2020

Provably Trustworthy and Secure Hardware Design with Low Overhead

Qutaiba Alasad
University of Central Florida

PROVABLY TRUSTWORTHY AND SECURE HARDWARE DESIGN WITH LOW OVERHEAD

by

QUTAIBA ALASAD

B.S. University of Mosul, 2007
M.S. University of Mosul, 2009

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2020

Major Professor: Jiann S. Yuan

ABSTRACT

Due to the globalization of IC design in semiconductor industry and outsourcing of chip manufacturing, 3PIPs become vulnerable to IP piracy, reverse engineering, counterfeit IC, and hardware Trojans. To thwart such attacks, ICs can be protected using logic encryption techniques. However, strong resilient techniques incur significant overheads. SCAs further complicate matters by introducing potential attacks post fabrication. One of the most severe SCAs is PA attacks, in which an attacker can observe the power variations of the device and analyze them to extract the secret key. PA attacks can be mitigated via adding large extra hardware; however, the overheads of such solutions can render them impractical, especially when there are power and area constraints. In our first approach, we present two techniques to prevent normal attacks. The first one is based on inserting MUX equal to half/full of the output bit number. In the second technique, we first design PLGs using SiNW FETs and then replace some logic gates in the original design with their SiNW FETs-based PLGs counterparts. In our second approach, we use SiNW FETs to produce obfuscated ICs that are resistant to advanced reverse engineering attacks. Our method is based on design a small block, whose output is untraceable, namely URSAT. Since URSAT may not offer very strong resilience against the combined AppSAT-removal attack, S-URSAT is achieved using only CMOS-logic gates, and this increases the security level of the design to robustly thwart all existing attacks. In our third topic, we present the usage of ASLD to produce secure and resilient circuits that withstand IC attacks (during the fabrication) and PA attacks (after fabrication). First, we show that ASLD has unique features that can be used to prevent PA and IC attacks. In our three topics, we evaluate each design based on performance overheads and security guarantees.

Dedicated to my dearest family and friends.

ACKNOWLEDGMENTS

In the Name of Allah the Most Merciful, the Most Compassionate. First, I would like to start thanks my advisor, Prof. Jiann S. Yuan, for his feedback, support, and guidance that help me fulfill my dissertation. During my studies, Prof. Yuan has supporting me a lot in creating and improving ideas, writing papers, answering reviewer questions. I have understood too many things, not only in research area, but also in my life to withstand in all situations. He has always supported and provided me with the great lab resources and software tools to achieve my work. His great experiences were very discerning to complete this dissertation.

Also, I would like to thank my four committee members: Dr. Murat Yuksel, Dr. Mingjie Lin, Dr. Amro Awad, and Dr. Zixia Song for spending times to attend my dissertation defense, reading the previous draft of the dissertation, and providing important feedback that refined and enhanced the dissertation. I would also like to thank Diana Camerino for her great help throughout my plan of study.

Next, I would like to admit my sponsor, The Higher Committee For Education Development (HCED) in Iraq, for their financial support. With this financial support, I was able to complete the Ph.D. program requirements.

Lastly, my sincere thanks are to my family, specifically my mother, father, wife , and friends, for their help and support that allow me to reach my dream. I would also like to thank all my collaborations. Dr. Yu Bi who help me understand the fundamental of the silicon nanowire, Dr. Deliang Fan who provided me useful resources to implement and simulate Spitronic devices, and Dr. Jie Lin.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Nowadays, the advances in field programmable gate array (FPGA), microprocessors, multimedia, integrated circuits, Internet of Things (IOT), System on Chip (SoC), and digital signal processing (DSP) have resulted in much more complicated circuits with billions of transistors. This one allows companies to add much more features on a single silicon chip that help human beings complete many tasks easily. CMOS scaling enable foundries to produce devices with small size, ultra-low power, high performance, high density, super large memory storage. Unfortunately, The ubiquity of the sophisticated technologies with CMOS technology scale-down leads to the potential opportunities for the malicious user, because a more intricate and larger system makes it more difficult to either detect or prevent hidden malicious hardware Trojans during the test and/or normal operation [5–7]. Traditional testing tools, e.g.Automatic Test Pattern Generation (ATPG), can not guarantee detecting malicious threats, especially for a very large scale design since generating all test patterns are not possible. Therefore, it is very important to make sure that hardware chips are secure regardless of the chip sizes.

Hardware attacks are not limited to the malicious user. Many kinds of serious hardware attacks have been introduced last decade. Examples of such severe attacks are intellectual property (IP) piracy, reverse engineering, integrated circuit (IC) counterfeiting, and IC overbuilding [8]. Unlike software attacks, hardware attacks are hard to be prevented or detected. Hardware security is a technique to protect the physical hardware device that the software is installed on it. As a consequence, securing the hardware becomes mandatory and is not less important than securing the software because if the hardware is not trustworthy, then the software is also not trustworthy [9]. This dissertation will concentrate on proposing techniques that can thwart serious hardware attacks with low overhead and finally make the globalization of IC design flow secure against various serious attacks.

1.1 The Globalization of IC Design Flow

Integrated circuit (IC) design is a part of electronics engineering, including logic and circuit design methods that are required to create ICs with a minimum number of semiconductor electronic components. Examples of the most preliminary electronic components are resistors, capacitors, light-emitting diodes (LEDs), transistors, and inductors. Currently, modern ICs become extremely complicated and contain many billions of transistors. The main purpose of producing a complex IC is to involve as many applications as possible on the same chip so the product can be used to perform many functions. Unfortunately, such complex IC cannot be tested and validated easily during the fabrication process. To solve this dilemma, companies use a special procedure. Figure 1.1 (a) shows the main steps of this procedure [10]. These steps are called "IC design flow".

IC design flow starts from the idea of the chip, specifications, and product requirements all the way down to the chip manufacturing. Typically, the first step is to represent the idea of the chip by logic gates, namely netlist. Each netlist has many different logic gates that are connected through wires (interconnections), and each type of the logic gates implements a different Boolean function [11]. The next step is the logic synthesis, which is used to convert the high level of the design description to the optimized gate-level. Then, the gate-level design is transferred to layout (GDS II), where all design components are mapped to geometric representation of wires, capacitors, transistors, and so on. The layout is then given to the manufacturer to fabricate the design. Afterward, the individual components of the chip are tested and packaged to make sure there are no errors and isolate the defective parts. The chip is produced by putting the individual components together, and then the whole chip is tested by the facility at the IC configuration department. If all the aforementioned steps are passed successfully, then the IC is ready to be sold in the market [12, 13]. This traditional procedure is good only for companies that have their own fabrication equipment, e.g. Intel and IBM.

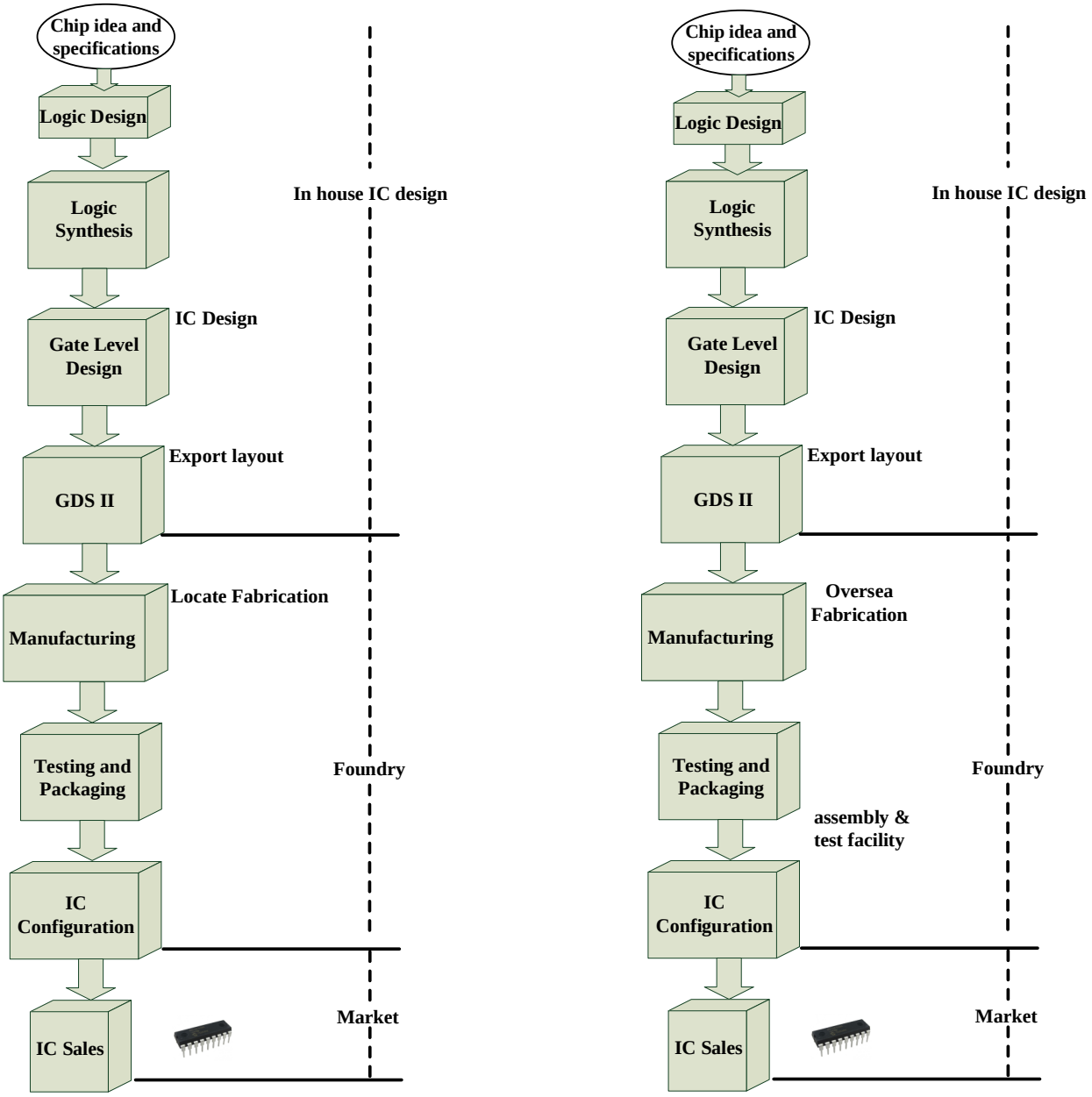


Figure 1.1: Main processing steps in IC design flow: (a) conventional IC design flow, and (b) globalization of IC design flow. Due to the offshore IC fabrication, IP owners do not have full control over untrusted foundries to protect their designs.

Unfortunately, since the complexity of the design becomes very large and needs modern manufacturing equipment, several fabless companies, e.g. Apple, only design the chips and then outsource the chips to semiconductor foundry for fabrication to reduce the fabricating costs [14, 15]. Thus, ICs might be imitated by an untrusted foundry or reversed engineering to be sold in markets illegally. Consequently, IP owners do not have full control over untrusted foundries to protect their designs, where they are susceptible in front of many attacks, such as reverse engineering, IP piracy, and IC counterfeiting [16]. Figure 1.1 (b) shows the steps of the globalization of IC design flow.

1.2 Hardware Security Attacks

Since the ICs are sent to oversea for fabrication purposes, they are subject to many attacks [1, 17, 18]. Attack requirements are different based on what the attacker wants to achieve. We will talk about each one, as follows:

1.2.1 Overproduction

An attacker in an untrusted foundry can illegally produce extra copies of a chip without knowledge of designers or IP owners. Usually, these extra copies are manufactured out of the contract signed between the IP owner and foundry. Since there is no previous cost of research and development on the chip, attacker can earn more money than IP owner by selling the same chips [19–25].

1.2.2 Counterfeiting

Counterfeit hardware devices are exact copies of the genuine devices but with low quality. Such devices are usually made from electronic parts that are recycled or no longer useful. Counterfeit

ICs become the main challenge in academic research and many commercial products due to being unreliable and insecure [18, 26]. Unfortunately, only a few countries (e.g., the USA and Japan) have a rigorous set of rules and regulations to prevent IC piracy from exposing or stealing electronic products [27–29]. A report from Information Handling Services (IHS) company in the USA showed that the untrusted ICs and the counterfeit chips have increased four-fold since 2009 [30]. The Group of Twenty (G-20) estimated the impact of IC piracy and counterfeiting reaches $1770 billion in 2015 [31].

1.2.3 Hardware Trojan

An attacker either in the design house or in the untrusted foundry may inject malicious circuits, namely hardware Trojans, into the original hardware design that cannot be detected easily during the design testing or normal operation [32]. Traditionally, there are two taxonomies of hardware Trojans; combinational and sequential. The purpose of inserting a Trojan is to cause a malfunction in the chip at a specific IC operation and/or leak sensitive information [33,34]. Each Trojan consists of two parts; trigger and payload. The trigger is a circuit that employs to activate in very rare conditions while the payload considers as a response or an act to the Trojan. Trojans can be inserted in various phases of IC design flow, such as design, testing, packaging, synthesis, verification, and fabrication. Different techniques have been proposed to prevent such threat [35]. However, Trojan detection/prevention methods are not guaranteed, especially if the Trojan is inserted into a large scale circuit.

1.2.4 Reverse Engineering

Reverse engineering is a process in which a design can be analyzed to see how it works in order to duplicate/improve the system. It comprises of, but not limited to, extracting the design netlist [36],

distinguishing the leveraged technology [37], recovering the functionality [38], getting layer images, and depackaging and delayering the IC [39]. Regrettably, exposing such information during the process may allow attackers to recover the design of the chip. The target is different depending on what the attacker wants to do. For example, if the aim of the attacker is to unauthorizedly duplicate the design, the target will be either gate-level or physical design. If the goal is to insert a Trojan, then the target will be RTL or gate-level netlist.

1.2.5 IP Piracy

An attacker can reverse engineer the IP design and reproduce the chips. IP piracy is one of the most severe attacks, in which an unauthorized person can duplicate the IP cores or exploits it without having permission from the IP owner in order for him/her to sell them illegitimately in the market as authenticities. Such threats can cause significant financial losses to the semiconductor industry [40].

1.2.6 Side Channel Attacks

The secret key in a smart device can be revealed by side channel attacks (SCAs) through observing the power dissipation [41], photonic emissions [42], electromagnetic emissions [43], or time analysis [44]. Power analysis attack (PAA) is more popular and powerful than others. The first PAA was presented by Kocher et al. [45]. There are three types of PAA; simple power analysis (the data are directly taken from the power dissipation), differential power analysis (the related information is obtained from large number power traces, and then statistical analysis is used to identify the differences in these traces), and correlation power analysis (CPA) (the real power is correlated with the predicted power model to get the leakage data). CPA is more advanced and faster than others [46]. Another combined attacks, namely fault-based side-channel attacks, using

active and passive SCAs, have gained much attention. The idea of this attack is to inject a fault into the hardware device to generate errors, in which faulty computations will reveal information about the implemented device [47–50]. Note that this attack is more efficient and requires lower computations to leak secret information than SCA.

1.3 Hardware Attack Prevention Techniques

The vulnerability of chip security during manufacturing spurs the research on countermeasure methods. There are several techniques have been proposed to thwart hardware attacks. In this subsection, we will briefly discuss and show the strengths and weaknesses of each of these methods.

1.3.1 Metering

Hardware metering technique is a collection of protocols that help the IP owners control their designs remotely during the post-fabrication stage [51, 52]. Metering could be either active (allow the designer to enable or disable the chip) or passive method (allow the designer to monitor their design passively through using physical identification, e.g. serial number) [19].

1.3.2 Split Manufacturing

The main idea of split manufacturing is to partition the circuit into two parts. The first part has the main operation components (transistors, capacitors and so on) with few wires, called front-end of line (FEOL), and the second one has the rest of the interconnection, namely the back-end of line (BEOL). These two parts are fabricated in different foundries and then connected together [53–55]. Broadly, split manufacturing process depends on 2D integration [56], 2.5D integration [57], and

3D integration [58]. It is a good technique to prohibit IP piracy in an untrusted foundry.

1.3.3 Watermarking

Hardware watermarking is a signature of the IP owner, e.g., a unique IP owner constraint, that is implanted into the design. Watermarking is a passive method that can only be used to detect IP piracy but not prevent it [59]. Moreover, it can be implemented at the logic design, logic synthesis, or physical design phase.

