chain through the presence of LFSR circuit and shadow chain structure. It was predominantly shown resilient to SAT attack, however, it was shown vulnerable to other attack [84, 85]. Sequential locking techniques are mainly demonstrated vulnerable to SAT/model-checker based attacks [86, 87].

Additionally, an adversary such as an untrusted foundry who has access to most advanced equipment is also capable of attacking any chip with physical attacks [88]. Apart from SAT-based attacks, probing attacks [89, 90] have also shown serious threats to the security of logic locking, where an attacker makes contact with the probes at signal wires in order to extract sensitive information, mainly, the secret key. With the help of a focused ion beam (FIB), a powerful circuit editing tool that can mill and deposit material with nanoscale precision, an attacker can circumvent protection mechanisms and reach wires carrying sensitive information. However, the countermeasures reflect the complexity of shield-structure and nanopyramid structures as the defense, making it difficult to perform these attacks [91, 92].

Another prominent physical attack includes tampering the netlist with a hardware Trojan to leak the secret key to the primary output. Jain et al. demonstrated how an untrusted foundry can tamper the netlist to insert the hardware Trojan, either combinational or sequential hardware Trojan that evades manufacturing or production test (e.g. stuck-at fault, etc.) [44, 29]. Under normal condition, the circuits functions correctly. However, upon activation of the Trojan through the primary inputs, the payload is delivered to leak the secret key directly to the primary output through the multiplexer (MUX). Moreover, the design of sequential Trojan requires satisfying the trigger condition R times to activate the Trojan. Only the adversary has the knowledge regarding the maximum counter value (R) making it very difficult for detection as it highly unlikely that same input pattern appear R times consecutively during the normal functioning of the circuit. Recently, Zhang et al. proposed an oracle less attack to extract the key from locked circuits [45, 93]. The notion of this

attack is to compare the locked and unlocked instances of repeated Boolean functions in the netlist to predict the key. A solution was also proposed to countermeasure the attack as well. Despite many solutions proposed over the years, none of the logic locking techniques can be categorized to provide complete security.

2.3 Fault Injection Methods

Over the years, several threats and methods have emerged to break a cryptosystem without performing either mathematical analysis or brute force attacks. Using these attacks, an adversary can subvert the security of protection schemes, primarily through extracting or estimating the secret key using physical attacks. Fault injection attacks, intentionally disturb the computation of cryptosystems in order to induce errors at the output response. To achieve this, external fault injection is performed through invasive or non-invasive techniques. This is followed by the exploitation of erroneous output to extract information from the device.

Fault based analysis on cryptosystems was first presented theoretically by Boneh et al. on RSA [94]. This contribution initiated a new research direction to study the effect of fault attacks on cryptographic devices. The comparison between the correct and faulty encryption results has been demonstrated as an effective attack to obtain information regarding the secret key [95, 96]. These can be realized into different categories:

• *Clock Glitch*: The devices under attack are supplied with an altered clock signal which contains a shorter clock pulse than the normal operating clock pulse. For successfully inducing a fault, these clock glitches applied are much shorter than the tolerable variation limit in clock pulse for the circuit. This results in setup time violations in the circuit and skipping of instructions from the correct order of execution [97, 98].

• *Power Variation*: This technique can be further bifurcated into two subcategories: either the malicious entity may choose to provide low power supply to the system (also abbreviated

as underfeeding) or the adversary may choose to influence the power line with spikes. This adversely affects the set-up time and influences the normal execution of operations. The state elements in the circuit are triggered without the input reaching any stable value, causing a state transition to skip operations or altering the sequence of execution [99, 100, 101].

• *Electromagnetic Pulses/Radiation*: The eddy current generated by an active coil can be used to precisely inject faults at specific location in the chip. This method doesn't require the chip to be decapsulated in order to inject the fault. However, the adversary is required to possess information regarding specific modules and their location inside the chip [102, 103].

• *Laser*: Fault injection using laser is also regarded as a very efficient method because it can precisely induce a fault at an individual register to change its value [104]. For optical fault injection, the laser can be focused at a specific region of the chip from the backside or front side. However, due to metal layers on the front side, it is preferred to perform the attack on the backside of the chip. Skorobogatov et al. [105] first demonstrated the effectiveness of this method by using a flashgun to inject fault to flip a bit in SRAM cell. Several other research groups also utilized and proposed different variants of this method to study the security of cryptographic primitives [47, 48, 49, 50].

• *Focused-ion Beam (FIB)*: The most effective and expensive fault injection technique is devised with focused ion beam (FIB) [106]. Technological advancement of this method enables it to cut/connect wires and even operate through various layers of the IC fabricated in the latest technology nodes [107].

• *Software Implemented Fault Injection*: This technique produces errors through software that would have been produced when a fault targeted the hardware. It involves the modification of the program running on the target system to provide the ability to perform the fault injection. It does not require dedicated complex hardware, gate-level netlist or RTL models that are described in hardware description languages. The faults are injected in

accessible memory cells such as registers and memories through software that represent the most sensitive zones of the chip [108, 109, 110].

In this chapter, we introduced the fundamental concept of logic locking and surveyed relevant literature that are essential for understanding the work presented in this thesis. We conclude that the existing methods mainly focus on SAT-resiliency as an important security metric to define the strength of any logic locking scheme. Additionally, different fault injection techniques are studied to understand how an adversary can attack the design without performing SAT analysis.

Chapter 3

ATPG Guided Differential Fault Analysis Methodology

The general hardware security strategy adopted for designing and manufacturing a circuit involves a logic locking technique, where a chip is locked by storing a secret key in the tamper-proof memory. As this secret key is the same for all the chips manufactured with the same design, finding this key from one chip undermines the security resulted from logic locking. An adversary can determine the secret key by injecting faults at the key registers [111, 43], which hold the key value during normal operation, and performing differential fault analysis (DFA). With this motivation, we extend the concept of fault injection attacks on logic locking to study the vulnerability of locking techniques and undermine its security.

3.1 Threat Model

The threat model defines the traits of an adversary and its position in the IC manufacturing and supply chain. It is very important to know an attacker's capabilities and its resources/tools to estimate its potential to launch the attack. The design house or entity designing the chip is assumed to be trusted. The attacker is assumed to be an untrusted foundry or a reverse engineer having access to the following:

- The attacker has access to the locked netlist of a circuit. An untrusted foundry has access to all the layout information which can be extracted from the GDSII or OASIS file. Also, a locked netlist can be constructed from layer-by-layer reverse engineering of the fabricated chip with advanced technological tools [106]. The attacker has the capability to determine the location of the tamper-proof memory. It can be trivial for an adversary to find the location of the key register in a netlist, as it can easily trace the route from the tamper-proof memory.

- The attacker has possession of an unlocked and fully functional chip, which can be easily acquired from the market.
- A fault injection equipment is necessary to launch the attack. It is not necessary to use high-end fault injection equipment. The basic requirement is to inject faults at the key registers (all the flip-flops) location on a de-packaged/packaged chip.

Notations: An original circuit, and its locked version are denoted by C_O and C_L, respectively. The two versions of fault-injected C_L are represented as C_F and C_A. C_F represents a locked circuit where all the key lines ($|K|$) are injected with logic 1 (logic 0) faults, denoted as a faulty circuit. C_A represents the same locked circuit where ($|K| - 1$) key lines are injected with the same logic 1 (logic 0) faults, leaving one key line fault free. We denote this circuit as a fault-free circuit for DFA. Both functional chips are loaded with the correct key in its tamper-proof memory. A fault is injected at the key register using a fault injection method (see details in Chapter 5). For any given circuit, we assume the primary inputs (PI) of size $|PI|$, primary outputs (PO) of size $|PO|$, and secret key (K) size of $|K|$. We also use key lines or key registers alternatively throughout this thesis as their effects are the same on a circuit. Note that *saf* is an abstract representation of a defect to generate test patterns, whereas, an injected fault is the manifestation of a faulty logic state due to fault injection.

3.2 Differential Fault Analysis Attack Methodology

The proposed fault injection attack relies on differential fault analysis, where the responses of two instances of faulty and fault-free circuits are compared to determine the secret key. An adversary can use any fault injection methods (see the details in Section 2.3) to create the faulty chip. However, we will not perform traditional fault injection methods to demonstrate the attack. Instead, we create a faulty circuit by partial reconfiguration, which is common in FPGA environment. Note that the primary objective of this thesis is to present a very powerful attack to break any logic locking technique.

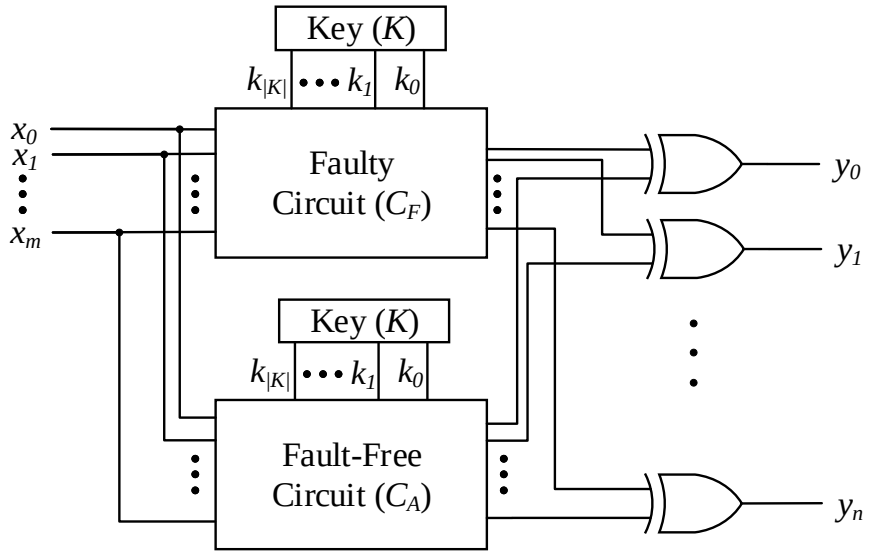

Figure 3.1: The abstract representation of the proposed ATPG guided differntial fault analysis attack.

Figure 3.1 shows an abstract representation of the proposed fault ATPG guided differential fault analysis attack. The fault-free circuit (C_A) is an unlocked chip bought from the market whose key-bits needs to be retrieved. Except the key-bit which is targeted to be extracted, all remaining key registers are fixed to a particular faulty value of either 0 or 1 corresponding to the selected fault. While, faulty circuit (C_F) is the same chip, which is injected with a particular fault to keep all the key registers or interconnects to faulty value of logic 1 or 0. One input pattern is first applied to C_A and its response is collected. The same input pattern is then applied to the C_F to collect the faulty response. The output responses are XORed to find any mismatch. If both the circuits differ in their responses, the XORed output will be 1, otherwise, it will be 0. If we find an input pattern that produces a conflicting results for both C_A and C_F only for one key bit, the key value can be predicted. The key value is same as the injected fault value if the XORed output is of logic 0, otherwise, the key value is complementary to the injected fault.