1.3.4 Camouflaging

Camouflaging is proposed to prevent reverse engineering attacks by replacing some gates in the original design with corresponding camouflaged counterparts. Camouflaged gates look like the original gates in the netlist but provide different functionalities, e.g., the same structure can be either NAND or NOR gate based on the design configurations. This technique can be achieved by adding dummy contacts [60], using diffusion programmable cells [61] or filler cells [62]. Camouflaging is achieved at the layout-design phase after the IC physical design. Therefore, it should be supported by a trusted fabrication company to manufacture the camouflaged gates [63].

1.3.5 Power Analysis Attack Countermeasures

There are many power analysis attack (PAA) countermeasures, but the most two popular ones are the masking [64] and hiding [65] techniques. Masking method targets to randomly mask the intermediate values, which requires additional hardware resources, where large overhead requires for masking complex operations. For hiding method, even though the performance penalty is not that much large for different types of operations, there is a trade-off between the security and power

dissipation. Increasing the noise power leads to increase the protection of the design, however, the signal to noise ratio will be reduced since the noise will control the power [66]. Employing a Sensing Amplifier Based Logic (SABL) [67] is more empirical technique to prevent PAA. A protected design, namely Current Mode Logic (CML), is employed to produce approximately a constant power dissipation at the output and thus PAA cannot identify the correct key. The above-mentioned techniques require large overhead for high protection level, which is not good for IoT applications since most of IoT devices are powered by battery [45].

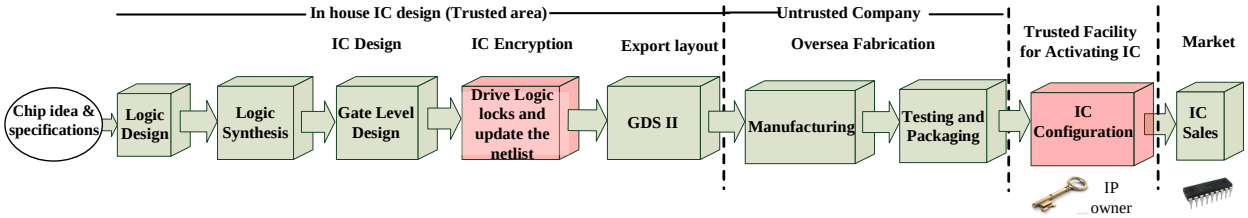


Figure 1.2: The modifications in IC design flow when a logic encryption technique is used to protect the design. The two red regions donate the two locations in which the IC get encrypted and activated by IP owners

1.3.6 Logic Encryption

Logic encryption locks a netlist through adding logic gates into the original netlist with a valid key. These added logic gates, namely key-gates, are inserted into the original netlist at different locations based on the algorithm that the designer uses (more details regarding logic encryption algorithms are mentioned in chapter 2). The key-gates could be XOR/XNOR gates [68], MUXes [26,69], OR/AND gates [70], or a collection of these elements [71]. The functionality of the design will not reveal unless the correct key is given. Logic encryption is considered as one of the best techniques since it provides a fully protection against hardware attacks. Also, it presumes that all processing steps in IC design flow are not trusted, except the design house. Figure 1.2 presents

the IC design flow combined with the logic encryption technique. The Figure demonstrates where the IC should be encrypted and activated. Instead of shipping the original netlist to the offshore manufacturing foundry, a logic-gate level encryption technique is applied to protect the IP design with low cost. Afterwards, the encrypted netlist is converted to Graphic Database System (GDS II) stream format and then sent to a foundry for fabrication. Once the encrypted circuits successfully pass the untrusted foundry, the key inputs should be driven by on-chip tamper-proof memory [72], as shown in Figure 1.3. In other words, after retrieving the fabricated chips, in order to exhibit its correct functionality (produce correct outputs), the valid key has to be supplied to the encrypted design, which is done by a trusted facility to active the IC, for a certified IP owner to unlock the chips. However, upon employing a wrong key, the encrypted design will show the wrong functionality (produce wrong outputs). It is worth mentioning that the embedded secret key of the encrypted circuit, after the fabrication, may be revealed by power analysis (PA) attacks [41]. Therefore, extra hardware resources should be added to mitigate PA attacks, more details regarding this matter is discussed in Chapter 2. In the following, we clarify how logic encryption can protect ICs from various IC attacks.

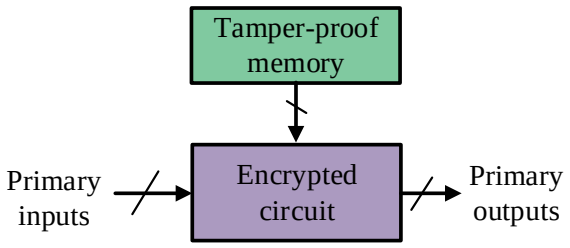


Figure 1.3: A block diagram of the logic encryption representation

Hardware Trojans: A malicious insider may illegally pirate the IP core without the permission of the designer and add a hardware Trojan into the design. Logic encryption can prevent an attacker from inserting a Trojan by spurring the rare signals in the design and inserting key-gates at these

rare signals. There are many ways to find these rare signals, but the most popular and efficient one is the Monte Carlo algorithm [73].

Reverse Engineering and IP Piracy: A malicious company can steal or pirate the encrypted design by IC reverse engineering, but extracting the design by using reverse engineering is really complicated and needs a lot of time and effort to be achieved. Also, since the secret key of the encrypted design is unknown to the malicious company, the design will not function correctly. Traditional logic encryption can easily prevent such attacks; however, if an attacker in an untrusted company can get an activated IC from the market, then he or she can apply input patterns to the activated IC and get the corresponding output patterns. By accessing the gate level of the encrypted IC (obtained by reverse engineering) and observing the corresponding outputs, the attacker can find the correct key of the encrypted circuit with a shortened time [74] [2]. We name these attacks as *advanced reverse engineering attacks* and will discuss more in detail about these attacks and the ways to prevent them in chapter 2.

IC Counterfeiting: Logic encryption can mainly be used to prevent IC piracy and many types of IC counterfeiting, such as cloning a design with low quality, that deliberately depend on reverse engineering [18].

Overproduction: An attacker in an untrusted foundry can easily overproduce or overbuild ICs to make extra IC copies without the designer's knowledge. However, the functionality of the encrypted IC is unknown since the IC has not been activated by the secret key yet.

The hardware attack prevention techniques are different based on the attack model and security goal. Table 1.1 gives a recapitulation of comparison among existing hardware attack prevention techniques. The Table ensures that only logic encryption methods can full prevent IC attacks.

Table 1.1: A comparison between different techniques that can be used to prevent IP piracy, IC overproduction, reverse engineering attacks.

Technique	Test facility	Company	Design integration	End user	3PIP seller
Metering [19, 51, 75]	Yes	No	No	Yes	No
Split manufacturing [53, 76]	No	Yes	No	No	No
Watermarking [59, 77, 78]	Yes	No	No	Yes	No
Camouflaging [60, 79–83]	No	No	No	Yes	No
Logic encryption [1, 2, 4, 68, 69, 84–86]	Yes	Yes	Yes	Yes	Yes

1.4 Contributions of the Dissertation

To produce a secure hardware design that can effectively prevent IC attacks during the fabrication, a strong logic encryption technique should be leveraged. Also, a designer needs to add additional hardware resources to make the IC resilient to side-channel attacks (SCAs). Therefore, enormous overheads are required to produce a secure and resilient IC against all hardware attacks during and after the fabrication. Regrettably, the large overhead may preclude the encrypted design from being fabricated on a real silicon chip. Orthogonal to the previous works, we develop various techniques that can offer a robust and secure design with ultra-low overhead compared to previous works. The main contributions of this dissertation are as follows:

1.4.1 Strong Logic Encryption against Normal IC Attacks

1.4.1.1 Approach1: Traditional Encryption Using Multiplexers with LFSR

Random insertion of key-gates based traditional logic encryption cannot provide high security level (50% Hamming distance (HD)) when incorrect keys are entered. Fault impact analysis based logic encryption is an improvement on random insertion techniques; however, it has two limitations: first, the complexity and the difficulty of getting around 50% HD for a design that is large or has

many output bits; second, the huge performance overhead to implement RSA and the reliability of PUF. From these two points, our contribution is described as follows:

- Ensuring that the HD is 50% (best case) once the number of output bits is even or close to 50% for the circuits having convenient output odd bits, for whatever the size of the circuit.
- Leveraging both the hardware Trojan idea and the linear feedback shift register (LFSR) random generation, to generate suitable random keys, for the fully-encrypted design instead of employing both RSA and PUF.

1.4.1.2 Approach 2: Conventional Encryption Using Hybrid CMOS and Emerging SiNW FETs

A state-of-the-art solution for logic obfuscation objectives is to leverage CMOS technology, but the challenge is to obtain a high level of chip protection without a high cost penalty. The required performance overhead for logic encryption purposes can exceed 25% when the number of inserted key-gates (XOR/XNOR) is about 5% of the total number of gates in the combinational International Symposium on Circuits and Systems (ISCAS)-85 benchmark [2]. In order to address this issue, we propose a technique based on the new characteristic of emerging technology for IP protection and hardware attack prevention. More specifically, we introduce the silicon nanowire (SiNW) FET based polymorphic logic gate to help obfuscate the netlist to further improve IP protections. Different from previous efforts, this paper presents an in-depth theoretical analysis and security evaluation with the proposed technique. The details of our contributions are listed as follows:

- We first present the polymorphic logic gate based on emerging SiNW polarity-contrallable FET and its advantages over conventional CMOS technology.

- We then incorporate polymorphic logic gates for encrypting combinational circuits. A polymorphic gate based logic encryption algorithm is further proposed with theoretical analysis.
- We evaluate the proposed SiNW FETs and CMOS hybrid logic encryption, achieving a hamming distance of 50% for most of the ISCAS'85 benchmark circuits.
- The performance penalty of the proposed technique has also been evaluated, where a much smaller overhead is incurred compared to the previous literature. A genuine energy-efficient logic locking is achieved.

1.4.2 Strong Logic Obfuscation against Advanced IC Reverse Engineering Attacks

In this work, we leverage SiNW polymorphic logic gates (PLGs) augmented with a light-weight resilient-SAT circuit to produce an obfuscated design that requires small overhead and is more secure. The contributions of this work are as follows:

1 PLGs designed using emerging SiNW FETs are exchanged with some logic gates in the original circuit, starting from the large gate input size [87]. To further reduce the performance overhead, we exchange PLGs starting from the small gate input size. Unlike the work in [88], *these PLGs are designed to cover/mimic all gate sizes and types in the original netlist for any logic circuit*.

2 We introduce an untraceable light-weight Resilient SAT (URSAT) circuit that provides a defense against SAT attack. URSAT also prevents an adversary from tracking and removing the URSAT output signal, thus avoiding resilient attacks. URSAT is built using different input sizes of AND/OR PLGs [87]. To make URSAT more general, we use 2-input AND/OR PLGs with larger-input AND/NAND PLG at the last stage in URSAT block.

3 Even though URSAT is secure against many attacks, it may not achieve strong resilience against the combined AppSAT-removal attack. We propose S-URSAT designed using only CMOS-logic gates to strongly thwart the combined AppSAT-removal attack. Also, the internal signals and the output signal of S-URSAT are more changeable when incorrect keys are applied, and this increases the security level of the proposal against other attacks.

4 Finally, we evaluate the performance overhead of our designs and show that it has a very small penalty.

1.4.3 Secure and Resilient Hardware Devices Using ASLD

In this work, we present a case study on how All Spin Logic Device (ASLD) can produce secure and resilient circuits against IC attacks during the fabrication and PA attacks after the fabrication with a small penalty. The contributions of this work are as follows:

1 We design simple PLGs that can be used to build up a secure circuit against IC attacks [89].

2 We design new efficient ASLD based logic gates to produce various polymorphic logic gates (PLGs), such as inverter/buffer, AND/OR, NAND/NOR, and XOR/XNOR PLGs, with different input-sizes. Each PLG has a valid key-input. These PLGs are used to produce encrypted circuits by only replacing the traditional CMOS circuit with ASLD based PLGs. The implemented circuit using ASLDs can be encrypted with a very complicate key without any adding extra resources. Therefore, the design is extremely resilient to IC attacks.

3 The new designed ASLD based PLGs require much less energy dissipation and are much faster than our previous implemented PLGs [89]. These improvements are achieved by implementing PLGs with a different structure (XOR/XNOR), employing optimal ASL parameters, using the pipeline technique, and adding ASL-buffers at necessary locations.

4 We extract a new unique feature of ASLD, in which the power dissipation of all read/write operations from/to nanomagnet unit is constant. This new feature of ASLD with the new designed PLGs enables us to implement resilient design against PA attacks with a small performance penalty. To evaluate the strength of the proposed technique, we implement a resilient ASLD-based Advanced Encryption Standard (AES) circuit as an example, where different kinds of ASLD logic gates with different input-sizes are generated, by fixing the key input of the designed PLGs to VDD or GND based on the required gate. Then, we attack the first round in AES with a correlation power analysis to guess the secret key information. The results show that the secret key cannot be revealed by PA attacks.

5 We also evaluate and compare the energy and area of our proposal with 14 nm CMOS technology. The energy dissipation and the required area of the proposal are retrieved using our own developed ASLD library.

1.5 Organization of the Dissertation

The outline of the dissertation is organized as follows: Chapter 1 summarizes the work of the dissertation, including the introduction for hardware security attack and defense methods, and the dissertation contributions. Chapter 2 provides a brief overview of related works that specifically talk about logic encryption attacks and defenses in section 2.2, side channel analysis attacks and defenses in section 2.3, and fundamental of emerging devices in section 2.4. Chapter 3 presents two approaches. In section 3.1, we present approach 1 that ensures producing 50% HD for any input patterns while section 3.2 elucidates approach 2 that produces robust encryption with much lower overhead compare to approach 1. Strong logic obfuscation against advanced IC reverse engineering attacks is given in chapter 4. Chapter 5 presents secure and resilient hardware devices using ASLD. Finally, we conclude in chapter 6.

CHAPTER 2: PRELIMINARIES

2.1 Traditional Logic Encryption

Rogue copy, mask theft, overproduction, and overbuilding are the main reasons to secure the IC from various assailants leveraging the logic encryption approaches. The cipher design can be done via inserting few gates to the original design to conceal the functionality, while the decipher can be rendered once the owner provides the circuit by a correct external key-bit inputs.

Logic encryption can mainly be classified into two types: sequential and combinational. For sequential, new states with transitions have to be added to the original Finite State Machine (FSM) to produce a boosted and security-enhanced FSM that provides a strong secured circuit [19]. In [90] and [91], the authors proposed two modes: obfuscated and normal. By inserting some state elements to the original FSM, the obfuscated mode is produced to create a counterfeit state. The functionality is always wrong unless the correct sequence of the key bits is applied. An untrusted foundry is not able to figure out the correct sequence of the new FSM without knowing the whole state transition graph which means going through all the possible cases. Also, the design goes in an unknown state when the sequence of the key is pulled out. Sequential logic encryption methods are not mature enough compared to combinational logic encryption

Combinational logic encryption techniques are more popular than sequential encryption methods, where most of the recent works concentrate on combinational encryption. In general, traditional combinational encryption techniques were presented based on three different methods: (1) random insertion of key-gates (RIKG) [1, 68], (2) maximization the hamming distance between the incorrect and correct outputs when wrong keys are applied (MHD) [22, 69], and (3) interference graph among the inserted key-gates (IGI) [2]. In RIKG, the design can be encrypted via randomly insert-

ing XOR/XNOR key-gates into the original circuit. Roy et al. [1, 68] proposed a chip-locking system for active IC metering, while targeted to make physical tampering infeasible. The chip-locking framework inserts XOR/XNOR key gates with fan-ins connected to the bits of keys that activate the circuit. The insertion is achieved at randomly selected locations before physical synthesis but after logic synthesis. In MHD, Baumgarten et al. [22] used lookup table-based locking units that hinder attempts to reverse-engineer functionality from the mask prospectives. It demonstrates how logic encryption can be propagated to the field programmable gate array (FPGA) domain. The authors attempt to insert the key-gates in a way that maximizes the relationship between correct and corrupt output patterns once wrong keys are applied. Similarly, Rajendran et al. [92] proposed fault analysis-based logic encryption that formalizes the fault impact of a given netlist and incorporates XOR/XNOR gates at the selected locations. A wrong key ensures the corrupting of output values. A continuing work [69] includes a multiplexer as logic cone for the encryption.

2.2 Analytical IC Reverse Engineering: Threats and Defense Techniques

It is conceivable that an attacker can obtain a functional IC (e.g., by purchasing one from the market) and then provide arbitrary inputs to the IC and observe the corresponding outputs for these inputs. Attackers may also be able to obtain encrypted IC netlist *as a black box* either through reverse engineering or IP theft from an untrusted foundry. Access to a gate-level netlist of the encrypted IC and the ability to query the output for arbitrary input is sufficient to default many logic locking techniques [2, 74].

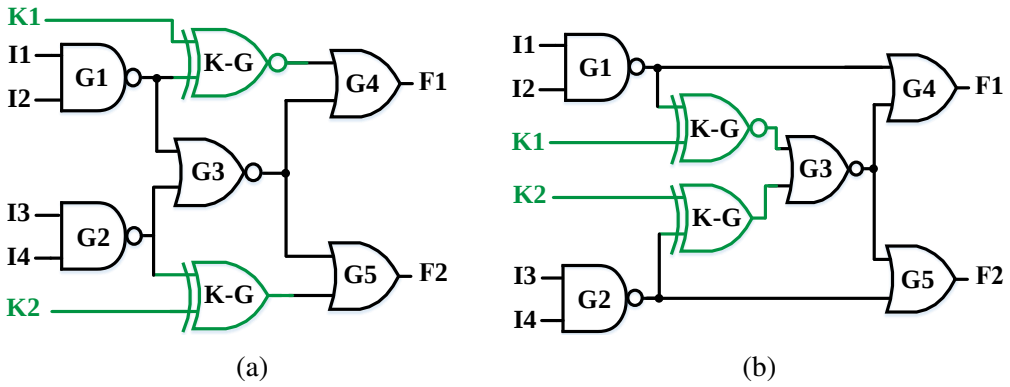


Figure 2.1: logic locking techniques: (a) encryption based on random insertion [1], and (b) strong encryption [2].

2.2.1 Pre-SAT: Sensitization Attack and Defenses

The encrypted IC becomes vulnerable to sensitized key-based attacks if the key-gates are inserted randomly or without interference graph among them, where the key values could be isolated to primary outputs using Automatic Test Pattern Generation (ATPG) tools [2]. For example, in the simple locked design illustrated in Figure 2.1 (a), both K1 and K2 are stored in a secure memory, and an assailant cannot access them. By supplying a specific input pattern, generated by ATPG tools, e.g. ”0X11”, both K1 and K2 values will be exposed on the outputs, F1 and F2, respectively. To prevent such threats, interference graph among the inserted key-gates should be adopted to make the attack execution time exponential, such as incorporating two key-gates to the same gate in the original circuit as illustrated in Figure 2.1 (b). However, this technique requires larger overhead than others, and the complexity of this algorithm is typically high.

2.2.2 SAT and Post-SAT: Attacks and Defenses

SAT attack is more powerful than any other existing threats [74]. The attack applies distinguishing input patterns (DIPs) to an activated chip and observes their corresponding output patterns. Then, at each iteration, SAT attack excludes a single or a bunch of incorrect key-bit(s) from the encrypted chip. After applying all distinguishing input/output (I/O) pairs, one unique key can satisfy all these I/O pairs, which implies that this key (KE) should be the secret key of the encrypted chip.

Adding a cryptography, e.g. AES, to a circuit can prevent any kind of attacks as long as the secret key of the cryptography is long enough ($>$ 64 key-bits). However, the performance overhead will increase significantly. Interestingly, it has been experimentally proven [74] that the execution time of SAT attack grows exponentially if the ciphered circuit has a Tree-of-AND (TOA) structure between the primary inputs of the ciphered netlist and the key-bits. Accordingly, inserting a tiny design having some XOR/XNOR gates (between the key-bits (KA) and the primary inputs) and few AND gates as a tree can prevent SAT attack, with a small penalty, as shown in Figure 2.2 (a). TOA works as a one-function output, in which its output signal (S-O/P), denoted as a red line in Figure 2.2 (a), will be constant for all key combinations (2^{KA}-1), except one key. This technique was implemented with more theoretical analysis in [3], referred to as Anti-SAT, in [85], denoted as SAT-Attack Resistant Logic Locking (SARLock), and in [81].

In Anti-SAT technique, a light-weight circuit was incorporated. This circuit has two complementary blocks, G1 and G2, where each requires n key-gates. The two block outputs are fed to an AND gate, and the AND gate output is connected with one primary output to XOR gate. Another 2 n key-gates are inserted to hide Anti-SAT; one is employed to increase the interconnection between G1 and G2, and the second is utilized to increase the interconnection between the Anti-SAT block and the original circuit. In SARLock technique, a comparator with a mask is incorporated to corrupt the output of the circuit unless the correct key is provided. Also, a scrambling circuit is

added to scramble KE and KA and hence prevents an attacker from breaking SARLock by flipping its output and applying a wrong key equal to an input pattern.

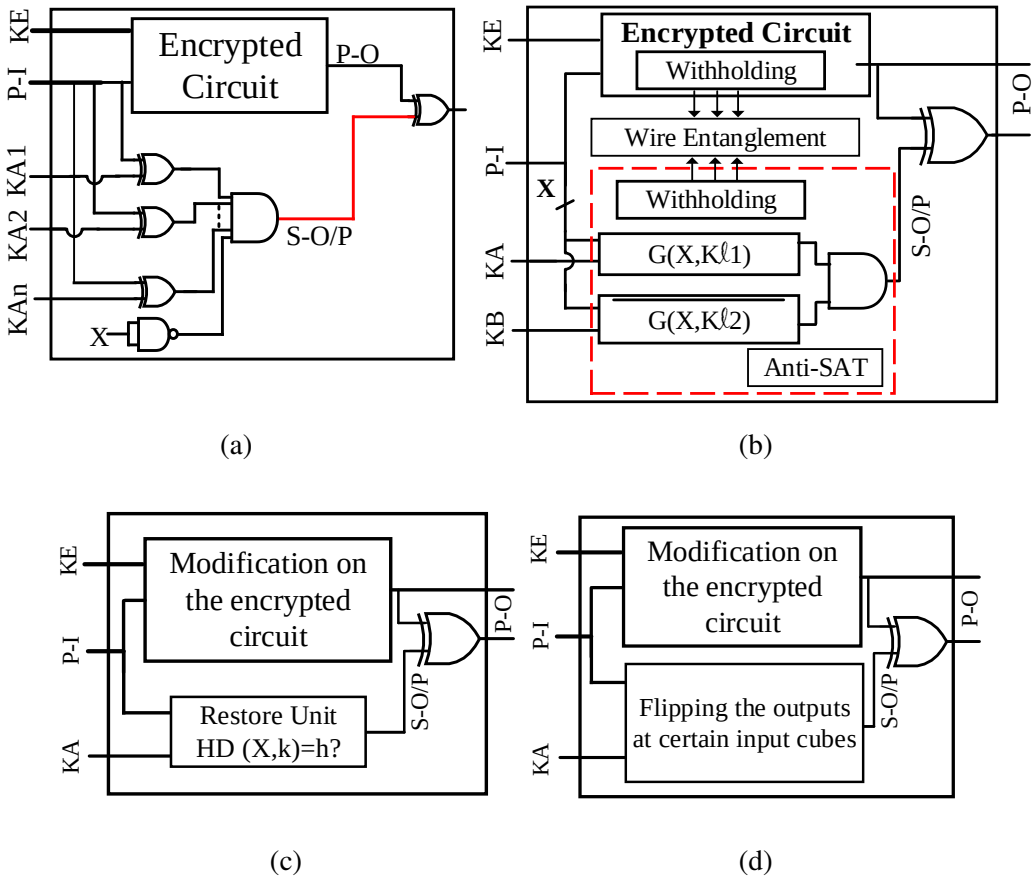


Figure 2.2: SAT prevention techniques: (a) traditional TOA: provides one-functional output, (b) Anti-SAT: produces a non-corruptibility for all key combinations, except one key [3], (c) SFLL-HD: gives correct output only when the keys are equal to the protected input cubes [4], and (d) SFLL-flex: flips the outputs for some set of input cubes.

Anti-SAT was shown to be vulnerable to AppSAT and Signal Probability Skew (SPS) [93] attacks. In AppSAT attack, random input patterns are applied to the encrypted circuit, and the corresponding output patterns are compared with the correct output patterns that are obtained from the activated circuit. Once the AppSAT attack finds a key that matches the correct outputs for that random

input patterns, the AppSAT attack will terminate SAT and this key will approximately satisfy all I/O pairs. SPS attack can identify and remove G1 and G2 blocks since they are inputs on a gate (the last AND gate in Anti-SAT technique) that has the maximum different signal probabilities. Similarly, the SPS attack can work as a tracking signal (TS) attack to detect the output signal of TOA block. Therefore, the Anti-SAT output signal can be detected and removed by TS attack since the value of the Anti-SAT output (S-O/P) is constant for all key combinations, except one. Although the 2 n key-gates that are used to obfuscate Anti-SAT can reduce the effectiveness of these attacks, the AppSAT guided removal attack (AGR) successfully broke the obfuscated Anti-SAT [93]. The main idea of this attack is to make AppSAT attack recover the correct key-bits of the traditional encrypted circuit as well as the key-bits that are used to obfuscate G1 and G2 blocks. Then, the removal attack (SPS) is applied to remove Anti-SAT since it will act as one-point function (TOA) after the AppSAT recovers a part of the total key-bits. The Anti-SAT structure [84] has been re-obscured using withholding and wire entanglement techniques to further mitigate AppSAT and SPS attacks, as shown in Figure 2.2 (b).

SARLock was shown to be vulnerable to double DIP (D-DIP) [94], AppSAT [95,96], Bypass [97], and TS attacks. D-DIP algorithm is used to corrupt the combined SARLock and rule out more and more incorrect keys at each iteration. Bypass attack selects two random keys for two encrypted circuits. Then, all DIPs that cause different outputs for the two selected keys are recorded. The correct corresponding outputs for those DIPs are recovered from the activated circuit. Afterwards, one of those selected keys is considered as the correct key, where an additional block is incorporated to recover the correct output at those DIPs. AppSAT and TS attacks can break SARLock block since it also works as TOA. TTLock [98] is an improvement on SARLock, where the original circuit is modified in a way to flip the output for a specific input pattern. Then, a block circuit is inserted, namely restore logic, to recover the correct output for that specific input pattern. An attacker cannot recover the original circuit encrypted using TTLock technique if he or she removes the S-O/P

signal. As a consequence, this technique is resilient to removal attack (SPS/TS attacks). Another improvement on TTLock, namely SFLL, [4] is achieved to further mitigate D-DIP, Bypass, and AppSAT attacks. The SFLL has two versions, SFLL based hamming distance (SFLL-HD) and flex (SFLL-flex). In general, each of these two versions utilizes many protected input patterns to increase the security level of the encrypted circuit; however, it requires large overhead. SFLL offers a trade-off between the security level of the design against the SAT attacks and the removal attack resilience. Figure 2.2 (c) and (d) show the SFLL-HD and SFLL-flex techniques, respectively. A similar proposal to SFLL has been introduced by Shamsi et al. but with lower performance overhead [99]. Unfortunately, SFLL was also shown to be vulnerable against a newly formulated attack, coined as Functional Analysis on Logic Locking (FALL) attack [100, 101]. An automated synthesis framework [102], called SFLL-fault, is proposed to further reduce the SFLL cost. However, such implementation has two limitations: (1) the locked output is obtained by modifying the original output at protected input cubes, where a synthesis tool is employed to mix the protected inputs into the flipped output. There is no theorem or formal proof that the synthesis tool can always accomplish this while preserve indistinguishability with the original circuit. Therefore, such implementation may not be secure since ATPG induced faults can likely be fingerprinted [103]. Mitigating such vulnerabilities has been addressed in [104], where a cryptography-based encryption technique has been presented. Unfortunately, incorporating cryptography incurs significant overhead. (2) the technique is not general as SFLL-HD and using SFLL-fault to protect a large number of cubic input patterns is challenge. Therefore, such technique may be good only for very large scale circuits, e.g., SoC [102]. Ideally, a scheme should be applicable to small circuits without loss of security because one conceivable way of using logic locking is to protect only a small security-critical part of the system. This can reduce the power, timing, and area overheads imposed by logic locking.

Note that the removal (SPS, TS, and AGR) attacks are strong due to the natural behaviour of TOA,

in which the internal signals (that are close to the last gate in the TOA) and the output signal of the TOA are barely changed when incorrect keys are provided. To be more specific, the input signals and the output signal at the last gate in TOA circuit are constant for all key combinations, except one. Such signals always have the lowest or highest signal skew values. Therefore, the removal attacks can easily detect and remove the TOA structure. Once the TOA circuit is removed, the encrypted circuit will again become vulnerable to SAT attack. These attacks can be prevented if the input signals and output signal (S-O/P) at the last gate in the inserted block, e.g. Anit-SAT or SARLock, are changeable and cannot be tracked or detected. In this paper, we will experimentally show how these signals in our new designs are changeable for most of the key combinations. Note that there is another type of removal attack, namely sensitization-guided SAT (SGS) attack. This attack is used to break TOA structure that is embedded inside the original circuit, in which the sensitization attack is used to get the DIPs. Then, SAT attack is used to break the encrypted circuit.

Using TOA structure is not the only way to prevent SAT attack. Shamsi et al. proposed a cyclic logic locking [105] to prevent SAT attack without the need to add a tree structure via creating a logic loop in the combinational circuit. The cyclic loop requires adding extra dummy gates and wires. The proposed technique is strong against traditional SAT-attacks; however, it has been broken by cycle SAT (CycSAT) algorithm based attack using different acyclic constraints [106]. Apart from using Boolean circuit behavior (e.g., TOA and cyclic logic locking schemes) to prevent SAT attack, parametric delay-based logic locking (DLL) is presented by Y. Xie et al [107]. DLL leverages tunable delay key-gates (TDKs) to obfuscate a design with two keys; one key is to encrypt the circuit functionality and the second is to control the time. Once incorrect keys are given, the encrypted circuit will produce wrong output and degrade the performance. Chakraborty et al. presented a new attack, coined as TimingSAT, to break the DLL technique [108]. TimingSAT attack uses pre-processor to analyze graph timing of DLL and then calls SAT solver to find the correct key. Recent work has also proposed a Satisfiability Modulo Theory (SMT) solver based attack that

enables the attacker to express constraints that cannot be easily expressed in propositional logic, such as graph representation, timing, and power [109]. Even though this attack successfully breaks some schemes of logic encryption techniques, e.g., DLL and cyclic logic locking, SMT attack fails to decrypt any scheme based on the TOA structure, e.g., Anti-SAT, SARLock, and SFLL techniques [110]. This is because SMT attack still uses SAT solver to find the secret key and therefore needs exponential time to break the TOA schemes.

2.3 Physical Attacks and Defenses

After the encrypted ICs get fabricated in an untrusted company, the secret key of the encrypted circuits should be stored in on-chip tamper-proof memory [72], which is not accessible to an attacker, by either a trusted facility [69] or a public key cryptography [24]. However, still the encrypted circuit may be subject to other advanced physical attacks. In general, there are three types of physical attacks: invasive (i.e., microprobing, physical reverse engineering tool, and the frontside/backside attacks), semi-invasive (i.e., fault-injection attacks), and non-invasive (i.e., power, timing, or electromagnetic analysis attacks) [111–113]. These attacks are powerful since they can extract sensitive information from secure ICs [114, 115]. Invasive attacks require expensive equipment and expert attacker to directly access specific internal parts in the chip (components or wires that carry sensitive information). Non-invasive attacks require cheap tools and do not physically damage the chip. Semi-invasive attack requirements are somewhere between non-invasive and invasive attacks. They do not require very expensive equipment and do not damage the chip, but they require expert attacker to depackaging the chip [116]. More information will be discussed regarding the potential physical attacks on our designs and other proposed techniques and the possible solutions.