The proposed attack can be described as follows:

- *Step-1*: The first step is to select an input pattern that produces complementary results for the fault-free (C_A) and faulty (C_F) circuits. The input pattern needs to satisfy the

following property – it must sensitize only one key bit to the primary output(s). In other words, only the response of one key bit is visible at the *PI* keeping all other key bits at logic 1s (or 0s). If this property is not satisfied, it will be impractical to reach a conclusion regarding the value of a key bit. Multiple key combinations can result the same. *Now the question is how can we find if such a pattern exists in the entire input space (ξ).*

To meet this requirement, our method relies on stuck-at faults (*saf*) based constrained ATPG to obtain the specific input test patterns (see details in Section 3.3). Considering the fact that adversary has access to the locked netlist (C_L), it can generate test patterns to detect *sa1* or *sa0* at any key lines and adding constraints to other key lines (logic 1 and 0 for *sa1* and *sa0*, respectively). A single fault, either *sa0* or *sa1* on a key line is sufficient to determine the value of that key bit. Therefore, we have selected *sa1* and the following sections are explained considering this fault only. This process is iterated over all the key-bits to obtain $|\boldsymbol{K}|$ test patterns. The algorithm to generate the complete test pattern set is provided in Algorithm section 3.3.

- *Step-2*: The complete set of generated test patterns are applied to fault-induced functional circuit (C_F). The circuit is obtained by injecting logic 1 fault on the key registers if *sa1* is selected in the previous step, else, the circuit is injected with logic 0 fault for *sa0*. The responses are collected for later comparison with the fault-free responses. For (C_A), the test patterns are applied such that it matches the fault modifications in the circuit. For example, the test pattern for the first key is applied to the circuit when the circuit instance does not pertain any fault on its corresponding key register and holds the correct key value while, the remaining key registers are set to logic 1 (for *sa1*) or 0 (for *sa0*). For the next key-bit, (C_A) instance is created by excluding this selected key bit from any fault while keeping all the other key registers to logic 1 (for *sa1*) or

0 (for *sa0*). This process is repeated for all key bits and their responses are collected for comparison in the next step.

- *Step-3*: The adversary will make the decision regarding the key value from the observed differences in the output responses of (C_A) and (C_F). For any test pattern corresponding to a particular key bit, when the output of both the circuits is same, it implies that the injected fault on the key lines in a C_F circuit is same as the correct key bit, only then output of both the ICs will be same. Otherwise, when C_F and C_A differ in their output response, it concludes the correct key bit is complementary to the induced fault. This process is repeated for all key bits. In this manner, the key value can be extracted by comparing the output responses of both circuits for the same primary input pattern.

3.2.1 Examples

In this section, we present two example circuits to illustrate the proposed attack. Test pattern generation for detecting stuck-at faults at the key lines are described using the D-Algorithm [71]. Combinational circuits are chosen as an example for simplicity. However, the attack is valid for sequential circuits as well as it can be transformed into a combinational circuit in the scan mode, where all the internal flip-flops can be reached directly through the scan chains [71].

Figure 3.4 shows a simple locked circuit with a 2-bit key, where the effect of one key does not impact the other key. The circuit has six inputs ($PI = 6$) and two outputs ($PO = 2$). It is necessary to generate a test pattern that detects a *saf* at k_0 with constraint $k_1 = 1$, which is shown in Figure 3.4.(a). $\overline{D}$ is assigned after the *sa1* at the key line k_0. D is defined as logic 1 for a good circuit and logic 0 for a faulty one [71]. To activate this fault, the ATPG tool will assign a logic 0 at k_0. It is required to propagate $\overline{D}$ to any of the primary outputs. For example, 1 at the output of gate G_1 will result in D at the output of key gate G_{k_0}. Inputs $[x_0\ x_1] = [1\ 1]$ can satisfy this condition since G_1 is an AND gate. Next, D

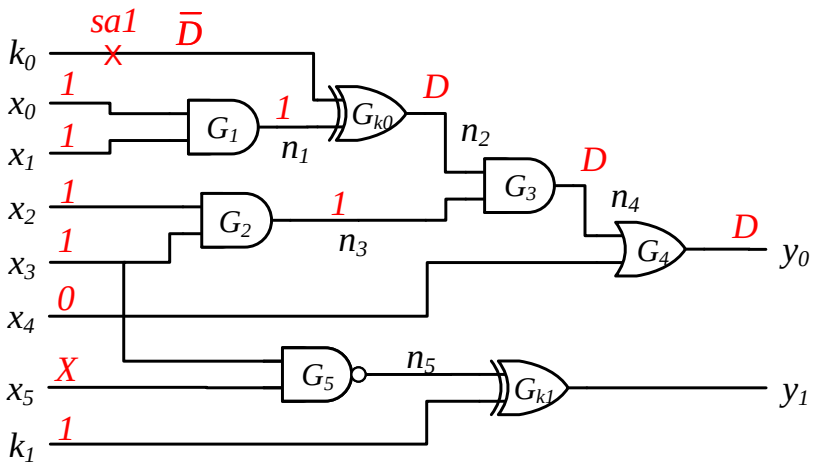

Figure 3.2: Test pattern generation for the test circuit locked with a 2-bit secret key, considering a *sa1* at key line k_0 with constraint $k_1 = 1$. Test pattern, $P_1 = [11110X]$ will be applied to the faulty circuit C_F and C_A to perform DFA.

appears at n_4 for $[x_2\ x_3] = [1\ 1]$. Finally to propagate D at the output y_0, x_4 needs to be put to logic 0 as G_4 is an OR gate. As a result, D is observed at the output y_0 for primary input $P_1 = [x_0\ x_1\ x_2\ x_3\ x_4\ x_5] = [1\ 1\ 1\ 1\ 0\ X]$. ***Note that the output y_0 will have complementary values for $k_0 = 0$ and $k_0 = 1$ when we apply P_1 at the input.*** This property of the input patterns will be used in DFA to recover the secret key. Similar analysis can be performed to detect a *saf* D on key line k_1.

After generating the test pattern P_1 for the *sa1* at key line k_0, the next step is to perform differential fault analysis between the responses of the C_F and C_A. As this pattern detects a *sa1* at line k_0, the faulty response will be propagated to the output y_0. The test pattern is applied first to the faulty chip C_F, injected with a logic 1 fault on k_0 and k_1 (shown in red color on the key inputs), and its response is captured, which is shown in Figure 3.3. The output response observed at y_0 is logic 1 for C_F. The same test pattern is applied to fault-free functional chip C_A which is injected with a logic 1 fault on k_1 (represented with red color on k_1 input in Figure 3.3) and no fault on k_0. With reference to Figure 3.3.(a), if the correct value for the key bit k_0 stored in tamper-proof memory is logic 0 (shown with green

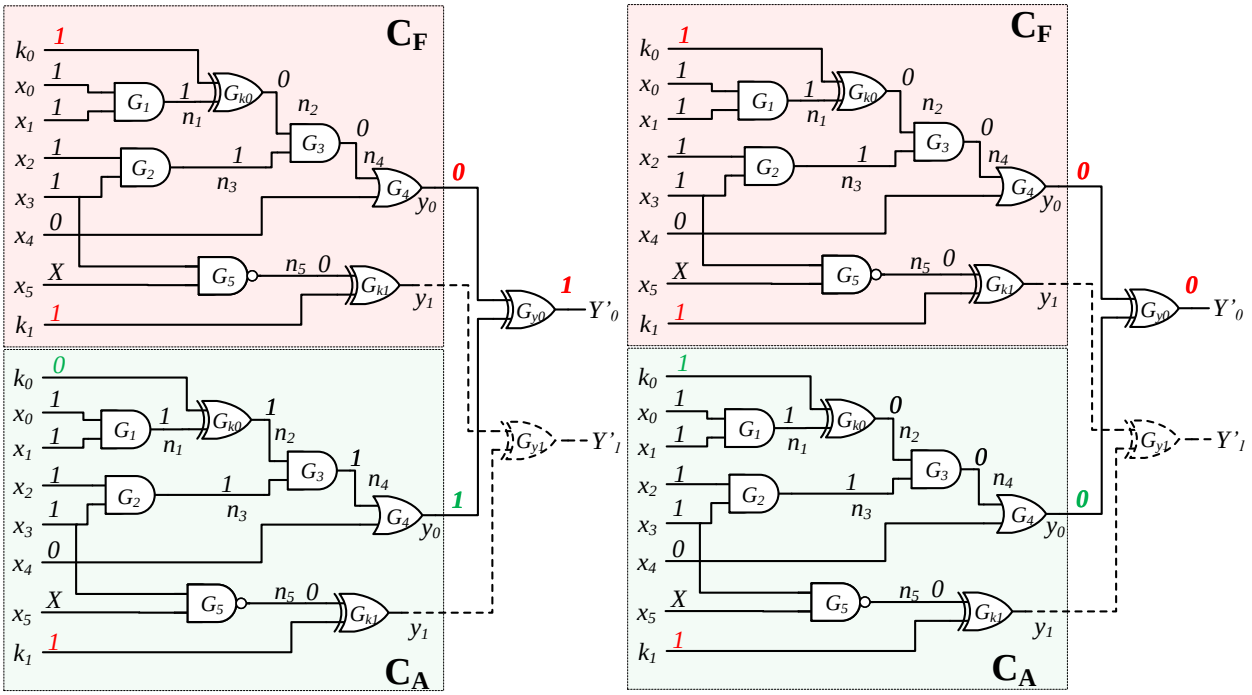

Figure 3.3: DFA attack using the generated test pattern $P_1 = [11110X]$ to determine the key-bit k_0 based on the response for C_F and C_A. (a) The output $Y_0^{'} = 1$, when the stored key k_0 is complementary to the injected fault (logic 1) on C_F (b) The output $Y_0^{'} = 0$, when the stored key k_0 is same as the the injected fault (logic 1) on C_F

color on key input k_0 in C_A), then output at $y_0 = 1$. Upon comparing the outputs of the two circuit (i.e. C_F and C_A) through the XOR gate (G_{y0}), if the output $Y_0^{'}$ is logic 1, which indicated that the two circuit instances produced different outputs. This also implies that the correct key value for k_0 is complementary to the injected fault on k_0 in C_F circuits, hence $k_0 = 0$. Similarly, if the correct stored value for k_0 is logic 1 (as shown in Figure 3.3.(b) with green color input k_0), the two circuit instances, C_F and C_A, produces the same output. This implies that the XOR between the corresponding outputs for C_F and C_A will produces $Y_0^{'} = 0$, indicating that the logic 1 fault injected on key register k_0 in C_F is the correct value for the key bit k_0. Similarly, the test pattern for detecting a *sa1* at k_1 can be applied to extract its value based on the difference between the two circuit instances.