2.4 Emerging Devices

Many researchers tend to utilize emerging devices, as an alternative to traditional CMOS methodology, in many applications in order to achieve lower power dissipation, smaller area and better performance. Several emerging devices have been experimentally demonstrated over the past few years, such as FinFETs [117], carbon nanotube [118], spin transfer torque [119], and ReRAM [120–122]. In this dissertation, we focus on ASLD and SiNW due to being the most promising devices for hardware security.

2.4.1 Spintronic Devices

While CMOS scaled-down reaches the end due to the physical limitations and the expensive cost, spintronic devices have been presented as an alternative to traditional CMOS technology. In the spintronic device, electron spin is utilized as a state variable for data storage and information processing. A magnetic field could be used to switch the magnetization direction of a nanomagnet at the cost of large energy and area consumption. Alternatively, switching a nanomagnet with current-induced spin-transfer torque (STT) is more power-efficient, and easier to control [123–126]. Hence, spintronic devices achieve lower power dissipation, smaller area, higher density, zero leakage current, non-volatile storing element, and faster switching. To further improve the performance of the spintronic devices and mitigate the consuming power when the device is off, CMOS-transistors are added to the nano-magnets instead of directly connecting the nano-magnets to the VDD supply [127]. This could be called Hybrid Spintronic-CMOS. Spintronic devices are suitable for the implementation of lightweight cryptography, image detection and processing, and machine learning [128–131]. However, differences between the read and write currents are generated, which increase the possibility of PA attacks [132].

The first proposal for spin-based logic in semiconductors was introduced by Dery et al. [133]. The proposed design contains non-local spin signals, which require a high amplification circuit to enlarge these signals and produce a sufficient current to switch the nanomagnets. An improvement on this work [134], namely All Spin Logic device, was achieved to switch a nanomagnet, which could be an input for the next stage, by employing the non-local spin signal without any extra hardware (amplification circuit). All Spin Logic Device (ASLD) is one of the best spintronic device candidates due to its unique properties: small area, no spin-charge signal conversion, zero leakage current, non-volatile memory, high density, low operating voltage, and its compatibility with conventional CMOS technology. Besides the ASL device, other emerging devices may also be leveraged to protect IC designs. For instance, the new generation of spintronic device, namely Domain Wall Motion (DWM). Even though this device can be used to implement different Boolean functions in-memory as polymorphic gates with ultra-low power and small area, it is not suitable for a large scale circuit since the layout of the design will be very complex, and the DWM reliability is a concern. The reason is that the DWM in not mature, and many researchers have worked on improving its reliability and scalability for a large circuit. It may really be a good candidate for future work.

2.4.2 Emerging Silicon NanoWire FETs

One of the issues we should discuss first is the phenomenon of ambipolarity, which is defined as the placement of both positive and negative charge carriers under bias constraints. It enables a designer to change the polarity of the device. A good example of leveraging ambipolarity is found in silicon nanowire [135], graphene [136], and carbon nanotubes [137], which have already been fabricated [138, 139]. Schottky barriers allow device functionality to be changed based on the external signal values. Among all of the above-mentioned devices, we concentrate on a vertically-stacked silicon nanowire FET, which includes two Gate-All-Around (GAA) electrodes [139].

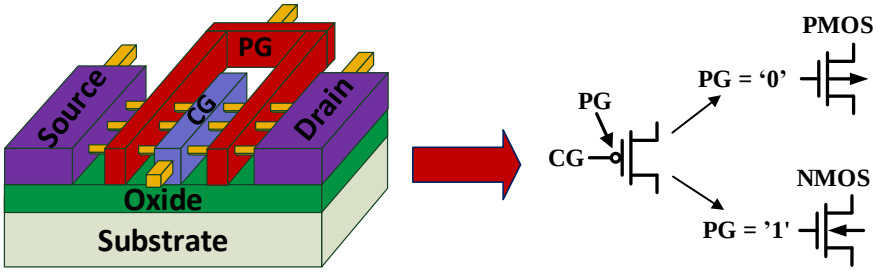


Figure 2.3: Three-dimensional scheme of the SiNW FETs with the characteristics of two separate gates, namely, the control gate (CG) and polarity gate (PG) to form either a p-channel metal-oxide-semiconductor (PMOS) or a n-channel metal-oxide-semiconductor (NMOS) field effect transistor.

The three-dimensional structure of Vertically-stacked GAA SiNWs is demonstrated in Figure 2.3. The benefit of using this structure is that it enhances electrostatic regulation. This device has two gates, namely control and polarity gates. In general, the transistor can be switched on and off based on the value of the supplied voltage at the control gate, while the polarity gate is used to swap the n and p channels, which is located between the Drain and Source junctions [139, 140]. There are several emerging devices that have polarity control features, such as carbon nanotube CNT FETs, SiNW FETs, nanoelectromechanical (NEM) relays, Graphene SymFET, and so on. This work is focused on SiNW FET because it is compatible with the modern CMOS. It should be noted that designing reconfigurable logic gates are not limited to emerging transistors only, e.g., SiNW FETs, Graphene transistors, or ASL. One can get similar characteristics using only CMOS transistors, but it requires a larger number of transistors as discussed in [141–143].

CHAPTER 3: STRONG LOGIC ENCRYPTION AGAINST NORMAL IC ATTACKS

3.1 Approach1: Traditional Encryption Using Multiplexer Insertion

Researchers have deduced that the logic encryption based on random XOR/XNOR insertion is vulnerable because it cannot achieve around 50% Hamming distance (HD) between the correct and corrupt outputs. Furthermore, the design functionality might be correct even when some of the wrong keys are entered in the ciphered design for some input patterns [69]. Fault impact analysis is an improvement on random insertion. Even though it can fulfill 50% HD for some benchmark circuits, it cannot grantee performing around 50% HD for any design, especially for those circuits with a large amount of gates and output bits. More specifically, in the fault impact analysis, the algorithm is going through each gate in the circuit to determine the location of the most impacted gate on the output and inserting the XOR/XNOR gate at each iteration. If the IC chip has a large number of gates, then the algorithm needs several months to achieve around 50% HD, which is practically impossible, and the performance overhead might be very large. Furthermore, in the fault impact analysis approach, both Rivest–Shamir–Adleman (RSA) cryptography and the physical unclonable function (PUF) have been utilized to boost the security. The hardware implementation of RSA is very expensive, which can override the area of VIRTEX5–XC5VLX50T device to implement such cryptography with only 32 bits of its key size [144]. Moreover, although PUF has low cost and provides unique keys for the chip, it is susceptible to environmental and operational condition variations, such as temperature, aging and humidity [145, 146], so that a noteworthy error correction process has to be employed [1].

[1] © 2017 Electronics. Part 1 of this chapter is reprinted, with permission, from [147]

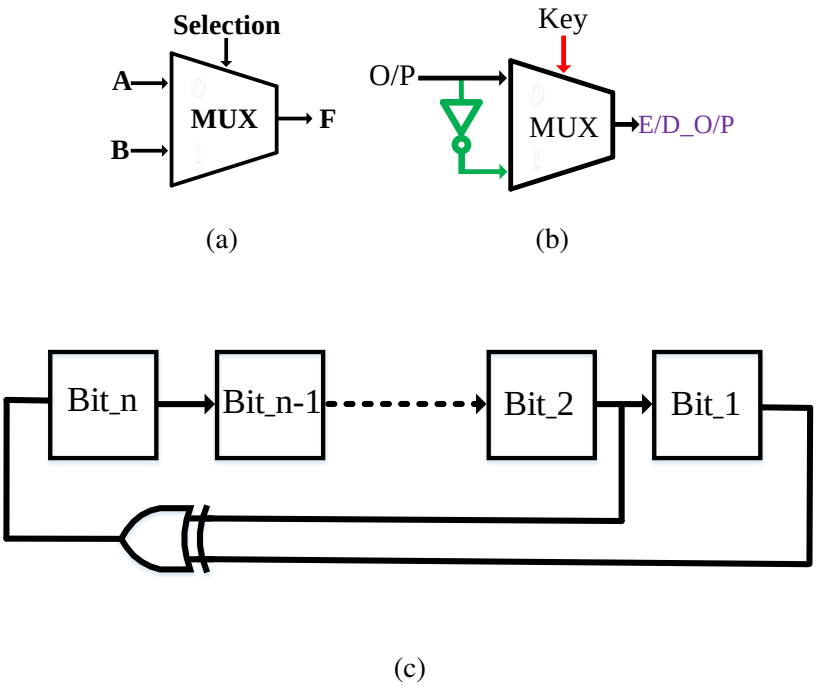


Figure 3.1: (**a**) Simple multiplexer. (**b**) Multiplexer to encrypt or decrypt the output bit. (**c**) Control the output value by the key bit.

3.1.1 Logic Encryption Using Multiplexers with LFSR

In this section, we demonstrate our methodology to secure a circuit design by leveraging multiplexers as key gates in a way that the output is corrupted by around 50% unless the correct key is supplied. The operation of the multiplexer is to propagate one of the input signals to the output based on the input selection. For instance, if the input selection is zero, then the output F is equal to input A, otherwise F is equal to input B.

Figure 3.1a demonstrates a two-to-one basic multiplexer diagram. Equation (3.1) explains the Boolean operation. A and B are two inputs, while selection is the chosen input and F the output.

Figure 3.1b illustrates an example of how the output can be determined by the key bit. If the key input is one, the Encryption/Decryption output (E/D_O/P) is equal to the complementary of the original output to encrypt the design; however, to decrypt the design, the E/D_O/P is equal to the original output (O/P) when the key input is zero. The functionality of the circuit can be changed based on the key value.

$$F = A \,.\, \overline{Selection} + B \,.\, Selection \tag{3.1}$$

Since the true random number generator (TRNG) is costly and not fast for several applications, such as creating keys and padding for encryption techniques, the linear feedback shift register (LFSR) is broadly employed to generate a pseudo random number generator (PRNG) instead, as in [148]. Generally, the LFSR is fast and inexpensive because it demands only a shift register operation and an XOR or an XNOR operation, as shown in Figure 3.1c. We employ the LFSR to generate pseudo random keys with half of the key bits ones and the other half zeros to increase the security level of the design. Basically, there are two types of LFSR: standard LFSR (also called internal feedback LFSR) and modular LFSR (referred to as external feedback LFSR) [149]. We choose the external feedback with 128 bits as a maximum length of the LFSR (number of flip-flops), and we consider the initial value of the LFSR as a constant value. The LFSR counts to $2^n - 1$ as a maximum number (periodicity) where n is the number of D Flip Flops (DFFs). An attacker needs more than five years to get all of the LFSR possible values once the length is larger than 63 bits [150]. We use Table 3 in [150] to get the polynomial equation with the maximum length LFSR counter, where the taps are an XNOR gate among bits 99, 101, 126 and 128.

The hardware Trojan mainly consists of two parts; trigger and payload. Trigger is the bare condition to activate the Trojan, while the payload is the act of the Trojan (the payload contains XOR/XNOR gates). We use this idea to conceal/expose the functionality of the design, and we

refer to it as the “hardware logic unit (HLU)”. The user key is considered as an input to a trigger; meanwhile, the payload is regarded as the activation of the design. Figure 3.2a illustrates the key idea of leveraging the HLU to protect the design, where K1, K2, ..., Kn are the user key bits, A is the target to give the initial values of each MUX and EN is to enable the LFSR generator. The functionality of the circuit will be correct if the output of the trigger is activated by supplying the valid user key. Meanwhile, the activation (A) will give the default of the MUX selections. Otherwise, the output will always be wrong. The fault analysis mode was used to affect as many outputs as possible, but it does not affect half of the output for every circuit. We insert the MUXs, at the output, in two ways based on the number of output bits in order to hit around 50% HD. The logic encryption based on MUX insertion is classified as follows:

3.1.1.1 MUX Insertion Based on Half Output Bits

To insert the multiplexers, we need to count the number of output bits in the design. Then, we randomly select and complement half of the output bits, as well as inserting an MUX with these two (output bit and its complementary) as inputs and a key bit as a selection. Due to the insertion, the Hamming distance between the correct and corrupt outputs is always 50% when the number of output bits is even or close to 50% when the number is odd. Even though the HD is around 50%, the design is not secure enough since the input key selections for all inserted MUXs are weak (all of the input key values have to be ones), which means that the key is very easily exposed to the attackers. In order to make the key more secure and increase the ambiguity of the attackers, we randomly create half of the input key bits as zeros and the other half as ones, then flip the value of the key bit once it is zero to make sure that all of the outputs of MUXs are equal to the complementary of the original output bits. Instead of changing the key bit value, one can invert the output of the MUX (but it also needs one more inverter) or exchange the inputs of the MUX when the key bit value is zero.

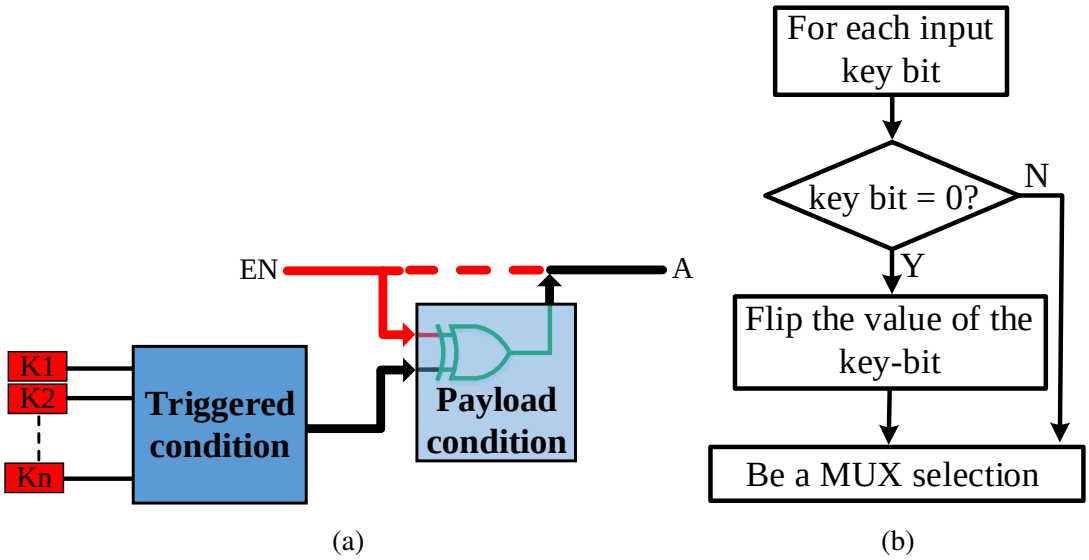


Figure 3.2: (**a**) Hardware logic unit (HLU) idea to protect the design. (**b**) Flip the key bit value.

To make the design much more secure, RSA cryptography and PUF were proposed in [92] to provide a fully-encrypted design. This was done without actual evaluation for the power, area and delay overheads, where the overhead of RSA cryptography is too large and the reliability issue of PUF is a concern. Instead of using RSA and PUF, we implement the HLU idea and the LFSR pseudo-random generation with some constraints to ensure that each generated random key consists of half zeros and half ones. The detailed constraints for the LFSR are: (1) making less than half of the initial values of the LFSR as ones to accelerate the generation of half zeros and half ones of the key value; (2) checking whether the new pattern has half zeros and half ones; if not, it neglects the pattern and picks up the next pattern, and so on, until reaching the correct pattern; (3) providing the MUX selection with the correct pattern. If the value of the key bit is zero, it must be inverted before providing it to the MUX selector, which is an additional simple condition after the key generation. Figure 3.2b displays the completed flipping key bit technique.