Figure 3.4 shows our proposed attack on the same test circuit locked with a 3-bit secret key, where the propagation of k_0 is dependent on k_1 and vice versa. The attack targets all

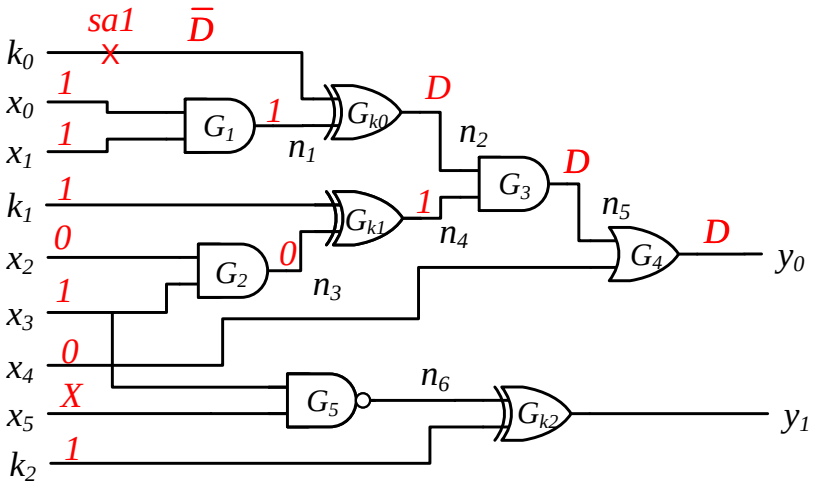

Figure 3.4: Test pattern generation for the test circuit locked with a 3-bit secret key, where the propagation of k_0 is dependent on k_1 and vice versa. Test pattern generation considering a *sa1* at key line k_0 with constraint $k_1 = 1$ and $k_2 = 1$. Test pattern, $P_1 = [1\ 1\ 0\ 1\ 0\ X]$ will be applied to the faulty circuit C_F and C_A to perform DFA.

the key bits separately like before. First, we target to find out the value of k_0. A test pattern P_1 is generated to detect a *sa1* fault at k_0 with constraint $k_1 = 1$ and $k_2 = 1$. As the value of k_1 is known during the pattern generation, the effect of the *sa1* at k_0 will be propagated to the primary output y_0. For a fault value $\overline{D}$ at k_0, if $[x_0\ x_1] = [1\ 1]$ then D propagates to n_2. To propagate the value at n_2 to the output of G_3, its other input (n_4) needs to attain logic 1. Since $k_1 = 1$ due to injected fault which is set as a constraint in ATPG tool, $n_4 = 1$ for $n_3 = 0$ which implies $[x_2\ x_3] = [0\ 1]$. At last, $x_4 = 0$ propagates D propagates the value at n_5 to the primary output y_0. The output y_0 can be observed as D for the test pattern $P_1 = [1\ 1\ 0\ 1\ 0\ X]$. Finally, this pattern P_1 needs to be applied to both C_A and C_F to determine the value of k_0.

Once the test pattern is generated, the output response for C_F and C_A needs to be compared for this test pattern to perform DFA, as shown in Figure 3.5. To determine k_0, C_F includes logic 1 fault on k_0, k_1 and k_2 (shown in red color on the key inputs in C_F circuit), injected through a fault injection tool. Whereas, C_A has the same fault (i.e. logic 1) on the

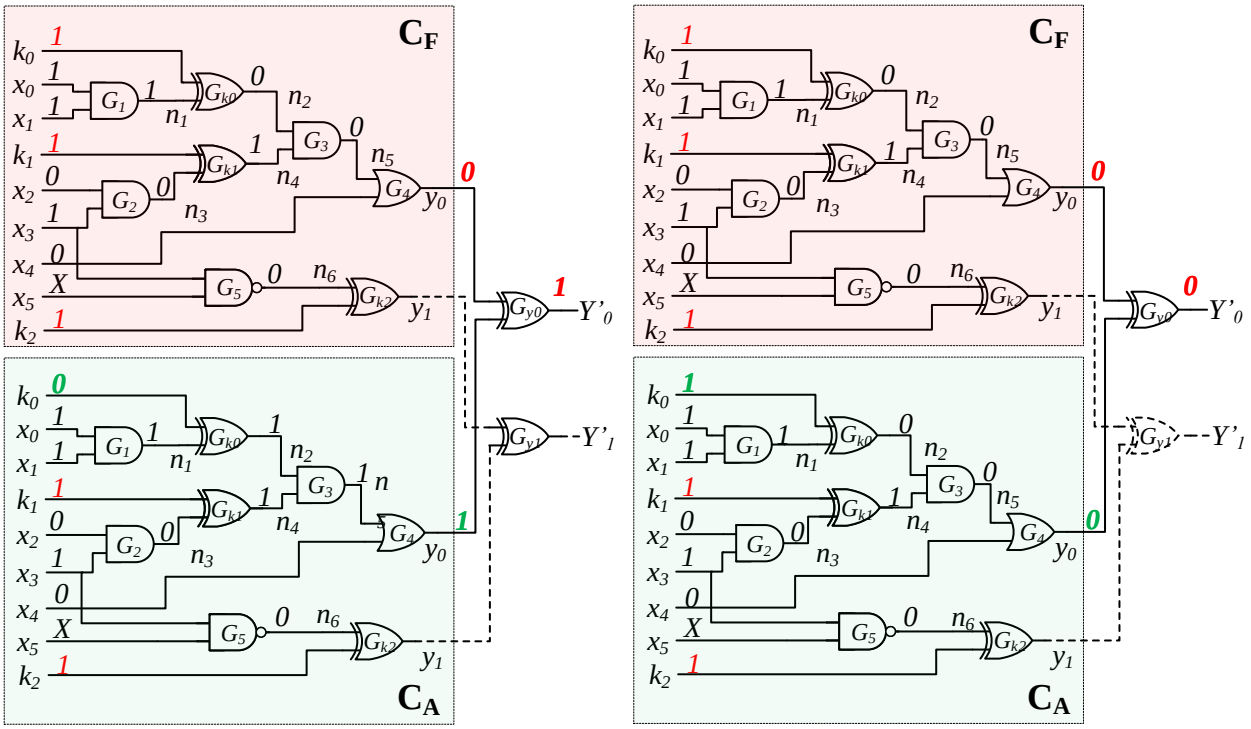

Figure 3.5: DFA attack using the generated test pattern $P_1 = [1\ 1\ 0\ 1\ 0\ X]$ to determine the key-bit k_0 based on the response for C_F and C_A. (a) The output $Y_0^{'} = 1$, when the stored key k_0 is complementary to the injected fault (logic 1) on C_F (b) The output $Y_0^{'} = 0$, when the stored key k_0 is same as the the injected fault (logic 1) on C_F.

key inputs k_1 and k_2 (represented with red color on key inputs in C_A circuit), excluding k_0. Consider that the correct key value stored for k_0 is equal to 0 (see Figure 3.5.(a), represented in green on k_0 in C_A), the output responses for C_F and C_A are not the same which is indicated by $Y_0^{'} = 1$, when XOR operation is performed between the y_0 output of C_F and C_A. This reflects that the injected fault on k_0 in C_F is complementary to the correct value of k_0 stored in the tamper-proof memory. Figure 3.5.(b) shows the case when the correct value $k_0 = 1$ (represented with green color on k_0). Upon performing the XOR operation between the y_0 output of the two respective instances, resultant $Y_0^{'} = 1$. This implies that C_F and C_A produced the same output, which also indicates that the injected fault (logic 1) on k_0 in C_F circuit is indeed the correct value for the key. Similar analysis can be performed for other two key bits, k_1 and k_2. In this manner, the correct key can be retrieved by preforming the differential fault analysis on logic locking.

3.3 Test Pattern Generation

To generate the test pattern set, an automated process relying on constrained ATPG is performed. The detailed steps to be followed are provided in Algorithm 1. Synopsys Design Compiler [56] is utilized to generate the technology dependent gate level netlist and its test protocol from the RTL design. A test protocol is required for specifying signals and initialization requirements associated with design rule checking in Synopsys TetraMAX [57]. Automatic test generation tool TetraMAX generates the test patterns for the respective faults along with constraints for the locked gate level netlist.

Algorithm 1: Test pattern generation for constrained ATPG

Input : Locked gate-level netlist (C_L), test protocol (T), and standard cell library

Output: Test pattern (P) set

```
1  Read the locked netlist (C_L) ;
2  Read standard cell library ;
3  Run design rule check with test protocol generated from design compiler ;
4  Determine key size |K| from C_L ;
5  for i ← 0 to (|K| − 1) do
6  |   Add a sa1 fault at key line k_i ;
7  |   for j ← 0 to (|K| − 1) do
8  |   |   if i ≠ j then
9  |   |   |   Add constraint at k_j to logic 1 ;
10 |   |   else
11 |   end
12 |   Run ATPG to detect the fault ;
13 |   Add the test pattern, P_i to the pattern set, P ;
14 |   Remove all faults ;
15 |   Remove all constraints ;
16 end
17 Report the test pattern set, P ;
```

The inputs to the algorithm are locked gate-level netlist (C_L), Design Compiler generated test protocol (T) and the standard cell library. The algorithm starts with reading the locked netlist and standard cell library (Lines 1-2). The ATPG tool runs the design rule check with the test protocol obtained from the Design Compiler to check for any violation

(Line 3). Only upon completion of this step, the fault model environment is set up in the tool. The size of the key ($|K|$) is determined by analyzing C_L (Line 4). The remaining key lines are selected one by one to generate test patterns (Line 5). A stuck-at-0 fault is added at the i^{th} key line to generate P_i (Line 6). The ATPG constraints (logic 1) are added to other key lines (Lines 7-10). A test pattern P_i is generated to detect the *sa1* at the i^{th} key line (Lines 12-13) and added to the pattern set, P. All the added constrains and faults are removed to generate the $(i+1)^{th}$ test pattern (Lines 14-15). Finally, the algorithm reports all the test patterns, P (Line 17).