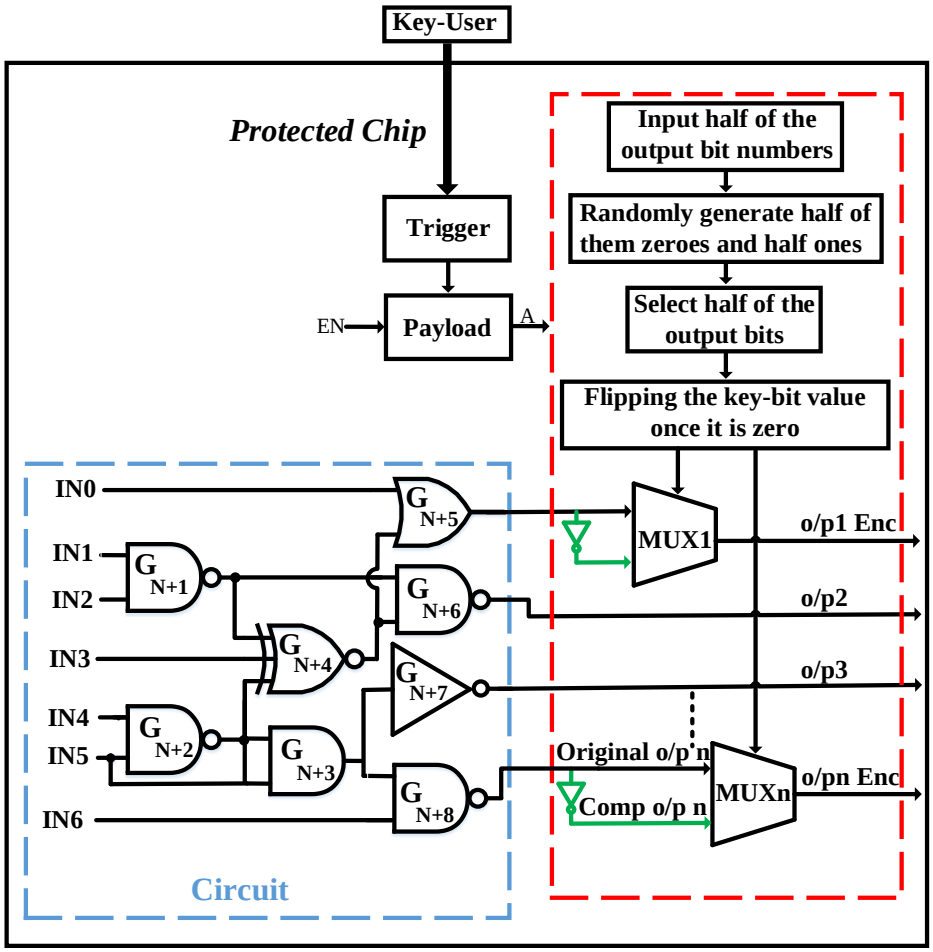


Figure 3.3: Logic encryption based on half multiplexer (MUX) insertions. o/p, Comp, n and Enc are the output, the complementary, the number of the output bits and encryption, respectively.

The maximum protection level can be achieved by combining the inserted MUXs, the HLU idea, flipping the key bits and distributing its values. The output will keep malfunctioning with random keys unless the valid user key is provided. With the right key, the output of the trigger will be activated, and the payload will set the activation (A). Once A is set, the value of the secret key will be provided to the MUX selections, and the functionality of the circuit will be correct. Otherwise, the LFSR enable (EN) will be activated to generate random key patterns with half zeros and half ones. Figure 3.3 shows the whole protected design based on half MUX insertions, where o/p,

Comp, n and Enc are the output, the complementary, the number of the output bits and encryption, respectively. Assuming that n is an odd number, half of the output number is considered as $(n-1)/2$. Although this technique is efficient against various threats, such as IP piracy and counterfeiting, it might be vulnerable to an attacker who may figure out the functionality of an IC based on some exposed output bits. To prevent experienced attackers, we then propose the full MUX insertion technique.

3.1.1.2 MUX Insertion Based on All Output Bits

To maximize the protection of an IC from various attackers, we propose to insert an MUX at each output bit, as shown in Figure 3.4. The inputs of the MUX will be the original output bit and its complementary, along with a key bit for the selection of each MUX. The values of the key bit selection must be random with half zeros and half ones to produce 50% HD. Since each output bit and its complementary are connected to a MUX with a random key bit selection, each output bit of the IC is changeable once the key is changed. In this case, not only the HD between the corrected and corrupted outputs is around 50%, but also the value of each output bit is variable.

An assailant cannot figure out the functionality of the design because each output bit will be varied by changing the supplied key via the LFSR generator, which is used to generate random keys (each key is generated to have randomly half zeros and half ones, as mentioned). Since the key value is unpredictable due to the random generation, each output bit will be consequently arbitrary. Once the correct user key is inserted, the output of the payload will be set, and then, the enable (EN) of the LFSR generator will be disabled, while the activation signal (A) will be activated to initialize the values of the MUX's selections. Then, the functionality of the circuit will be correct. If the value of one bit in the user key is incorrect, the corrupted output ratio will still be around 50%.

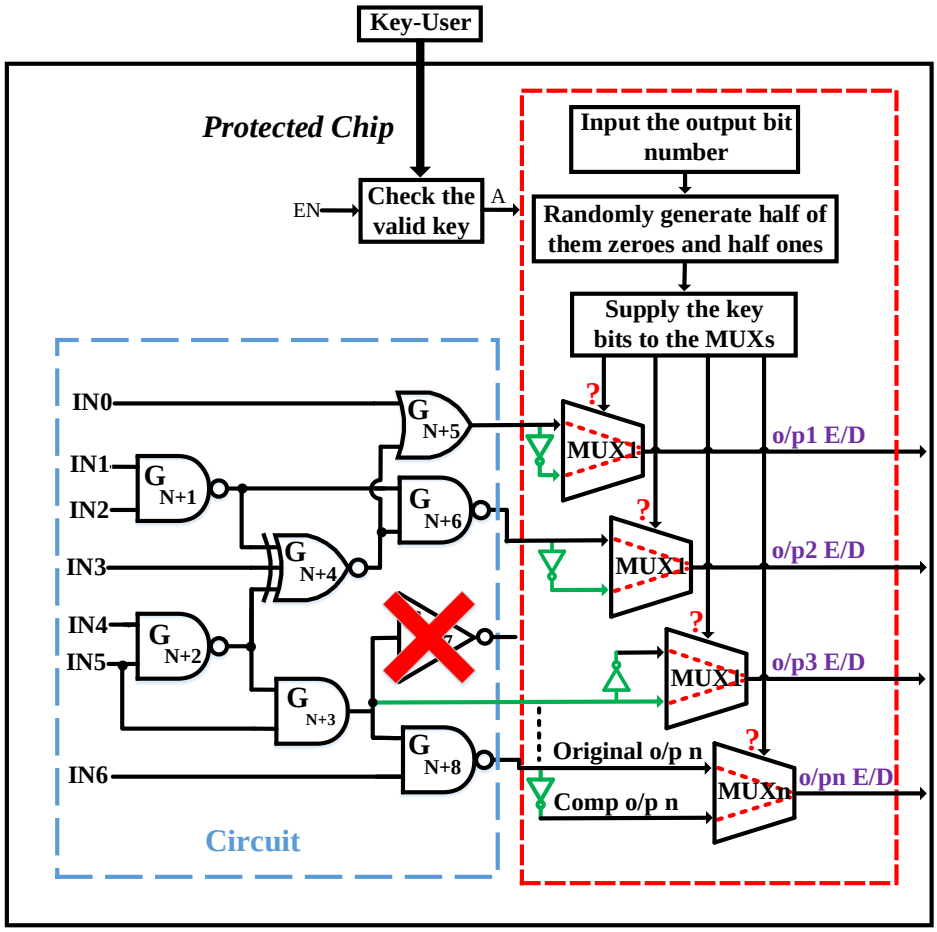


Figure 3.4: Logic encryption based on full MUX insertions.

Although inserting MUX at each output bit will obviously maximize the protection of the design, as well as the ambiguity of an attacker, the power and area overheads will largely increase. Therefore, this technique is more suitable either for large circuits that include a large amount of output bits or for an expensive IC chip. In both half and full MUX insertions, if there is an inverter at an output, we replace it by an MUX with switching its inputs. Furthermore, all components of the encrypted circuit (in half and full MUX insertion techniques) are made at a pre-layout stage.

3.1.2 Results

3.1.2.1 Experimental Results

Both the IEEE International Symposium on Circuits and Systems (ISCAS)-85 (combinational) and ISCAS-89 (sequential) benchmarks are employed to analyze our methodology. The C language is used to randomly select and add key gates (or replace an inverter with a key gate if the gate at the output is an inverter) for both half and full MUX insertions. The performance is evaluated by Synopsys Design Compiler.

The Verilog language is also leveraged to design two LFSR random generations; one of them is used to generate 1000 random patterns for the inputs of each netlist, and the second is utilized to generate random keys with equal probability, so the value of the output bits will be controlled by these key bits (each selected output bit is inverted when each key bit is set). We adopted the 1000 random input patterns that the first LFSR generated for the original netlist, and the output results were saved in an array. Afterwards, we supplied the same 1000 random input patterns to the same netlist with a random key that the second LFSR generated, and the results were also saved in an array. We make the initial value of the second LFSR as a constant value. In order to evaluate the Hamming distance between the corrected output when the key selection is at the default value (the valid key is given) and the corrupted output when the second LFSR key generator provides a random wrong key with equal probability to the MUX selections, we designed a shift register to provide each random key bit for each of the 1000 random input patterns starting from the first key bit till the last one in the register.

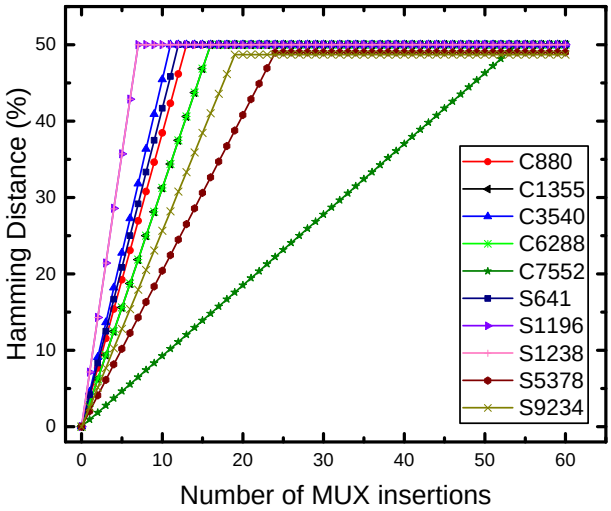
Table 3.1: number of MUX insertions based on half and full output numbers with the Hamming Distance evaluating and comparing with random and Fault Impact Analysis approaches

Benchmark	**number of XOR/ XNOR insertions**	**number of Half / Full MUX insertions**	**Hamming Distance**		
			Random	**Fault Impact**	**Full / Half MUX**
C880	28	13/26	19	50	50
C1355	42	16/32	26	50	50
C3540	22	11/22	23	50	50
C6288	27	16/32	32	50	50
C7552	89	54/108	13	46	50
S641	29	12/24	37	50	50
S1196	-	7/14	-	-	50
S1238	-	7/14	-	-	50
S5378	106	24/49	29	50	49
S9234	39	19/39	14	50	48.72

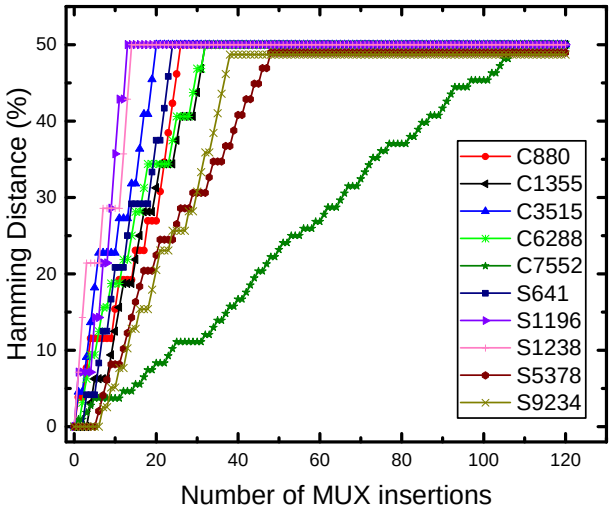
3.1.2.2 Hamming Distance Evaluation

The inserted MUX method based on half and full output bit numbers is compared with both the random and fault impact analysis-based on XOR/XNOR insertion approaches, and the illustrated outcomes for the minimum required number of inserted MUX key gates to achieve the hamming distance (for each benchmark circuit) are depicted in Table 3.1, where sequential circuits S1196 and S1238 were not tested by the random and fault impact methods.

The use of the random insertion method did not achieve 50% HD, while the fault impact analysis achieved 50% HD, but not for any circuit and not for any input patterns, especially when the number of output bits is very high (more than 100 output bits), such as the combinational benchmark circuit C7552 , besides the complexity of the fault impact algorithm to find the highest impact gates on the output, which will take a very long time to test the whole design for a large circuit, such as C7552. In contrast, our methodology ensures that the HD will be 50% or close to it (such as S9234) once the design has an even or odd output bit number, relatively.



(a)



(b)

Figure 3.5: (**a**) MUX insertions based on the half output number for different ISCAS-85 and ISCAS-89 benchmark circuits. (**b**) MUX insertions based on the full output number for different ISCAS-85 and ISCAS-89 benchmark circuits.

In addition, we evaluated the HD for all of the benchmarks that are mentioned in Table 3.1. For each benchmark, we only picked up one random key that was generated by the LFSR random key generation and fed it to each netlist to evaluate the HD between the correct and corrupt outputs based on half and full MUX insertions. Figure 3.5a, b demonstrates the analyzed HD for

the ISCAS-85 and ISCAS-89 benchmarks based on the full and half MUX insertions for logic encryption, where the minimum required length of LFSR to achieve the HD in the figure is equal to the number of output bits. The HD for these benchmarks is 50%, except for S9234, which is 48.72% due to having an odd output number.

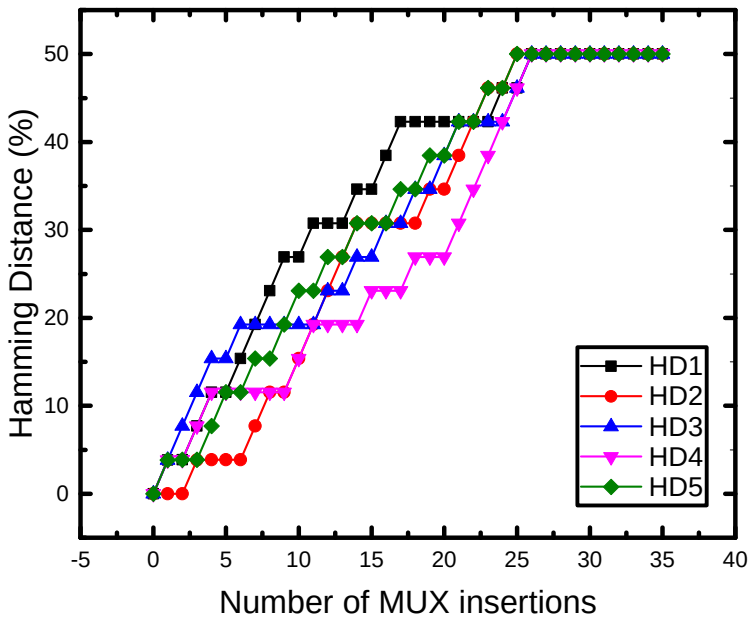


Figure 3.6: The Hamming distance when five different random keys were supplied for C880.

In our work, not only the functionality of the circuit is incorrect when the wrong user key is entered, but also the Hamming distance is always 50% or close to it, for whatever the input test patterns, the wrong input key and the size of the circuit are, unless the correct user key is inserted. To analyze the effectiveness of the random keys that the LFSR generated, we tested the C880 benchmark with five random keys using the full MUX insertion technique. First, we supplied 1000 random input patterns to the original C880 and monitored the correct output, then we supplied each key bit in each of these random keys with the same 1000 random inputs starting from the first bit until the last one by leveraging a shift register, to evaluate the HD for these five random keys. The HD was always 50% with different output patterns, as shown in Figure 3.6. This means that all of the

output bits are always changeable and cannot be predicted, even if there is only one bit in the user key that is incorrect.

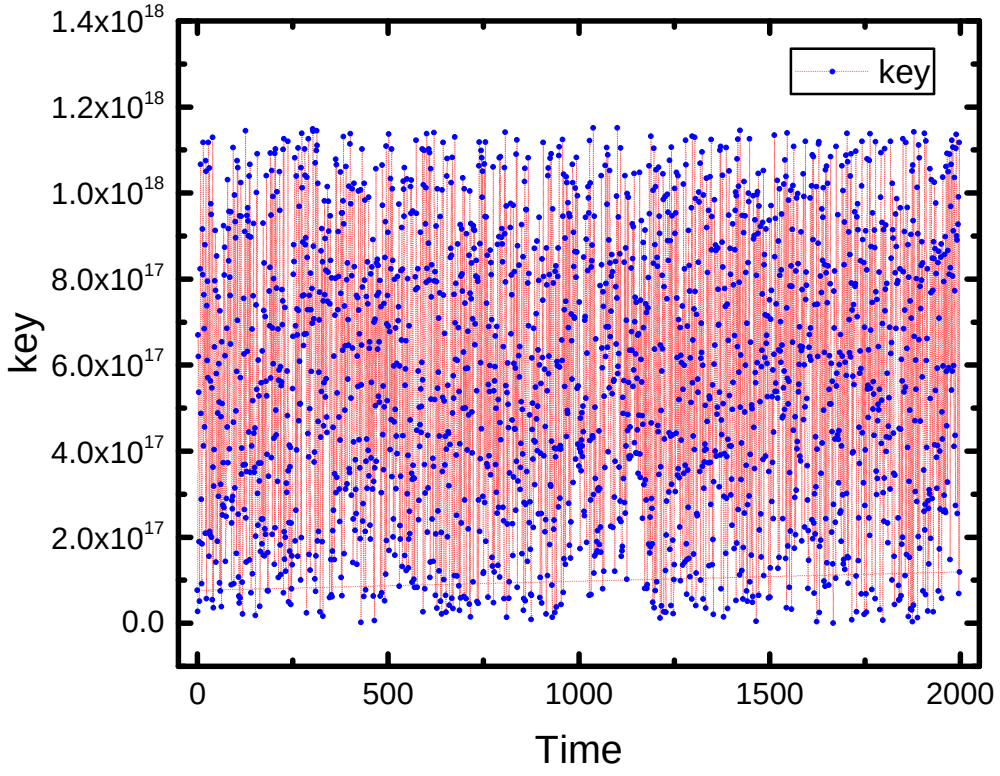


Figure 3.7: Random key generation, each having half zeros and ones.

3.1.2.3 Random Key Generation

The selections of all MUXs must be provided with the right key bit values at any time for whatever the input patterns are to maximize the protection of the circuit. LFSR always provides the MUX selections with these values to corrupt half of the output bits when the enable signal (EN) is set. Once the valid key (user key) is inserted, the payload will provide the selections of the MUXs with their default values, and the functionality of the circuit will be correct. To make sure that the generated keys by the LFSR are random with equal probability, we observed and saved the outputs

of the LFSR in a file only for the first 2000 numbers by making its initial length 60 bits (each output key has 30 zeros and 30 ones), and the outputs are as shown in Figure 3.7.

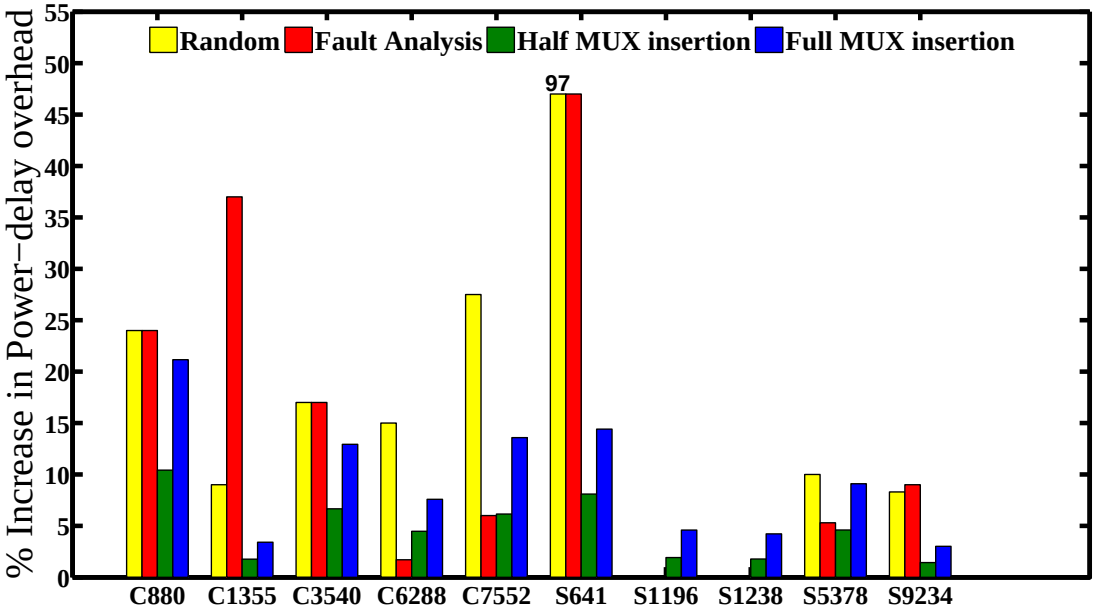


Figure 3.8: Comparing the power-delay overhead of random, fault analysis and full/half MUX insertions for logic encryption.

3.1.2.4 Delay, Power and Area Performances

We measured the delay, power and area overheads for each benchmark circuit using the Synopsys design compiler as a tool with the 45-nm CMOS library. Since the MUXs were inserted only at the output of the netlist, the delay overhead (timing path) is almost zero for all of the benchmark circuits. Meanwhile, the power and area overheads for each benchmark depend on the number of output bits. Increasing the number of output bits will significantly increase the overheads for the area and power. Figures 3.8 and 3.9 show the power-delay product and area overheads for all of the benchmark circuits that are listed in Table 3.1 with the corresponding number of MUX insertions that are mentioned in Column 3.

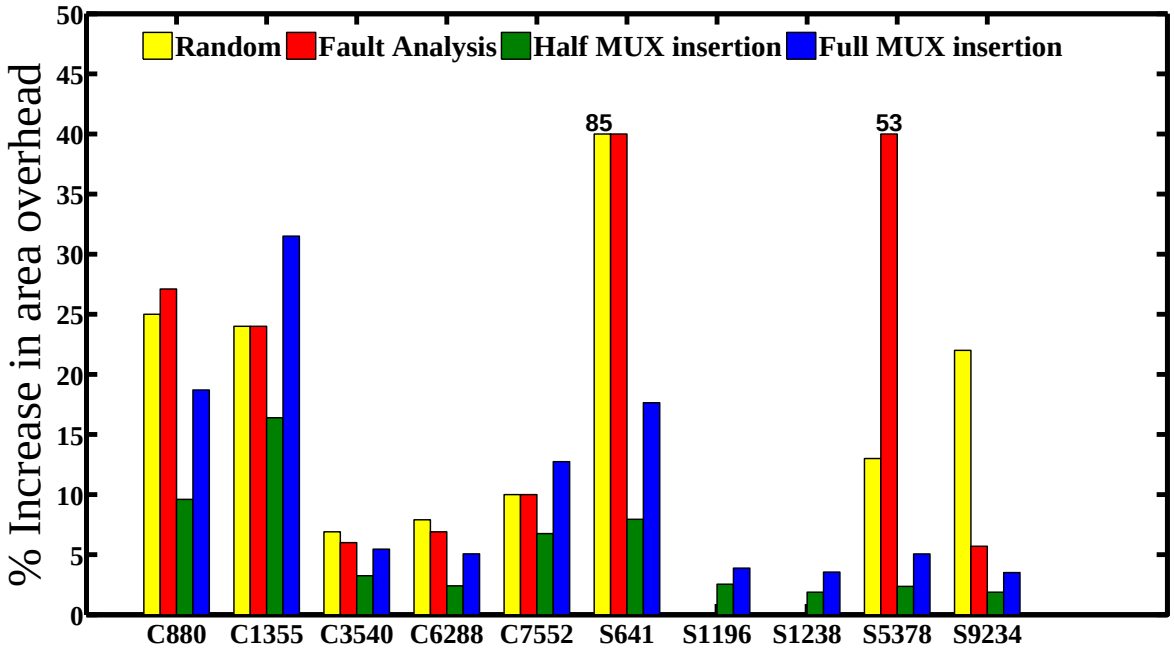


Figure 3.9: Comparing the area overhead of random, fault analysis and full/half MUX insertions for logic encryption.

From the figures, we can see the full MUX insertions consuming power and area approximately twice a half of the MUX insertions. The reason is obviously because the number of MUX insertions in full insertion is twice a half MUX insertions. It is worth noting that the half and full MUX insertions save area more than 2× and 3.6× and the power-delay product with more than 2× and 3.4× on average, respectively, compared to fault impact analysis.

Furthermore, in order to enhance the security level for the design by generating a unique key for each IC, Rajendran et al. in [92] employed the PUF circuit and RSA encryption asymmetric cryptography. The RSA overhead will be very high (it overrides the area of VIRTEX5, as mentioned in Section 3.1), where the authors only proposed using PUF with RSA cryptography without a true evaluation for the delay, power and area overheads.

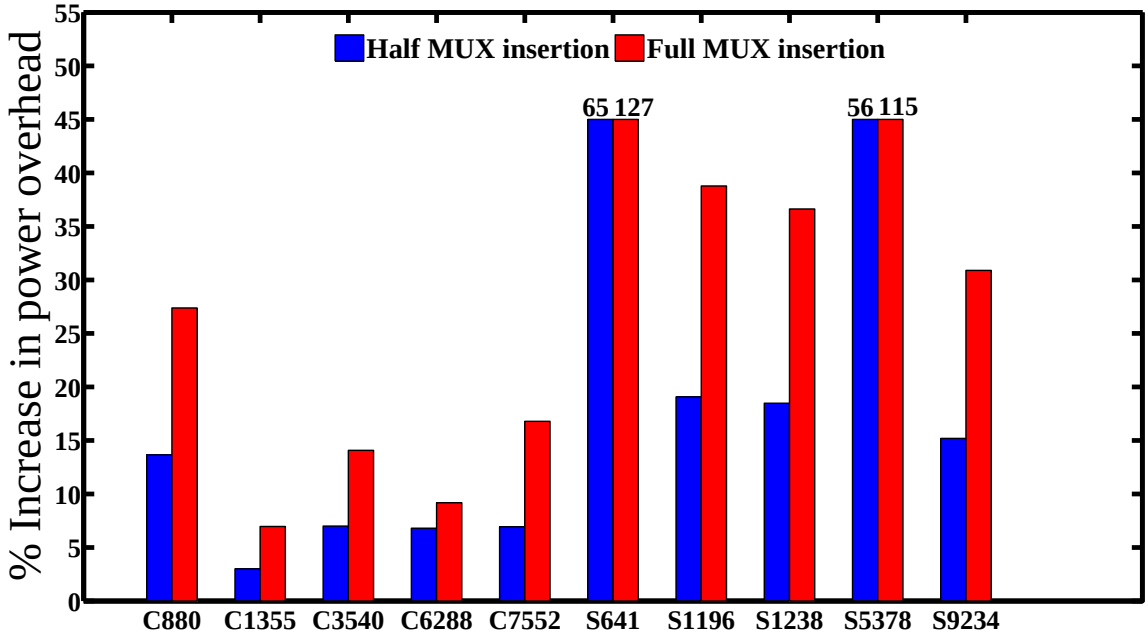


Figure 3.10: Comparing the power overhead of full and half MUX insertions for logic encryption.

Instead of using RSA and PUF, we adopt both HLU and LFSR as a different technique to protect the secret key and generate random keys with 0.5 probability for each one, respectively, as mentioned in Section 3.1.1.1. Both the HLU and the LFSR random key generation are designed and synthesized with the full and half MUX insertions using Xilinx ISE. We also evaluate the delay, power and area overheads for the whole design, including all components. Figures 3.10 and 3.11 demonstrate the total power and area overheads for the whole design, which are appropriate for several benchmark circuits, but are not suitable for other benchmarks, such as C880 and S641, due to the small area compared with S9234 and C6288. Moreover, the delay (timing path) overhead is very small for most of the benchmarks, except for C6288, which was 8.49%.

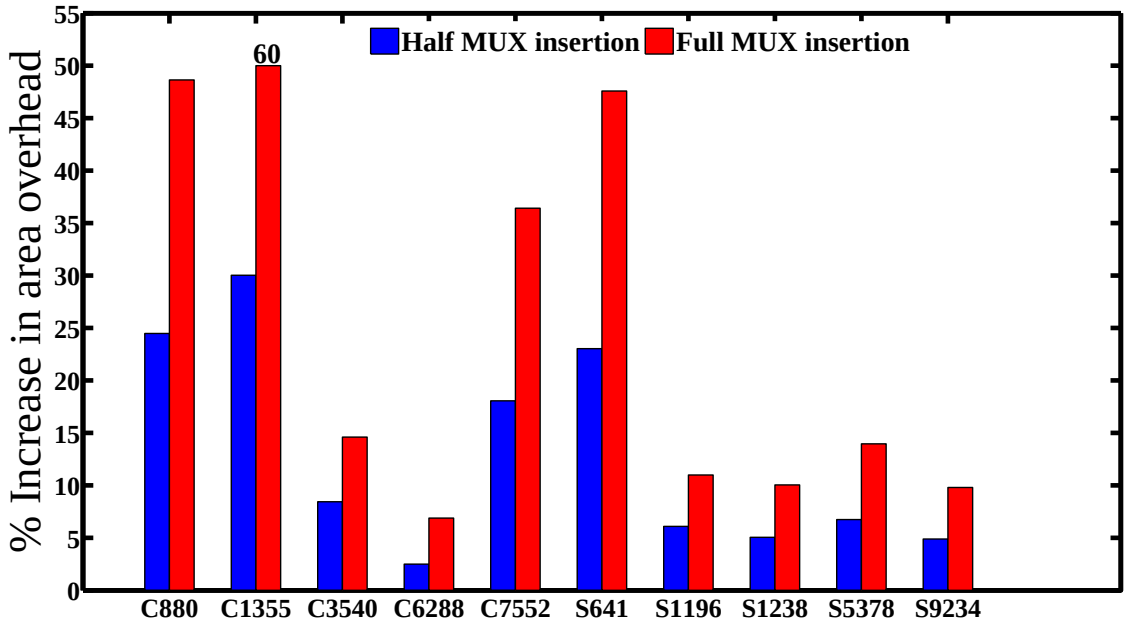


Figure 3.11: Comparing the area overhead of full and half MUX insertions for logic encryption.

3.1.3 Discussion

3.1.3.1 Durability of the Logic Encryption

The most substantial part in the encrypted circuit is the valid key. Once it is known by an attacker, the functionality will be revealed despite how strong the encryption technique is. The attackers can utilize many different ways to expose the functionality of the circuit, but the most dangerous attacks are in the following:

3.1.3.1.1 Using the Brute Force Algorithm

An attacker can use the traditional brute force algorithm to get the right key and expose the functionality of the circuit. The number of different combinations that an attacker needs to decipher the

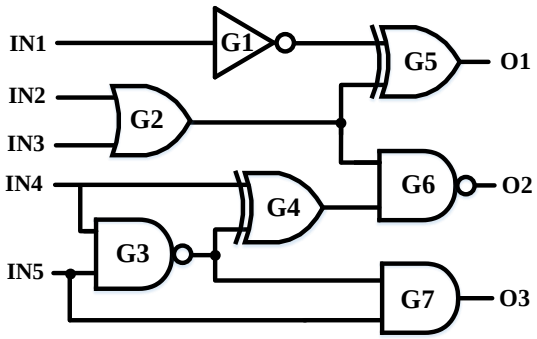
design can be computed by Equation (3.2), where K is the key size, and the input test patterns are different from one circuit to another. If the length of the user key is large enough, such an attack becomes infeasible. However, increasing the key size means inserting more gates (each new key needs an inserted XOR/XNOR gate), which leads to a great increase of the power, area and delay overheads, where the designer has the limitation of increasing the key size. In our work, increasing the user key size will not increase the performance overhead that much, since the user key size will only increase in the trigger part (see Figure 3.2a). Adding a new key bit to the user key requires only to add either an inverter or a wire if the designer needs the value of the new key bit to be zero or one, correspondingly.

$$Number\ of\ different\ combinations = 2^K * input\ test\ patterns \tag{3.2}$$

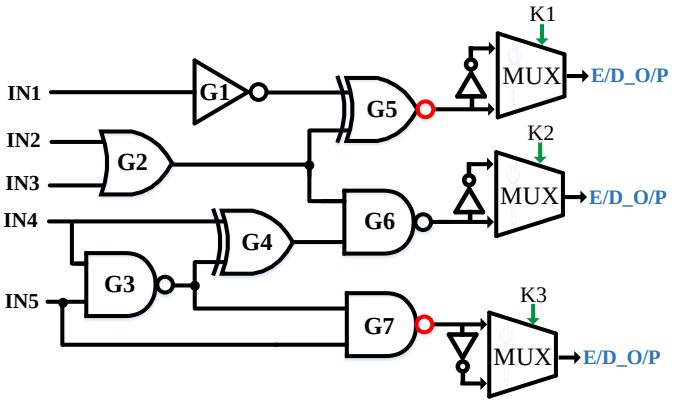
3.1.3.1.2 Removing Both LFSR and MUX Key Gate Insertions

An attacker can remove the LFSR random generator and the inserted MUX key gates to get the original design. Then, he or she can copy the IC illegally. However, the attacker can not get the correct functionality by removing the LFSR and the MUX key gate insertions for two reasons: (1) The MUXs are not only inserted at the outputs, but also some of them are exchanged with inverters in the original design. In this case, the attacker will not know whether the inserted MUX at each output is added or replaced by an inverter. If the original circuit has no inverters at all of the outputs or each primary-output has an inverter, then the attacker can easily get the original IC. In this case, a designer can use another way to prevent such an attack by adding inverters before some of the inserted MUXs and combining them with their previous gates. For example, Figure 3.12a shows an original circuit, and Figure 3.12b shows the encrypted circuit with three different key values as “K1K2K3=011”. An attacker cannot know whether an inverter is added and combined with the last gate before the inserted key gate or not. Furthermore, we consider adding or replacing

the key gates at the outputs as a minimum requirement to achieve 50% HD. One can increase the ambiguity of the attacker by randomly adding more key gates and replacing others with inverters through the circuit with keeping the correct value of the key bit (without changing the correct value of the internal net signal after the insertion). (2) The synthesis tools can also help prevent an attacker from realizing whether there is an inverter added since the synthesis tools use inverters through the regular synthesis of a design (not for logic locking purposes).