3.4 Complexity Analysis on Logic Locking

The complexity of the proposed ATPG guided fault injection attack is linear with the key size (K). In this section, we show how the proposed attack is very effective at breaking any logic locking technique. However, the fault injection time may vary depending the effectiveness of the equipment. It is practically instantaneous to obtain the secret key once the responses are collected from the C_A and C_F.

Lemma 3.4.1 *One input pattern is sufficient to recover one key bit.*

A single test pattern is sufficient to detect a *saf* if such a fault is not redundant [71]. A redundant fault results from a redundant logic that cannot be exercised from the inputs. As the key gates are placed to modify the functionality, it cannot be a redundant logic. As there exists one test pattern to detect a *saf* at the key line, it can be used to recover one key bit.

Theorem 3.4.2 *The attack recovers the entire secret key, K using at most $|K|$ number of test patterns, i.e.,*

$$TP[f_K(C_L) = f(C_O)] \leq |K| \tag{3.1}$$

where, $f_K()$ represents the functionality with K as the key.

A C_L with a $|K|$-bit key is injected with a *saf* fault on every key lines. As the proposed attack requires one test pattern to obtain one key bit (see Lemma 3.4.1), the upper bound of the number of test patters is $|K|$. However, a single pattern can detect two or more stuck-at faults on the key lines if their effect is visible in different logic cones (e.g., different outputs). As a result, the required number of test patterns to recover the entire key (K) can be less than $|K|$.

Theorem 3.4.3 *ATPG guided DFA attack is applicable to strong logic locking [58], where pairwise key gates are inserted to block the propagation of one key by the other.*

In strong logic locking, the propagation of one key is blocked due to the other key. However, $(|K| - 1)$ faults are injected at $(|K| - 1)$ key lines except for the one whose value needs to be determined. Once an external fault is injected to the functional chip, the key value is fixed and no longer remains unknown. Hence, the proposed attack is applicable to strong logic locking.

In this chapter, we demonstrated how differential fault analysis can be performed on logic locking techniques to extract the key bit and undermine its security. The complete steps for the attack including the efficient test pattern generation using constrained ATPG is proposed. Then, we discussed how fault injection methods can assist in performing the differential fault analysis on functional chips for the generated test patterns, where constraints are converted to external induced faults.

Chapter 4

ATPG Guided Key Sensitization Attack Methodology

The important aspect for any logic locking technique is to conceal the secret to obscure the circuit functionality. An adversary with the intention of exposing the key can directly observe the key information by propagating the key value, either at the primary output or through scan-chain by applying a suitable input pattern. The differential attack methodology demonstrated in the previous chapter requires large number of constraints while generating the test pattern and corresponding external injected faults on the functional circuits as well. Moreover, differential fault analysis requires creating the two different functional circuit instances (i.e. fault-free and faulty) for comparison. This chapter presents a novel ATPG-guided key sensitization attack that overcomes the drawbacks of the previous method. Additionally, a functional chip is sufficient to launch the attack which requires much less number of external faults to be injected. The threat model remains consistent with the previous chapter, where an untrusted foundry or an reverse engineer is assumed to be the adversary. The attacker possess an unlocked functional chip, loaded with the correct key in its tamper-proof memory. Also, it has access to the locked netlist and the capability to inject fault with the fault injection equipment.

4.1 Key Sensitization Methodology

The proposed fault injection attack relies on sensitizing the key bits, where an input pattern is chosen such that it results in directly the value at any primary output irrespective of the other key bits in the circuit. To mask the information of other keys, a practical fault injection approach described in Section 4.2 can be used. Figure 4.1 shows an abstract representation of our proposed approach. For an input pattern, the output responses collected

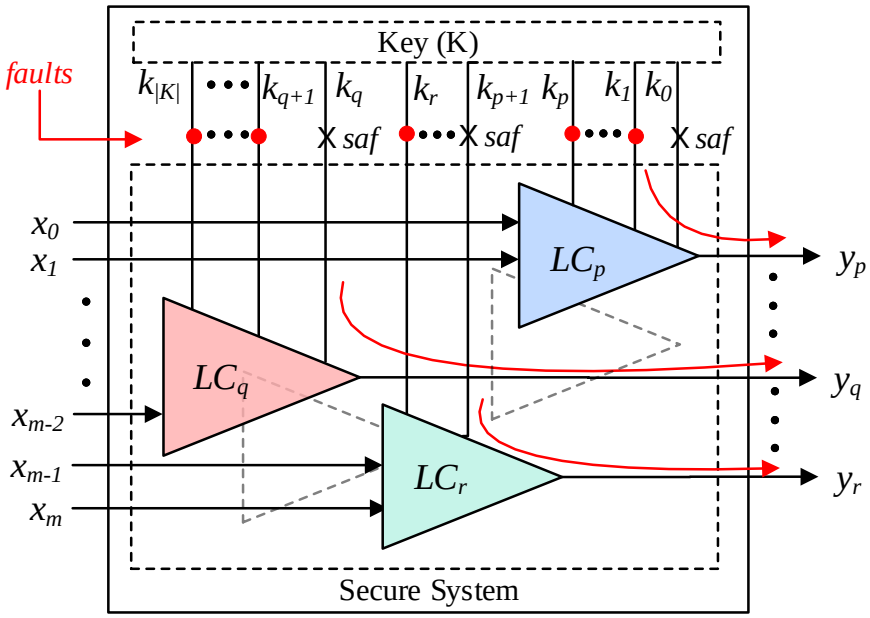

Figure 4.1: The abstract representation of the proposed ATPG guided key sensitization attack using fault injection on logic locked circuit.

for the fault induced functional chip reveals key bits. To generate such an input pattern, we utilise the stuck-at fault based test pattern generation which is a common VLSI testing. A single stuck-at fault detection on the key inputs is sufficient to generate the test pattern that activates the fault and propagates it to the observable primary output. The entire process can be divided into two critical steps described as follows:

- *Step-1*: To begin with, the first step includes the selection of an input pattern that leads to the exposure of the key bit to the primary output from a functional chip. In other words, an input is required such that it satisfies the condition on all the internal nodes of the circuit to propagate the key value to the primary output. However, it is very likely to observe the inter-dependency of the key bits such that the propagation of a particular key bit to the primary output is not possible as one/more key bit(s) may mask or block the propagation. As the adversary do not possess the correct key value for other key bit(s), it will be impractical to obtain such an input pattern. To generate the input patterns, we utilise the Automatic test pattern generation tool (ATPG) [71]. The basic idea of ATPG is to generate a pattern for stuck-at fault at a key line so that

a key bit can be propagated to an output. As the adversary do not know the value for other key bits, using the unconstrained ATPG will not help in generating the test vector because one one/more other keys may block propagation of the targeted key input. Fault injection is an efficient possible solution to this problem, so that ATPG can set a constrain on these other key bits as 1 (or 0) depending on the corresponding to the injected faults, and generate a test pattern.

To effectively generate a test pattern that can sensitize multiple key bits with less number of injected faults, logic cone analysis needs to be performed so that we can identify groups of keys for each cones and also the primary output at which they can be observed. When we target a particular key bit to be sensitized, the other key bits in the same cone are the only ones blocking the propagation. That is reason, only these the key bits in the same cone needs to be constrained. In the abstract circuit view shown in Figure 4.1, the logic cone analysis revealed the different logic cones in the circuit and the key bits associated in the respective cones. Note that the sequential elements are not shown here for simplicity of understanding, however, with access to the internal scan chains (test architecture), any sequential circuit can be converted to individual combinational circuits [71]. A test pattern can detect a *saf* (e.g., *sa1* or *sa0*) at key line k_0 while other key lines $k_1 - k_p$ are constrained to logic 1 (or 0) as they lie in the same logic cone (LC_p). This is mainly because other keys in that particular cone (LC_p) shows dependency while generating the test pattern for k_0. The same test pattern can also be used to detect a *saf* at key line k_{p+1} while all the keys in the logic cone (LC_r) are constraint with logic 1 (or 0). Additionally, a *saf* detection on k_q can also be included in this test pattern because k_q belongs to a separate logic cone (LC_q), where all the other in that cone $k_{q+1} \ldots k_{|K|}$ will be added with the constraint. This is mainly due to no dependency within the three logic cones (i.e. LC_p, LC_q, and LC_r) and their responses being observable at different primary outputs.

- *Step-2*: The next step involves the extraction of the targeted key bit(s) from the functional chip loaded with the correct key in its tamper-proof memory, using the test pattern generated in the previous step. We need to first inject faults at the constrained key lines (registers) using any practical fault injection methods. Any external fault injection methods can used to obtain the fault injected circuit as mentioned in section 2.3. The test pattern obtained in the previous step is then applied to the fault-induced functional chip. This will lead to the exposure of the targeted key bit(s) to a their corresponding primary output associated with their logic cones. Once these key bit(s) are determined, the test pattern generation for the next targeted key bit(s) in the same or different logic cones is performed. Once the correct value of a key is determined for a particular cone, this correct key value shall be used as a constraint at its key input in the ATPG. The above two steps are repeated for other key bits unless all the key bits are determined for a cone. This reduces the overall number of the faults to be induced as only the keys affecting the key propagation through a particular logic cone are induced with the fault. Additionally, once a key bit is known already, that key register need not be injected with external fault for the next input pattern as it already holds the correct value in in its tamper-proof memory which is used as the constraint. Moreover, multiple test patterns can be combined together into a single pattern if the targeted key bits lies in different logic cones propagating at different primary output with no conflicts at the primary inputs to reduce the overall number of test patterns.