(a)



(b)

Figure 3.12: (a) Example for an original circuit; (b) the encrypted circuit using the MUX key-gate insertion technique.

3.1.3.2 Limitations and Future Works

Although using the LFSR random generation produces random keys with 0.5 probability, it will not guarantee generating each key during each clock cycle, especially when the number of output bits in the design is too large (larger than 100). In this case, the old generated key will still be used by the selections of the MUXs until the new generated key by the LFSR is created, even when a new input pattern is supplied. Besides, the power, area and delay overheads increase for a circuit having a large number of output bits. In [151], Dubeuf et al. utilized dynamic scrambling to change the order of instructions before storing them in the main memory and after reading from it to protect the system from several types of Trojans. A designer can use one register with 0.5 probability of a random key as an initial value, where the length of the register is the same as the number of output bits, then scrambles the key value before providing it to the MUX selections and makes a rotate right or left operation after each clock cycle to change the initial value in the register and to ensure generating a new random key for each input pattern.

Even though the half and full MUX insertion techniques achieve 50% HD for most of the benchmark circuits with a reasonable cost, both might not be strong enough against an experienced assailant. More specifically, the MUX insertion method based on the half output number gives lower power, area and delay overheads, but it might be vulnerable to the uniformity distribution attack. A proficient attacker could get the correct functionality of the design based on some true output bits, since half of the output bit number is always exposed. For instance, an attacker can input a constant vector with the wrong user key and makes the clock run for a long number of cycles, where the LFSR will switch the outputs. Afterwards, he or she will realize that some of the outputs are constant and tries to infer the internal functionality. Furthermore, the MUX insertion at each output bit is very strong against many types of attacks, including this one, because each output bit in the design will become changeable once a random key is supplied to the selections of

the MUXs. Nevertheless, both half and full MUX insertion approaches are vulnerable against the LFSR tracking attacks.

Since the initial value of the LFSR is constant and the LFSR sequence can be tracked, an attacker can use the temporal repeatable response to figure out the functionality of the design. For instance, an attacker can first supply a constant input vector (e.g., Input vector A (I_A)) with the wrong user key and then record the secured output in an array Secured Output A (SOA(t)). After that, he or she can reset the circuit, use another input vector (e.g., Input vector B (I_B)) and record the output in another array Secured Output B (SOB(t)). Next, the attacker compares the two secured output vectors and produces a post-process data as follows: SOA(t) XOR SOB(t) = OA(t) XOR OB(t), where OA(t) and BO(t) are the correct outputs for the input vectors IA and IB, respectively because the key is the same in both cases (due to the temporal repeatable response of the LFSR). Finally, he or she can take SOA(t) as a reset response and generate new outputs for other inputs (Input vector C (I_C), Input vector D (I_D), Input vector E (I_E), etc.), and from the post-processing results, he or she might infer the correct functionality of the circuit. Preventing an attacker from tracking the LFSR may be achieved by using a dynamic scrambling to scramble the seed value of the LFSR, where its configuration bits should be coming from a true random number generation. In this way, an attacker cannot use the temporal repeatable response to track the LFSR because its initial value will be changed by the dynamic scrambling.

In addition, the designer can leverage our dynamic scrambling proposal instead of employing LFSR random generation for two purposes: (1) reduce the performance overhead; (2) expedite creating a new random key with 0.5 probability and then providing the selections of the MUXs with the new one. The direction of the rotating operation and the configuration of the dynamic scrambling should be changed for each IC to evade the rogue duplicating by the untrusted companies and/or adversaries. Figure 3.13 manifests the way of using the rotating operation with the dynamic scrambling.

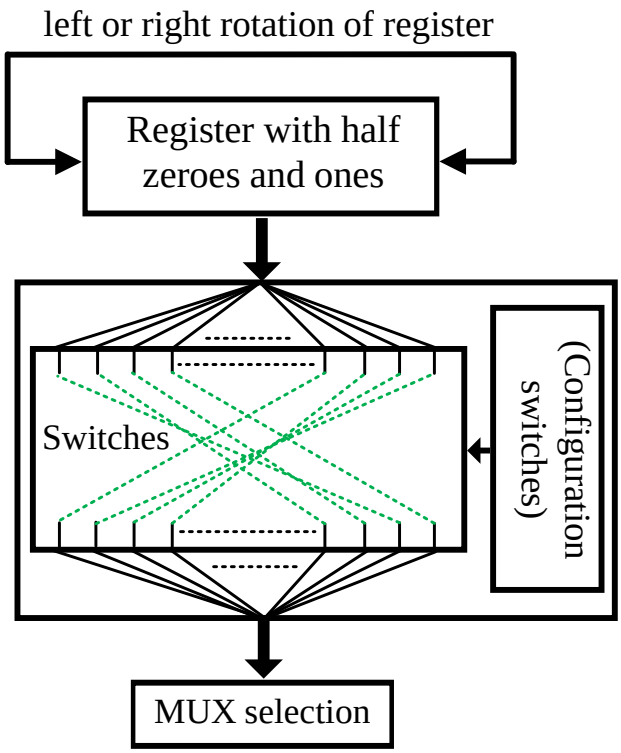


Figure 3.13: Generating a random key using the rotate right or left operation with dynamic scrambling.

3.2 Approach 2: Conventional Encryption Using Hybrid CMOS and Emerging SiNW FETs

3.2.1 Designing Polymorphic Gates Using SiNW FETs

Polymorphic electronics, which were first introduced in [141], are based on the idea of having multiple functionalities built in the same cell and deciding the input–output relation by means of a controllable factor in the circuit. For instance, a polymorphic gate presented in [141] would be an AND gate when the supply voltage (VDD) is 3.3 V and it functions as an OR gate when VDD is lowered to 1.5 V. Such multi-functional gates would prove useful in a number of applications [152, 153]. Circuits that change functionality with temperature variation can find use in aerospace

applications, or those that respond to VDD variation could be used to change functionality when the battery is low. In addition, polymorphic electronics could be beneficial in evolvable, intelligent or self-checking hardware [143]. For security objectives, adding polymorphic gates to a digital circuit can hide the real functionality of the circuit. Since the circuit functions correctly only in a certain configuration of the control signals known to the designer, even if the adversary knows the whole netlist (including the dummy and true contacts), he or she will not be able to utilize the circuit in his or her own design [92]. Carefully encrypting logic in this way can ensure that it will take too long for the adversary to find the key (a vector constructed from all the morphing signals of the polymorphic gates). Therefore, the polymorphic gate becomes a good candidate for integrated circuits protection against IP piracy [2].

Table 3.2: A summary of developed polymorphic gates.

Polymorphic Gates	**Techniques**	**# of Transistors**	**Publications**
XOR/NAND	3.3/0V External signal	9	[143]
NOR/NAND	1.8/3.3V VDD	6	[141]
OR/AND	3.3/1.2V VDD	8	[142]
XOR/AND/OR	1.5/3.3/0.0V External signals	10	[142]
OR/AND	0.0/3.3V External signals	6	[142]
AND/NAND/XOR/NOR	1.8/0.0/1.1/0.9V External signals	11	[142]
OR/AND	125/27 C Temperatures	6	[142]
NOR/NAND	exchanging *key* and $\overline{key}$	4	Our Work

Various polymorphic logic gates using CMOS technology are implemented via leveraging several techniques, such as external signals, different temperatures, and multiple VDD values. Table 3.2 shows a brief recapitulation of implementing different polymorphic logic gates. In [141], polymorphic logic gates were achieved using a smart algorithm. However, on applying an external signal, the designs encounter a problem through the simulation test, which is producing constant current at the output signal of the polymorphic gates, e.g., NOR/NAND. Moreover, connecting

[2] © 2017 Electronics. Part 2 of this chapter is reprinted, with permission, from [88]

many stages of polymorphic gates in series causes another problem because, in some cases, their inputs might be connected to VDD or ground (GND). A more empirical technique is to use different VDD values, which has been already done [141]. However, employing many VDD values is not a feasible solution, especially with the new scaling technology, where the ranges of VDD are restricted. Designing XOR/NAND polymorphic gate with nine transistors [143] is considered as a good technique for emerging devices.

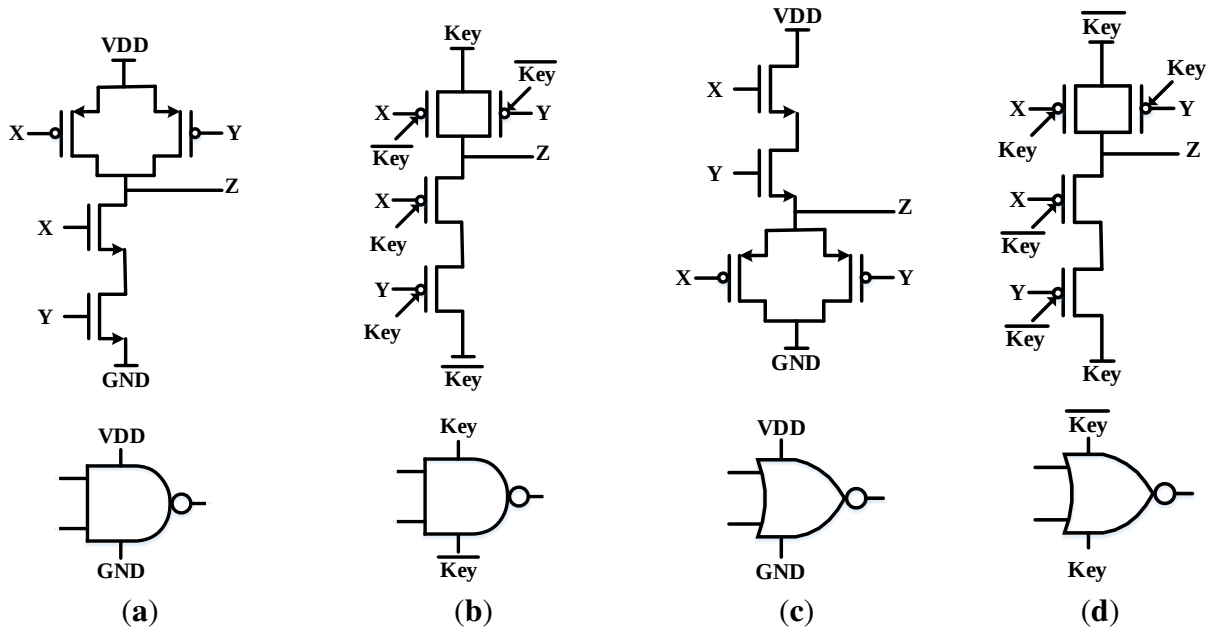


Figure 3.14: Different logic gates using complementary metal oxide semiconductor (CMOS) and silicon nanowire (SiNW) devices: (**a**) traditional CMOS NAND gate (**b**) silicon nanowire field effect transistors (SiNW FETs) NAND gate (**c**) conventional CMOS NOR gate (**d**) SiNW FETs NOR gate.

Now, we present our technique to implement different polymorphic gates for IP protection features employing the polarity control signal of the SiNW FET device. SiNW FET is very similar to CMOS except for the addition of the polarity gate between the drain and source junctions. As demonstrated in Figure 3.14, the structure of both a NAND and a NOR gate is not different in CMOS and SiNW devices. By only swapping the value of the control signal, denoted as $key/\overline{key}$

in Figure 3.14b,d, a designer can easily exchange the functionality of a gate with the same structure without any other extra resources. More precisely, in Figure 3.14b, if the $\overline{key}$ value is zero and the key value is one, the logic gate works as a NAND gate, while it works as a NOR gate when the values of $\overline{key}$/key are interchangeable (see Figure 3.14d). Note that swapping the VDD and GND signals in any CMOS based logic produces the complement of the original function at the output. However, full voltage level at the output will not be achieved due to the presence of PMOS in the pull-down network or NMOS in the pull-up network. Consequently, key-bits can be formalized via only gathering the $\overline{key}$ and key signals to a wire with an inverter. As a result, ICs can be encrypted by exchanging some logic gates in the original circuits with different polymorphic logic gates with much less area and Power and Delay Product (PDP) penalties, instead of incorporating XOR/XNOR gates or multiplexers as key-gates, which increase the performance overhead extensively as in [69]. Different functionalities with the same structure using CMOS could also be accomplished, but at the penalty of larger number of transistors as mentioned in Table 3.2. Recently, many spintronic device-based PLGs have been presented as an alternative to conventional CMOS-based PLGs to accomplish a smaller area and lower power consumption. The first spintronic device-based PLGs have been introduced using ASLD [89]. The new generation of spintronic devices (DWM, GSHE, and MTJ) have been leveraged to also produce PLGs. These new generations of devices require lower power and smaller area compared to traditional ASLD. Examples of these devices are Domain wall motion (DWM)- [154, 155], giant spin-Hall effect (GSHE) switch- [156, 157] and hybrid SHE-Magnetic Tunnel Junction (MTJ)-based PLGs [158]. Unlike ASLD, these devices (DWM, GSHE, and MTJ) require the controlling CMOS transistors (for write/read) that essentially impose excessive area, power, and most importantly a two-cycle delay operation to convert and transfer the spin current (Is) to the next device (next PLG) [159]. This one is true because the Is in these new generations of spintronic devices is very small and cannot be used to drive the next PLG. In the first cycle, the CMOS circuit is required to realize a voltage divider with a sensing amplifier to read-out the output voltage (as '0' or '1'), and the

second one is needed to convert the output voltage from the first CMOS circuit to current (+ or - current) and then supply this current to the next state (next PLG). Therefore, the delay and power overheads will be large after adding these two circuits, and the whole design works as a sequential circuit. Improvements on ASLD-based PLGs have been achieved to significantly reduce the energy dissipation (power-delay product) and the required area via using optimal ASLD parameters, using the pipeline technique, and adding ASL-buffers at necessary locations throughout the design [160]. It is worth mentioning that the new generations of spintronic devices are slower than CMOS by about 20X before adding the two CMOS circuits. Therefore, these devices may be good in some applications that do not require high frequency.