4.1.1 Example

In this section, an example circuit is presented to illustrate the proposed attack on logic locking. The D-Algorithm is used to describe the test pattern generation for detecting stuck-at faults at the key inputs/wires [71]. For simplicity, a combinational logic circuit is chosen as an example. However, with scan access, any sequential circuit can be transformed

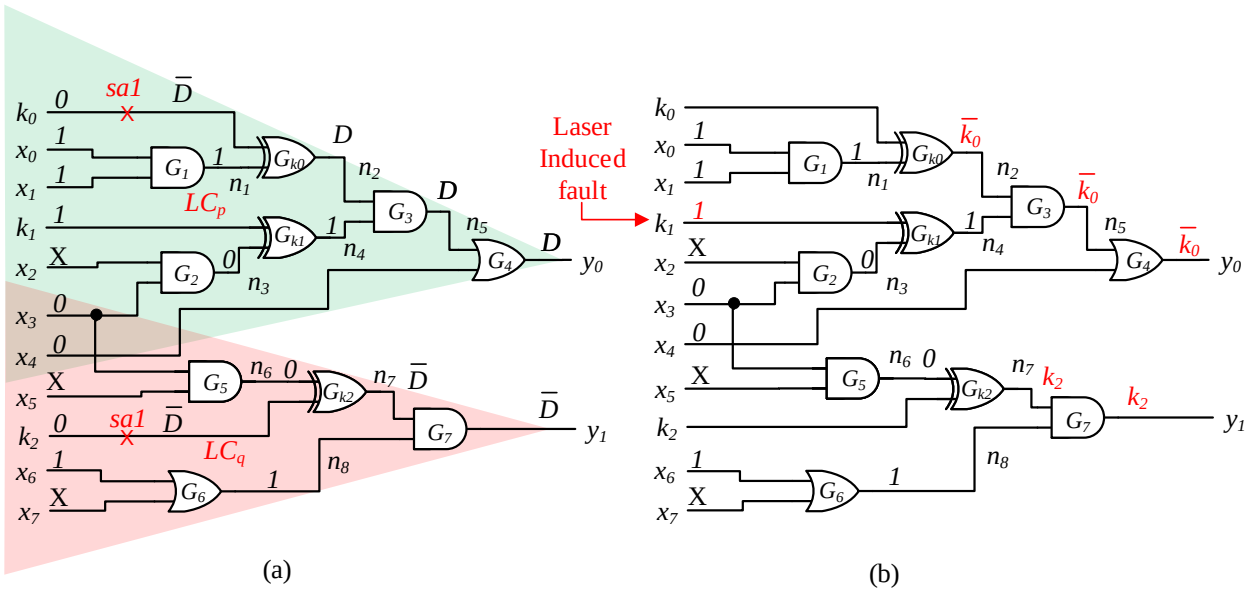

Figure 4.2: Key sensitization attack on a test circuit locked with 3-bit key (a) Test pattern generation after performing logic cone analysis to identify keys in LC_p and LC_q to detect *sa1* on k_0 and k_2 with constraint $k_1 = 1$. Test pattern $P_1 = [1\ 1\ X\ 0\ 0\ X\ 1\ X]$ will be applied to fault induced functional chip (b) The same test pattern is applied to the functional chip induced with external logic 1 fault on k_1 to extract k_0 and k_2.

into a combinational circuit in the scan mode, where all the internal flip-flops can be reached directly through the testability architecture (i.e. scan chain).

Figure 4.2 shows the proposed attack on the combinational test circuit, locked with a 3-bit secret key. The circuit consists of six inputs ($|PI| = 8$) and two outputs ($|PO| = 2$). Before generating the test pattern, it is important to analyze the circuit to identify the key bit in different logic cones which will assist in determining the number of constraints required and also the number of key bits that can be determined with every test pattern. The cone analysis reveals that k_0 and k_1 are in LC_p, while k_2 are in LC_q and these two groups can be targeted simultaneously. First, we target the key bits k_0 and k_2. The test pattern is generated to detect a *sa1* at k_0 and k_2. However, since k_0 and k_1 are in LC_p, a constraint of $k_1 = 1$ is required which is shown in Figure 4.2.(a). D is assigned after the sa1 at the key line k_0 and k_2. From logic point of view, D is represented as logic 1 for a good circuit and logic 0 for a faulty one [71]. To activate these faults, the ATPG tool will assign a logic 0 at k_0 and k_2. First, we analyze the fault propagation for k_0 in logic cone (LC_p). As the value of

k_1 is constraint during the pattern generation, the effect of the sa1 at k_0 will be propagated to the primary output y_0. A fault value $\overline{D}$ at k_0 propagates D to n_2, if $[x_0\ x_1] = [1\ 1]$ as G_1 is an AND gate. To propagate the value at $n2$ to the output of G_3, its other input (n_4) needs to attain logic 1. To satisfy this, we put $n_3 = 0$ and $k_1 = 1$ already due to injected fault which is set as a constraint in ATPG tool. This implies $[x_2\ x_3] = [X1]$. Finally to propagate the D to the primary output y_0, x_4 is set to logic 0. Similarly, if we analyze logic cone (LC_q), the response for the *sa1* on k_2 will be observed at y_1. The $\overline{D}$ represented on k_2 can be observed at n_7 when $n_6 = 0$. To satisfy this primary inputs $[x_3\ x_5] = [0\ X]$ as G_5 is an AND gate. The $\overline{D}$ propagates to the primary output y_1 to detect *sa1* on k_2 when n_8 is 1, implies $[x_6\ x_7] = [1\ X]$. Test pattern $P_1 = [x_1 x_2 \dots x_7] = [1\ 1\ X\ 0\ 0\ X\ 1\ X]$ can detect *sa1* faults at key lines k_0 and k_2. Figure 4.2.(b) shows the attack to extract k_0 and k_2 from the functional chip which is injected with a logic 1 fault on the key register k_1, corresponding to the constraint. The logic value obtained at output node y_0,and y_1 are $\overline{k_0}$ and k_2, respectively. Next, we need to generate the next pattern P_2 for extracting key k_1, while we add a constraint at key line k_0 as its correct value extracted from the previous pattern. Note that we do not need to put a constraint at k_2 as the propagation of k_1 is not dependent on k_2 because they belong to different logic cones.

4.2 Fault-injection Approach

Fault-injection attack has been widely used in the past to extract secret assets and bypassing security measures in the device [112]. An adversary can use several fault-injection approaches depending on the budget and expertise. The basic fault-injection approach includes voltage, timing, electromagnetic, and laser-based fault-injection methods [113, 114, 115]. Laser-fault injection (LFI) offers the most precision in both spatial and temporal domains during the operation of the chip, hence, used for deploying DFA attack for extracting the secret key. Laser with photon energy higher than silicon bandgap energy used to induce

faults in an integrated circuit [115]. Therefore, the laser with a wavelength less than 1.1 μm is used in our experiment. The LFI attack can be completed in the following steps:

• **Sample Preparation:** LFI can be injected from both frontside and backside of the chip. However, the interconnecting metal layers at the front of the die obstruct the optical path of photons. On the other hand, the absence of any metal obstacle or reflective coating at the backside of the die allows an adversary to access the transistors with the laser. In a typical packaged chip (bondwire IC), the backside can be exposed by wet etching. Nonetheless, the flip-chip substrate is typically covered with a metallic lid, which can be easily removed to expose the silicon die. The backside of the silicon can be further polished to 30 – 100 μm to reduce the power loss along the laser path due to photon absorption phenomena [116, 115].

• **Target Localization and Fault-injection:** The method of localizing key-register location depends on the capability and asset availability to an adversary. An adversary, like an untrusted foundry or an expert reverse engineer, can localize the key location, i.e., tamper-proof memory, key-register, key-gates by analyzing the GDSII or partial/full-blown reverse engineering. Once the target is localized, an attacker needs to identify the fault sensitive location for injecting fault. Localizing the most reverse biased P-N junction in the key-register can be identified as the potential candidate for fault-injection [116]. Therefore, depending on logic 1 (logic 0) fault, the laser can be applied to the drain location of the p-type (n-type) MOS transistors for fault injection.

Another challenge is that a single laser source can only inject a single fault at once. Therefore, the fault can be injected in a sequential order where the laser source can be moved from one key-register to another for injecting fault. After localizing the targeted key registers, an adversary can automate the sequential fault-injection process with the help of computer vision and image processing [117, 30]. Since the key is imperative for the IP operation, it is safe to assume that once secure boot-up is complete, the locking key will remain stored in the key-register during the operation of IP [111, 43]. Therefore, an adversary can initiate the fault-injection method after the secure boot-up of the chip is complete. An adversary can

identify the clock-cycle required for secure-boot up by monitoring the power consumption of the circuit.

In this chapter, we demonstrated how ATPG-guided key sensitization attack relying on fault injection can be performed on logic locking. This attack overcomes the drawbacks of the previous differential fault analysis in terms of the number of constraints and corresponding faults required. Next, we discussed the laser fault injection approach, which is an efficient fault injection approach due to its high spatial and positional resolution capability.

Chapter 5

Experimental Results

In this chapter, we demonstrate the hardware implementation of the proposed novel fault injection attack methodologies presented in Chapter 3 and Chapter 4. The experiment is performed on FPGA, however, it is valid on any ASIC/SoC as well. Moreover, the attack is also valid for large industrial design compared to the small benchmark circuits selected for the experimentation. We first discuss the experiment setup and steps involved in it. Thereafter, we present provide the details for laser fault injection tool and how it is used to perform induce faults on the circuit implemented in the FPGA.

To evaluate the effectiveness of our proposed attack, we adopted and performed the laser fault injection technique on a Kintex-7 FPGA, which is used as the device-under-test (DUT). Different benchmark circuits are implemented in a Kintex-7 FPGA, where the faults are injected on the key registers. The complete experimentation flow is shown in the Figure 5.1. First, the RTL netlist for ISCAS'99 benchmark circuits [54] are synthesized using 32nm technology libraries in Synopsys Design Compiler [56]. The technology-dependent gate-level locked netlist is given to the Synopsys TetraMAX ATPG tool [57] to generate test pattern set P using Algorithm 1. The same netlist is then converted into a technology-independent gate-level Verilog code using our in-house PERL script. This is primarily done to assure that the circuit implemented in the FPGA is exactly the same circuit for which the test pattern set is generated. Otherwise, fault propagation cannot be ensured. The implemented circuit in FPGA for ISCAS'85 c432 is shown in Figure 5.2. Fault injection is performed on the circuit loaded into the FPGA, which leads to the instances of faulty and fault-free circuits by laser-induced faults on the key registers. Additionally, the implemented design includes a separate universal asynchronous receiver/transmitter (UART) module, which is used for

communication between the computer and the FPGA. The inputs are applied through the real-term monitor and responses are collected on the same. Once the response for any key-bit is obtained, the step is repeated for all the key bits in a benchmark circuit. Finally, the key-bits are exposed through the comparison between the corresponding instances of the circuits as explained in Section 3.2.

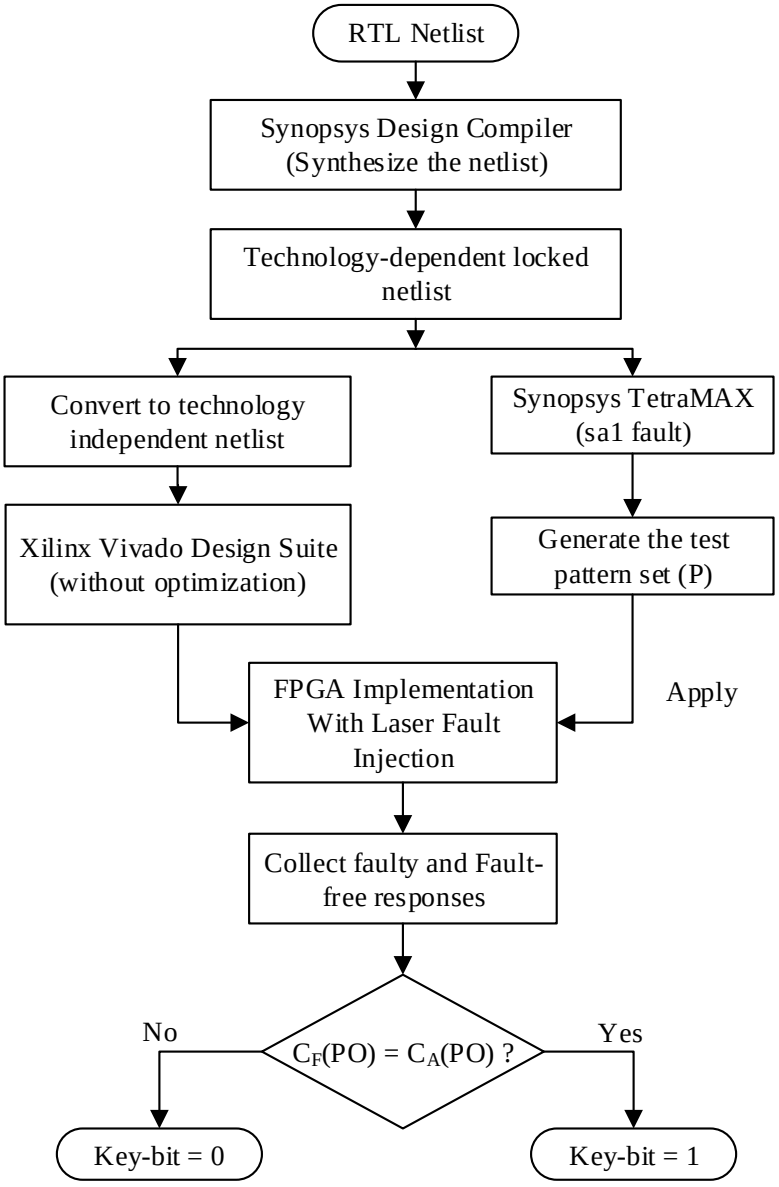

Figure 5.1: Methodology for ATPG guided fault injection attacks in Kintex-7 FPGA

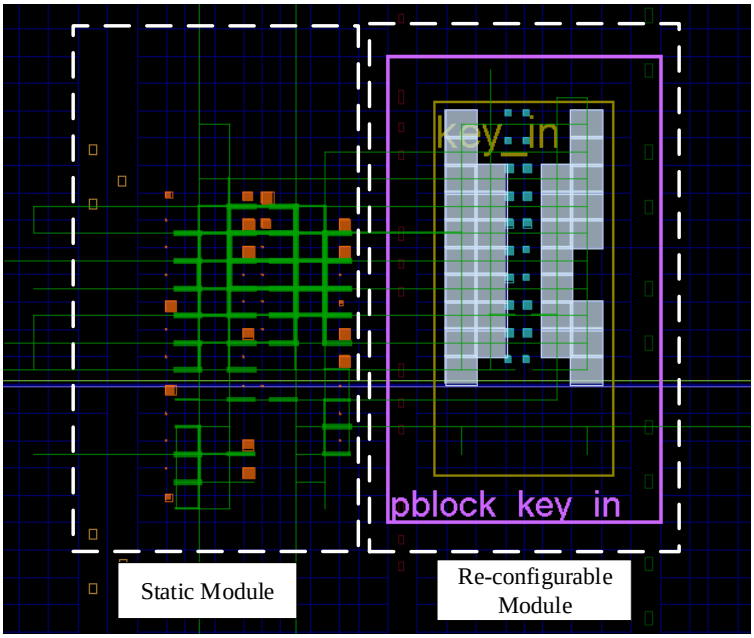

Figure 5.2: Vivado Design Suite generated design for c432 benchmark targeting Kintex-7 FPGA

5.1 Laser Fault Injection Attack

The laser fault injection (LFI) setup is provided by a Hamamatsu PHEMOS-1000 FA microscope as shown in 5.3, available in FICS Research SeCurity and AssuraNce (SCAN) Lab at the University of Florida [1]. Specifically, the LFI experimentation was performed by our collaborators at University of Florida. The equipment consists of a diode pulse laser source (Hamamatsu C9215-06) with a wavelength of 1064 nm. Three objective lenses were used during this work: 5x/0:14 NA, 20x/0:4 NA, 50x/0:76 NA. The 50x lens is equipped with a correction ring for silicon substrate thickness. The laser diode have two operation modes – a) low power (200 mW) pulse mode, and b) high power (800 mW) impulse mode. The high power impulse mode can be used for laser fault injection. The laser power can be adjusted from 2% to 100% in 0.5% steps. Photon emission analysis [118] is used to localize the implemented locked circuitry in the DUT. Thereafter, The DUT is placed under the laser source for LFI. A trigger signal is fed to the PHEMOS-1000 to synchronize the LFI with DUT operation. Once the device reaches a stable state after power-on, the laser is triggered

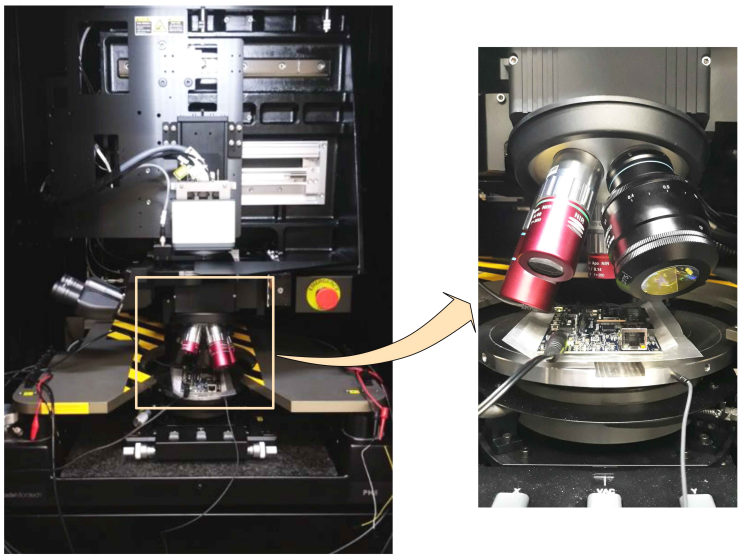

Figure 5.3: The FPGA board placed under the lens for laser-fault injection at the target registers[1].

on target key-registers. After the fault injection, we have to guarantee that the device is still functioning as expected and has not entered into a completely dysfunctional state. The laser triggering timing can be checked by a digital oscilloscope for greater precision.

We have performed and verified our results for different benchmark circuits implemented with random logic locking (RLL) [119], strong interference-based logic locking (SLL) [20] and fault-based stripped functionality logic locking (SFLL-Fault) [120]. For RLL, we selected locked instances of c432 and c2670 benchmark circuits with a 32-bit key and 128-bit key respectively obtained from Trust-hub [55]. For SLL, we selected c1355 and c1908 locked benchmarks with 128-key bits, also obtained from Trust-Hub. We also implemented the attack on the circuit locked with a combination of SFLL-fault (40-bit key) and RLL (40-bit key) technique. We successfully recovered the entire key for all the circuits which proves the effectiveness of our proposed ATPG-guided fault injection attack.

This chapter presented the experimentation flow and results to validate the proposed fault injection attack. With the use of a laser fault injection tool, the key registers for the circuits implemented in the FPGA can targeted to perform the attack methods proposed in this thesis to undermine the security of logic locking techniques.

Chapter 6

Future Research Direction and Conclusion

The security of a logic locking technique can be tied together with the detection and avoidance of fault injection attacks. Developing a SAT-resilient logic locking is not sufficient enough to prevent IC overproduction or to protect IPs. It is required to address the detection of external fault injection attacks and prevent them. The countermeasures to secure the device against such techniques involve modification either at the design or package. The design-based defense techniques detect a fault by using recalculation or error detection code [121, 122]. These methods are regarded efficient for cryptographic primitives where duplication of the hardware blocks (e.g., s-box) is possible. However, these methods cannot be applied for logic locking as the key gates are distributed in the entire design. Meanwhile, at the package level, sensor structures are studied to restrict the vulnerability against external fault injection [123, 124], where a sensor detects an attempt of fault injection and takes a respective action of powering down or flushing internal storage. However, these sensors works on detection of a bit-flip, which can reveal the crucial information regarding the key value. If no alarm is raised, one can infer the value of a key bit is the same as the injected fault [125]. Besides, Li at el. also demonstrate an attack using side-channel analysis [126]. Despite various countermeasures, the community still lacks a robust countermeasure against laser fault injection. Once the detection of external fault injection attacks is ensured, an SoC designer can choose a SAT-resistant logic locking to prevent IC overproduction and IP piracy.

The future research work will focus on preventing these proposed fault injection attacks on logic locking through restricting the attacker's access to the locked netlist such that test pattern generation through ATPG is prevented. To do so, we plan to develop a design-level

obfuscation technique, where it is difficult to determine the functionality of the gates. Additionally, we plan to implement these ATPG-guided fault injection attack on cryptographic IP core and develop the preventive measure as well.

In conclusion, we have presented novel ATPG-guided stuck-at fault based attacks to undermine the security of any logic locking technique. Both the attacks presented in this thesis relies on injecting faults on the key registers through hardware to expose the secret key from the functional IC for the ATPG generated test patterns. We have demonstrated and validated the attack on circuits implemented in the FPGA using the laser fault injection method. The results depicted the success of the proposed attack on different logic locking techniques, irrespective of their SAT resiliency. The detection and prevention against such physical attacks is very important to achieve the robust hardware security solution against the emerging threats due to untrusted semiconductor manufacturing and supply chain.

Bibliography

[1] FICS Research SeCurity and AssuraNce (SCAN) Lab, , University of Florida, url = http://fics.institute.ufl.edu/facilities/.

[2] Age Yeh, "Trends in the global IC design service market," DIGITIMES Research, 2012.

[3] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*, 2008, pp. 1069–1074.

[4] Y. Alkabani and F. Koushanfar, "Active Hardware Metering for Intellectual Property Protection and Security." in *USENIX security symposium*, 2007, pp. 291–306.

[5] R. S. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 674–677.

[6] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proc. of IEEE/ACM int. conf. on Computer-aided design*, 2007, pp. 674–677.

[7] J. Huang and J. Lach, "IC activation and user authentication for security-sensitive systems," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 76–80.

[8] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[9] U. Guin, Q. Shi, D. Forte, and M. M. Tehranipoor, "FORTIS: a comprehensive solution for establishing forward trust for protecting IPs and ICs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 4, p. 63, 2016.

[10] U. Guin, Z. Zhou, and A. Singh, "A novel design-for-security (DFS) architecture to prevent unauthorized IC overproduction," in *Proc. of the IEEE VLSI Test Symposium (VTS)*, 2017, pp. 1–6.

[11] E. Castillo, U. Meyer-Baese, A. García, L. Parrilla, and A. Lloris, "IPP@ HDL: efficient intellectual property protection scheme for IP cores," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 5, pp. 578–591, 2007.

[12] M. Tehranipoor and C. Wang, *Introduction to hardware security and trust*. Springer Science & Business Media, 2011.

[13] M. M. Tehranipoor, U. Guin, and D. Forte, "Counterfeit integrated circuits," in *Counterfeit Integrated Circuits*. Springer, 2015, pp. 15–36.

[14] S. Bhunia and M. Tehranipoor, *Hardware Security: A Hands-on Learning Approach*. Morgan Kaufmann, 2018.

[15] U. Guin, D. DiMase, and M. Tehranipoor, "Counterfeit integrated circuits: Detection, avoidance, and the challenges ahead," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 30, no. 1, pp. 9–23, 2014.

[16] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[17] S. Adee, "The hunt for the kill switch," *iEEE SpEctrum*, vol. 45, no. 5, pp. 34–39, 2008.

[18] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2012, pp. 23–40.

[19] F. Koushanfar and G. Qu, "Hardware metering," in *Proceedings of Design Automation Conference*, 2001, pp. 490–493.

[20] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of Annual Design Automation Conference*, 2012, pp. 83–89.

[21] E. Charbon, "Hierarchical watermarking in IC design," in *Proc. of the Custom Integrated Circuits*, 1998, pp. 295–298.

[22] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Constraint-based watermarking techniques for design IP protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, pp. 1236–1252, 2001.

[23] G. Qu and M. Potkonjak, *Intellectual property protection in VLSI designs: theory and practice*. Springer Science & Business Media, 2007.

[24] W. Wang, U. Guin, and A. Singh, "A zero-cost detection approach for recycled ics using scan architecture," in *2020 IEEE 38th VLSI Test Symposium (VTS)*. IEEE, 2020, pp. 1–6.

[25] U. Guin, W. Wang, C. Harper, and A. D. Singh, "Detecting recycled socs by exploiting aging induced biases in memory cells," in *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2019, pp. 72–80.

[26] M. Alam, S. Chowdhury, M. M. Tehranipoor, and U. Guin, "Robust, low-cost, and accurate detection of recycled ics using digital signatures," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2018, pp. 209–214.

[27] P. Cui, J. Dixon, U. Guin, and D. Dimase, "A blockchain-based framework for supply chain provenance," *IEEE Access*, vol. 7, pp. 157 113–157 125, 2019.

[28] A. Stern, D. Mehta, S. Tajik, U. Guin, F. Farahmandi, and M. Tehranipoor, "SPARTA: Laser Probing Approach for Sequential Trojan Detection in COTS Integrated Circuits," *IEEE International Conference on Physical Assurance and Inspection of Electronics (PAINE)*, 2020.

[29] Z. Zhou, U. Guin, and V. D. Agrawal, "Modeling and test generation for combinational hardware trojans," in *VLSI Test Symposium (VTS)*, 2018, pp. 1–6.

[30] N. Vashistha, H. Lu, Q. Shi, M. T. Rahman, H. Shen, D. L. Woodard, N. Asadizanjani, and M. Tehranipoor, "Trojan scanner: Detecting hardware trojans with rapid sem imaging combined with image processing and machine learning," in *Proc. Int. Symposium for Testing and Failure Analysis*, 2018, p. 256.

[31] H. Salmani, "Cotd: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 338–350, 2016.

[32] A. Waksman, M. Suozzo, and S. Sethumadhavan, "Fanci: identification of stealthy malicious logic using boolean functional analysis," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 697–708.

[33] J. Zhang, F. Yuan, L. Wei, Y. Liu, and Q. Xu, "Veritrust: Verification for hardware trust," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1148–1161, 2015.

[34] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.

[35] K. Shamsi, M. Li, K. Plaks, S. Fazzari, D. Z. Pan, and Y. Jin, "IP Protection and Supply Chain Security through Logic Obfuscation: A Systematic Overview," *Trans. on Design Automation of Electronic Systems (TODAES)*, p. 65, 2019.

[36] X. Wang, D. Zhang, M. He, D. Su, and M. Tehranipoor, "Secure Scan and Test Using Obfuscation Throughout Supply Chain," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1867–1880, Sep. 2018.

[37] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly stripping functionality for logic locking: A fault-based perspective," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[38] U. Guin, Z. Zhou, and A. Singh, "Robust design-for-security architecture for enabling trust in IC manufacturing and test," *Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 5, pp. 818–830, 2018.

[39] R. Karmakar, S. Chatopadhyay, and R. Kapur, "Encrypt flip-flop: A novel logic encryption technique for sequential circuits," *arXiv preprint arXiv:1801.04961*, 2018.

[40] S. Potluri, A. Kumar, and A. Aysu, "SeqL: SAT-attack Resilient Sequential Locking," 2020.

[41] H.-Y. Chiang, Y.-C. Chen, D.-X. Ji, X.-M. Yang, C.-C. Lin, and C.-Y. Wang, "LOOPLock: LOgic OPtimization based Cyclic Logic Locking," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[42] K. Juretus and I. Savidis, "Increasing the SAT Attack Resiliency of In-Cone Logic Locking," in *International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[43] M. T. Rahman, S. Tajik, M. S. Rahman, M. Tehranipoor, and N. Asadizanjani, "The key is left under the mat: On the inappropriate security assumption of logic locking schemes," in *Conference on IEEE Int. Sym. on Hardware Oriented Security and Trust (HOST)*, 2020.

[44] A. Jain, Z. Zhou, and U. Guin, "TAAL: Tampering Attack on Any Key-based Logic Locked Circuits," *arXiv preprint arXiv:1909.07426*, 2019.

[45] Y. Zhang, P. Cui, Z. Zhou, and U. Guin, "TGA: An Oracle-less and Topology-Guided Attack on Logic Locking," in *Proc. of the ACM Workshop on Attacks and Solutions in Hardware Security Workshop (ASHES)*, 2019, pp. 75–83.

[46] A. Jain, U. Guin, M. T. Rahman, N. Asadizanjani, D. Duvalsaint, and R. D. S. Blanton, "Special Session: Novel Attacks on Logic-Locking," in *VLSI Test Symposium (VTS)*, 2020.

[47] S. Skorobogatov, "Optical fault masking attacks," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2010, pp. 23–29.

[48] G. Canivet, P. Maistri, R. Leveugle, J. Clédière, F. Valette, and M. Renaudin, "Glitch and laser fault attacks onto a secure AES implementation on a SRAM-based FPGA," *Journal of cryptology*, pp. 247–268, 2011.

[49] V. Pouget, A. Douin, D. Lewis, P. Fouillat, G. Foucard, P. Peronnard, V. Maingot, J. Ferron, L. Anghel, R. Leveugle *et al.*, "Tools and methodology development for pulsed laser fault injection in SRAM-based FPGAs," in *Latin-American Test Workshop (LATW).*, 2007.

[50] B. Selmke, J. Heyszl, and G. Sigl, "Attack on a DFA protected AES by simultaneous laser fault injections," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2016, pp. 36–46.

[51] F. Zhang, Y. Zhang, H. Jiang, X. Zhu, S. Bhasin, X. Zhao, Z. Liu, D. Gu, and K. Ren, "Persistent Fault Attack in Practice," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 172–195, 2020.

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

[53] A. Jain, M. T. Rahman, and U. Guin, "Atpg-guided fault injection attacks on logic locking," in *Conference on IEEE Int. Conf. on Phy. Assurance and Inspection of Electronics (PAINE)*, 2020.

[54] D. Bryan, "The ISCAS'85 benchmark circuits and netlist format," *North Carolina State University*, vol. 25, 1985.

[55] H. Salmani and M. Tehranipoor, "Trust-hub," 2018, [Online]. Available: https://trust-hub.org/home.

[56] DC Ultra: Concurrent Timing, Area, Power, and Test Optimization, "[online] available at: https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html," 2019.

[57] Synopsys Inc., Mountain View, CA, USA, "TetraMAX ATPG: Automatic Test Pattern Generation," 2017.

[58] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1411–1424, 2015.

[59] M. Yasin and O. Sinanoglu, "Transforming between logic locking and IC camouflaging," in *International Design & Test Symposium (IDT)*, 2015, pp. 1–4.

[60] B. Liu and B. Wang, "Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–6.

[61] Q. Yu, J. Dofe, and Z. Zhang, "Exploiting hardware obfuscation methods to prevent and detect hardware trojans," in *International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 819–822.

[62] A. Sengupta and S. P. Mohanty, "Functional obfuscation of DSP cores using robust logic locking and encryption," in *Computer Society Annual Symposium on VLSI (ISVLSI)*, 2018, pp. 709–713.

[63] Q. Yu, J. Dofe, Z. Zhang, and S. Kramer, "Hardware Obfuscation Methods for Hardware Trojan Prevention and Detection," in *The Hardware Trojan War*. Springer, 2018, pp. 291–325.

[64] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," in *International Conference on Cryptographic Hardware and Embedded Systems*, 2016, pp. 127–146.

[65] M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2016, pp. 236–241.

[66] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. J. Rajendran, "What to lock?: Functional and parametric locking," in *Proc. of Great Lakes Symposium on VLSI*, 2017, pp. 351–356.

[67] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1601–1618.

[68] S. M. Plaza and I. L. Markov, "Solving the third-shift problem in IC piracy with test-aware logic locking," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 961–971, 2015.

[69] Y.-W. Lee and N. A. Touba, "Improving logic obfuscation via logic cone analysis," in *Latin-American Test Symposium (LATS)*, 2015, pp. 1–6.

[70] S. Khaleghi, K. Da Zhao, and W. Rao, "IC piracy prevention via design withholding and entanglement," in *Asia and South Pacific Design Automation Conference*, 2015, pp. 821–826.

[71] M. Bushnell and V. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Springer Science & Business Media, 2004, vol. 17.

[72] Y. Xie and A. Srivastava, "Anti-sat: Mitigating sat attack on logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 199–207, 2019.

[73] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic obfuscation for creating sat-unresolvable circuits," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, 2017, pp. 173–178.

[74] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks," in *Int. Conf. on Cryptographic Hardware and Embedded Systems*, 2017.

[75] Y. Shen and H. Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," in *Proceedings of the Great Lakes Symposium on VLSI*, 2017, pp. 179–184.

[76] D. Duvalsaint, X. Jin, B. Niewenhuis, and R. Blanton, "Characterization of Locked Combinational Circuits via ATPG," in *2019 IEEE International Test Conference (ITC)*, 2019, pp. 1–10.

[77] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *Int. Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 95–100.

[78] H. Zhou, R. Jiang, and S. Kong, "CycSAT: SAT-based attack on cyclic logic encryptions," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 49–56.

[79] D. Sirone and P. Subramanyan, "Functional Analysis Attacks on Logic Locking," *arXiv preprint arXiv:1811.12088*, 2018.

[80] K. Juretus and I. Savidis, "Increased Output Corruption and Structural Attack Resiliency for SAT Attack Secure Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[81] M. S. Rahman, A. Nahiyan, S. Amir, F. Rahman, F. Farahmandi, D. Forte, and M. Tehranipoor, "Dynamically obfuscated scan chain to resist oracle-guided attacks on logic locked design," 2019.

[82] S. Potluri, A. Aysu, and A. Kumar, "SeqL: Secure Scan-Locking for IP Protection," *arXiv preprint arXiv:2005.13032*, 2020.

[83] J. Sweeney, M. Zackriya V, S. Pagliarini, and L. Pileggi, "Latch-Based Logic Locking," *arXiv preprint arXiv:2005.10649*, 2020.

[84] UF/FICS Hardware De-obfuscation Competition, https://trust-hub.org/competitions/hwobfuscation1, 2019.

[85] L. Alrahis, M. Yasin, N. Limaye, H. Saleh, B. Mohammad, M. Alqutayri, and O. Sinanoglu, "ScanSAT: Unlocking Static and Dynamic Scan Obfuscation," *Transactions on Emerging Topics in Computing*, 2019.

[86] K. Shamsi, M. Li, D. Z. Pan, and Y. Jin, "KC2: Key-condition crunching for fast sequential circuit deobfuscation," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019, pp. 534–539.

[87] L. Alrahis, M. Yasin, N. Limaye, H. Saleh, B. Mohammad, M. Alqutayri, and O. Sinanoglu, "ScanSAT: Unlocking Static and Dynamic Scan Obfuscation," *IEEE Transactions on Emerging Topics in Computing*, 2019.

[88] A. Jain, U. Guin, M. T. Rahman, N. Asadizanjani, D. Duvalsaint, and R. S. Blanton, "Special Session: Novel Attacks on Logic-Locking," in *2020 IEEE 38th VLSI Test Symposium (VTS)*, 2020, pp. 1–10.

[89] H. Wang, D. Forte, M. M. Tehranipoor, and Q. Shi, "Probing attacks on integrated circuits: Challenges and research opportunities," *IEEE Design & Test*, vol. 34, no. 5, pp. 63–71, 2017.

[90] M. Rahman, S. Tajik, M. Rahman, M. Tehranipoor, and N. Asadizanjani, "The key is left under the mat: On the inappropriate security assumption of logic locking schemes," Tech. Rep., 2019.

[91] H. Wang, Q. Shi, D. Forte, and M. M. Tehranipoor, "Probing Assessment Framework and Evaluation of Antiprobing Solutions," *Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 6, pp. 1239–1252, 2019.

[92] H. Shen, N. Asadizanjani, M. Tehranipoor, and D. Forte, "Nanopyramid: An Optical Scrambler Against Backside Probing Attacks," in *Proc. Int. Symposium for Testing and Failure Analysis(ISTFA)*, 2018, p. 280.

[93] Y. Zhang, A. Jain, P. Cui, Z. Zhou, and U. Guin, "A novel topology-guided attack and its countermeasure towards secure logic locking," *arXiv preprint arXiv:2006.05930*, 2020.

[94] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Int. conf. on the theory and applications of cryptographic techniques*, 1997, pp. 37–51.

[95] G. Piret and J.-J. Quisquater, "A differential fault attack technique against SPN structures, with application to the AES and KHAZAD," in *Int. workshop on cryptographic hardware and embedded systems*, 2003, pp. 77–88.

[96] P. Dusart, G. Letourneux, and O. Vivolo, "Differential fault analysis on AES," in *Int. Conf. on Applied Cryptography and Network Security*, 2003, pp. 293–306.

[97] T. Fukunaga and J. Takahashi, "Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2009, pp. 84–92.

[98] N. Selmane, S. Guilley, and J.-L. Danger, "Practical setup time violation attacks on AES," in *Seventh European Dependable Computing Conference*, 2008, pp. 91–96.

[99] S. Guilley, L. Sauvage, J.-L. Danger, N. Selmane, and R. Pacalet, "Silicon-level solutions to counteract passive and active attacks," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2008, pp. 3–17.

[100] A. Barenghi, G. M. Bertoni, L. Breveglieri, and G. Pelosi, "A fault induction technique based on voltage underfeeding with application to attacks against AES and RSA," *Journal of Systems and Software*, pp. 1864–1878, 2013.

[101] A. Barenghi, G. M. Bertoni, L. Breveglieri, M. Pellicioli, and G. Pelosi, "Low voltage fault attacks to AES," in *Int. Sym. on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 7–12.

[102] J.-M. Schmidt and M. Hutter, *Optical and EM fault-attacks on CRT-based RSA: Concrete results*, 2007.

[103] A. Dehbaoui, J.-M. Dutertre, B. Robisson, and A. Tria, "Electromagnetic transient faults injection on a hardware and a software implementations of AES," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2012, pp. 7–15.

[104] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, pp. 3056–3076, 2012.

[105] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *Int. workshop on cryptographic hardware and embedded systems*, 2002, pp. 2–12.

[106] R. Torrance and D. James, "The state-of-the-art in IC reverse engineering," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2009, pp. 363–381.

[107] H. Wu, D. Ferranti, and L. Stern, "Precise nanofabrication with multiple ion beams for advanced circuit edit," *Microelectronics Reliability*, pp. 1779–1784, 2014.

[108] D. Skarin, R. Barbosa, and J. Karlsson, "GOOFI-2: A tool for experimental dependability assessment," in *IEEE/IFIP Int. Conf. on Dependable Systems & Networks (DSN)*, 2010, pp. 557–562.

[109] T. Tsai and R. Iyer, "FTAPE-A fault injection tool to measure fault tolerance," in *Computing in Aerospace Conference*, 1995, p. 1041.

[110] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, "Fault injection techniques and tools," *Computer*, pp. 75–82, 1997.

[111] M. T. Rahman, M. S. Rahman, H. Wang, S. Tajik, W. Khalil, F. Farahmandi, D. Forte, N. Asadizanjani, and M. Tehranipoor, "Defense-in-depth: A recipe for logic locking to prevail," *Integration*, 2020.

[112] C. H. Kim and J.-J. Quisquater, "Faults, injection methods, and fault attacks," *Design & Test of Computers*, vol. 24, no. 6, pp. 544–545, 2007.

[113] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz, "Electromagnetic fault injection: towards a fault model on a 32-bit microcontroller," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2013, pp. 77–88.

[114] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte, "Ram-jam: Remote temperature and voltage fault attack on fpgas using memory collisions," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2019, pp. 48–55.

[115] S. Tajik, H. Lohrke, F. Ganji, J.-P. Seifert, and C. Boit, "Laser fault attack on physically unclonable functions," in *Workshop on fault diagnosis and tolerance in cryptography (FDTC)*, 2015, pp. 85–96.

[116] C. Champeix, N. Borrel, J.-M. Dutertre, B. Robisson, M. Lisart, and A. Sarafianos, "Seu sensitivity and modeling using pico-second pulsed laser stimulation of a d flip-flop in 40 nm cmos technology," in *International symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFTS)*, 2015, pp. 177–182.

[117] F. Stellari, C.-C. Lin, T. Wassick, T. Shaw, and P. Song, "Automated contactless defect analysis technique using computer vision," in *Proceedings from the International Symposium for Testing and Failure Analysis (ISTFA)*, 2018, p. 79.

[118] M. T. Rahman and N. Asadizanjani, "Backside security assessment of modern socs," in *International Workshop on Microprocessor/SoC Test, Security and Verification (MTV)*, 2019, pp. 18–24.

[119] J. A. Roy, F. Koushanfar, and I. L. Markov, "Ending piracy of integrated circuits," *Computer*, pp. 30–38, 2010.

[120] A. Sengupta, M. Ashraf, M. Nabeel, and O. Sinanoglu, "Customized locking of IP blocks on a multi-million-gate SoC," in *Int. Conf. on Computer-Aided Design (ICCAD)*, 2018, pp. 1–7.

[121] L. Bu and M. A. Kinsy, "Hardening AES Hardware Implementations Against Fault and Error Inject Attacks," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 2018, pp. 499–502.

[122] M. Doulcier-Verdier, J.-M. Dutertre, J. Fournier, J.-B. Rigaud, B. Robisson, and A. Tria, "A side-channel and fault-attack resistant aes circuit working on duplicated complemented values," in *2011 IEEE International Solid-State Circuits Conference*. IEEE, 2011, pp. 274–276.

[123] K. Matsuda, T. Fujii, N. Shoji, T. Sugawara, K. Sakiyama, Y.-i. Hayashi, M. Nagata, and N. Miura, "A 286 f 2/cell distributed bulk-current sensor and secure flush code eraser against laser fault injection attack on cryptographic processor," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 11, pp. 3174–3182, 2018.

[124] K. Matsuda, N. Miura, M. Nagata, Y.-i. Hayashi, T. Fujii, and K. Sakiyama, "On-chip substrate-bounce monitoring for laser-fault countermeasure," in *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*. IEEE, 2016, pp. 1–6.

[125] T. Sugawara, N. Shoji, K. Sakiyama, K. Matsuda, N. Miura, and M. Nagata, "Side-channel leakage from sensor-based countermeasures against fault injection attack," *Microelectronics Journal*, vol. 90, pp. 63–71, 2019.

[126] Y. Li, R. Hatano, S. Tada, K. Matsuda, N. Miura, T. Sugawara, and K. Sakiyama, "Side-channel leakage of alarm signal for a bulk-current-based laser sensor," in *International Conference on Information Security and Cryptology*. Springer, 2019, pp. 346–361.