3.2.2 SiNW in Logic Encryption

A design could be encrypted via inserting different types of key-gates though different locations in an original circuit, such as look up tables (LUTs), multiplexers, XOR/XNOR and AND/OR gates. The locked chip with XOR/XNOR insertion is stronger against the most serious threats [74] than any other types of key-gates. However, building an XOR/XNOR gate requires a higher number of transistors than other gates, such as AND, OR, etc. As a result, the performance overhead will elevate significantly, especially for a small scale circuit (<800 gates) where the power and area overheads might override the original circuit size. For instance, by adding few XOR/XNOR key-gates (less than 5% of the total number of gates in an original circuit), the penalty of the power and area is approximately larger than 31% and 20% for the majority benchmark circuits, respectively [92]. It is worth mentioning that this amount of adding ratio is not enough to prevent the brute force attack, where the key-size should at least be larger than 64 bits [68]. With the scaling of CMOS technology, it becomes more expensive to achieve similar security level by compromising the performance. Due to the defects of existing work, we would like to present our improved method to implement logic locking using emerging transistors.

3.2.2.1 Fundamental of Logic Locking

A simple demonstration of logic locking is shown in Figure 3.15. The original logic gate is two-input AND gate. An exclusive-OR gate is further added to combine the original output f with a locking enable signal k. Then, the locked netlist consists of two logic gates, AND gate and XOR gate, respectively. The locked Boolean logic function is given in Equation 3.3.

$$f_{lock} = f \cdot \overline{K} + \overline{f} \cdot K, \quad f = ab. \tag{3.3}$$

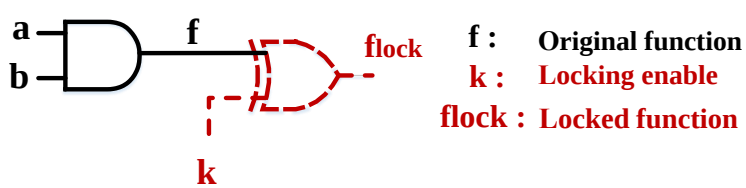


Figure 3.15: A simple example of logic locking.

When $K = 0$, it functions as the original AND logic gate. Meanwhile, when $K = 1$, it locks the original AND gate and works as a NAND gate. With triggered key ($K = 1$), the output will report all the four input vectors as failing patterns. It is important to note that $K = 1$ is not dedicated to lock the function. For instance, when an XNOR gate is incorporated, the locking key is switched to $K = 0$. The choice of either XOR or XNOR gate relies mainly on the definition of key value, where normally $K = 1$ is more favorable. Furthermore, the key-bit (K) could be configured as one or zero (based on the designer's desirability). For instance, if the inserted key-gate is XOR, K should be set as zero to recover the correct functionality. However, one can configure such key to one for the correct functionality by only adding an inverter before or after the inserted XOR key-gate. Chakraborty et al. [90] introduced a methodology of defining logic cone, in which more

logic elements are included so that the number of failing input patterns will increase accordingly. This scenario will not be covered in our work due to the larger area overhead.

3.2.2.2 Encrypted Logic Circuit Leveraging Polymorphic Logic Gates

Since inserting key-gates that are designed using traditional CMOS technology for logic encryption purposes leads to a high performance overhead, our technique is to select gates in an original circuit that have a high impact on output and then exchange them with polymorphic logic gates designed using SiNW.

A simple example of obfuscating a circuit using our proposal is shown in Figure 3.16. The original design has two 2-AND, 2-NAND, and 2-OR gates with five primary input and two primary output signals as demonstrated in Figure 3.16a. To encrypt this circuit, a designer can replace one OR and one NAND gate with AND/OR and NAND/NOR polymorphic logic gates, respectively, as shown in Figure 3.16b. The two polymorphic gate keys, referred to as ($K1$) and ($K2$) in Figure 3.16b, are specified as "00" to recover the correct functionality. For any other $K1$ or/and $K2$ value(s) (incorrect key values), the encrypted design will produce the wrong output. An attacker cannot know what the original gates are before the replacements since the original gates before the exchanged AND/OR polymorphic gates could be either AND or OR gates, and they could be either NAND or NOR gates before the exchanged NAND/NOR polymorphic gates. Note that each of the two incorporated polymorphic gates has an inverter (to create a uniform key-bit as mentioned in Section 3.2.1). As a result of using this approach, the performance penalty should be much less than inserting XOR key-gates.

The locked design should produce corrupt outputs for most of the combination incorrect key values. Otherwise, the encryption technique will be vulnerable [92] to an attacker who might figure out the correct functionality. Consider the same encrypted circuit in Figure 3.16b. On applying input

pattern “00110”, the correct output of the circuit, which is “10”, will be revealed once the correct value “00” of $K1$ and $K2$ is supplied . In contrast, the design will produce incorrect outputs “01” at F1 and F2, respectively, if both $K1$ and $K2$ values are “11”, and therefore the Hamming distance between the correct and corrupt outputs will be 100%. In this case, the first polymorphic gate switches from original OR to AND gate, and the second one switches from NAND to NOR gate. Moreover, if the value of either $K1$ or $K2$ is ‘1’, the output signal of F1 and F2 will be either “00” or “11”, respectively, where for each case the Hamming distance will be 50%.

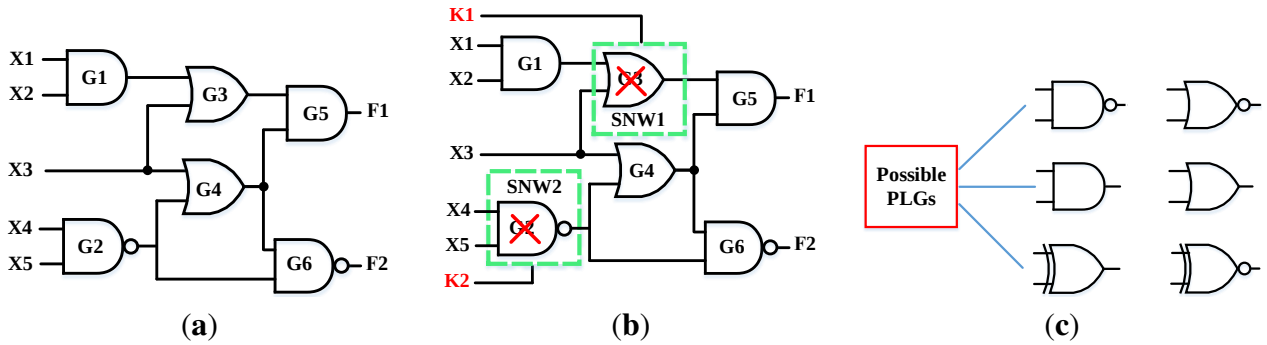


Figure 3.16: An example of encrypted a circuit using polymorphic logic gates designed using SiNW FETs (**a**) an original circuit (**b**) encrypted circuit via exchanging some gates in the original circuit with polymorphic logic gates, where both of AND/OR and NAND/NOR polymorphic gates are incorporated) (**c**) three possible polymorphic logic gates produce six different logic gates.

In additional to these two polymorphic gates, another XOR/XNOR polymorphic gate is designed as shown in Figure 3.16c. The XOR/XNOR polymorphic gate could be swapped to XNOR/XOR gate. Adding more reconfigurable gates is important to increase the ambiguity of an attacker from identifying or comprehending the structure of the original circuit. The three possible aforementioned polymorphic gates have been leveraged for the encryption purposes. The detailed security evaluation will be discussed further in the following section.

3.2.2.3 Security Metrics

Before the discussion of the detailed implementation, it is essential to explain the security metrics on evaluating the proposed logic locking technique.

As expected, the attacker is not aware of the key values for encryption and decryption. An extensive test plan might be launched in order to retrieve the correct keys from the attackers' perspective, thereby decrypting the protected IP. Certainly, increasing the key size can increase the effort of an attacker. By applying the wrong key values on the encrypted design, the attacker will get wrong outputs.

To further formalize the security metric, we assume, as the authors in the fault impact analysis assumed [69], that the IC design consists of T primary input bits, Y primary output bits and M encryption key bits. Let $N = \{0, 1\}$. Assume that a valid input $x \in N^T$ and a corresponding correct output $z \in N^Y$. Let $k \in N^M$ be the correct key values. A function f with encryption variables should be defined as two scenarios:

- On employing the valid secret key k, the function produces correct outputs for all input test patterns.
- On employing the incorrect secret key values, the function generates wrong outputs correspondingly:

$$\mathbf{f}(\mathbf{x},\mathbf{w}) = \begin{cases} \mathbf{z} \forall \times \in \mathbf{N^T}, \mathbf{z} \in \mathbf{N^Y}, & \textbf{when } \mathbf{w} = \mathbf{k}, \\ \mathbf{z'} \forall \times \in \mathbf{N^T}, \mathbf{z'} \in \mathbf{N^Y}, \textbf{but } \mathbf{z'} \neq \mathbf{z}, \textbf{when } \mathbf{w} \neq \mathbf{k}. \end{cases} \tag{3.4}$$

To define the security metrics, Hamming distance (HD) has been commonly adopted. The defini-

tion of Hamming distance is a number used to denote the difference between two binary strings. By that means, the wrong output z' can be quantitatively differentiated from the correct output z by applying HD measurement. For instance, when $HD(z, z') = 0$, the corner case shows that the outputs of encrypted netlist function independently of the locking key. It indicates that the applied encryption is drastically weak. If $HD(z, z') = Y$ (the number of output bits), z' is complementary to z, which is also weak in case an attacker tries to reverse the output value [69].

Consequently, it is substantial for the defender to identify the system and define the encryption mechanism such that the attacker is unable to recover the correct functionality. With the minimized correlation between the wrong and the correct outputs, a maximum ambiguity can be generated for the attacker. Let B be the number of output-bit combinations corresponding to certain HD between the correct and wrong outputs. If $HD(z, z') = P$, then B is calculated as $\binom{Y}{P}$.

Similar to cryptography, a larger B would imply greater ambiguity, thereby improving the robustness. Clearly, B is maximum when $P = Y/2$ (or $HD(z, z') = Y/2$). Therefore, the security metric for logic locking/encryption technique should be defined in a way that the Hamming distance is evaluated between the output bits by employing the correct key values and the wrong key values. A Hamming distance of half of the output-bit number (HD = $Y/2$ or 50% of Y) indicates the most robust implementation.

3.2.2.4 Algorithm for Insertion of Polymorphic Logic Gates

It is substantial to formalize the previous analysis into a universal method. Algorithm 1 is proposed to choose the optimized locations for incorporating polymorphic logic gates.

In general, the algorithm has two inputs-netlist and keysize, while the output is the locked netlist with inserted key. The encryption algorithm starts with inputting one key bit into an original netlist.

Algorithm 1 Logic Locking Algorithm

```
1:  Input: Netlist, keySize
2:  Output: Encrypted netlist with key
3:  for i ← 1 to KeySize do
4:      for each gate_J at the output of the netlist do
5:          Call the GATE and update the netlist;
6:          if Corrupted Output ≥ (Threshold) then
7:              Call the CAL_HD;
8:          else
9:              Call the GATE;
10:         end if
11:     end for
12:     for each gate_k ∈ netlist do
13:         Call the GATE and compute the corrupted output;
14:     end for
15:     Select the highest impact gate on output;
16:     if (KeySize = MAX) or (Inc HD = smallest) then
17:         Call the GATE;
18:         Terminate;
19:     end if
20:     Call CAL_HD
21: end for
22: function CAL_HD:
23:     Increment i;
24:     Accumulate the corrupted output;
25:     if HD == 50% then
26:         Terminate;
27:     else if HD > 50% then
28:         Compare the HD at exchanging gate_{J-1} with gate_J;
29:         Select the exchanging gate that is closer to 50 % HD;
30:         Terminate;
31:     end if
32: end function
33: GATE: case (gate):
            {
            NAND ⟺ NOR ;
            OR ⟺ AND ;
            XOR ⟺ XNOR ;  }
```

Each selected gate close to the output will be calculated regarding certain test patterns. If incorrect output bits are 50% different from correct output bits, i.e., HD = $Y/2$, the algorithm will terminate and output the encrypted netlist. When HD = 50% is not satisfied for gates close to output, the selection will iteratively go over the remaining gates in the netlist and calculate the highest impact

(HD = 50%).

Note that two conditions are required, increasing-rate HD $\leq 0.01\%$ and KeySize == MAX (MAX is 128 bits in this paper), respectively. When HD is increased by 0.01% every iteration, we will terminate the program. The reason is because HD almost hits the limit, and it merely adds extra overhead by incorporating more encryption key. The for loop continues incrementing the key size until a desired HD is satisfied.

Two functions CAL_HD and GATE are also attached following the abstracted main pseudocode. CAL_HD enables the computation of Hamming distance, while GATE selects the potential exchanging gates. As mentioned, three different polymorphic logic gates are employed, resulting in six various cases.

3.2.3 Results

3.2.3.1 Experimental Setup

In this section, we provide empirical results regarding the implementation penalty and the security level of the proposed approach. The effectiveness of our proposal has been evaluated using combinational benchmark circuits from ISCAS'85 benchmark suites [161]. We leverage the Synopsys Hailey Simulation Program with Integrated Circuit Emphasis (HSPICE) for the circuit simulation to design and simulate the SiNW based polymorphic logic gates. Afterwards, the Java language is utilized to implement the algorithm of the proposed logic locking technique. One thousand random input patterns are applied to the encrypted netlist to further evaluate the Hamming distance. The Synopsys Design Compiler, including both silicon nanowire 20 nm and CMOS 20 nm technologies, is used to further evaluate the performance overhead of all ISCAS'85 benchmark circuits.

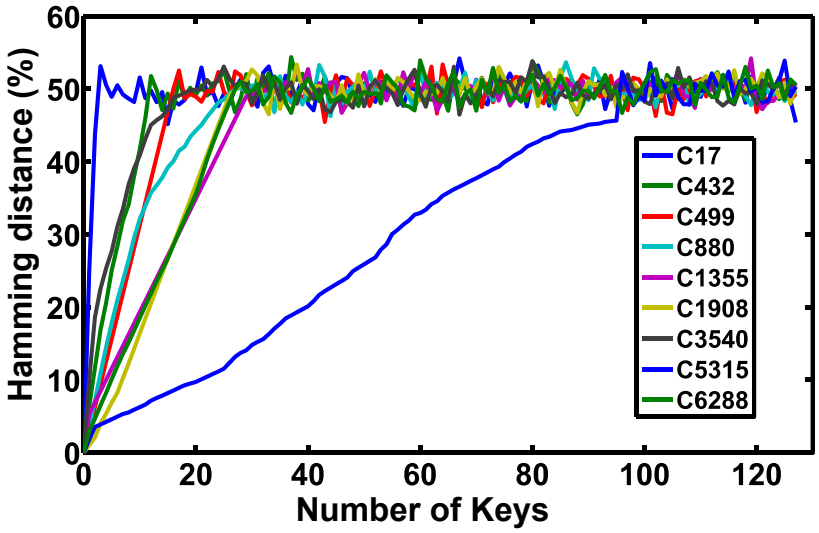


Figure 3.17: Hamming distance of ISCAS'85 (International Symposium on Circuits and Systems) benchmark circuits.

3.2.3.2 Security Evaluation

To evaluate the security of logic locking, a Hamming distance based metric is mostly applied in [22, 69, 92, 147]. Figure 3.17 shows the Hamming distance analysis of ISCAS'85 benchmark circuits using our proposed algorithm. Approximately 50% Hamming distance is achieved for all benchmark circuits. The slope of the traces implies the effectiveness of logic locking technique. If the slope is steeper, a smaller amount of key gates is required for encryption purposes, thereby reducing the performance overhead.

The majority of benchmark circuits hits the 50% mark in less than 40 key gates, except for one outliner C5315, which needs 95 key gates. Furthermore, as shown in Figure 3.17, when an encrypted circuit reaches 50%, its Hamming distance value does not swerve more by incorporating more key gates. In other words, the minimum number of key gates for achieving 50% HD is defined as the encryption threshold. The defender can intentionally increase the key gates for extra obfuscation

without changing the robustness of the logic locking.

Table 3.3: The number of polymorphic logic gates to achieve 50% Hamming distance using our proposed scheme compared to previous techniques.

Benchmark Circuits	**# of XOR PLGs**	**# of NAND PLGs**	**# of AND PLGs**	**# of XOR/XNOR Gates [68, 92]**	**Hamming Distance (%)**		
					Ran [68]	**FA [92]**	**PLGs**
C17	-	3	-	6	42	51	53
C432	-	10	1	17	29	50	50.06
C499	16	-	-	40	26	50	50
C880	-	18	13	28	19	50	48.3
C1355	-	32	1	42	26	50	50
C1908	-	23	4	28	26	50	49.9
C3540	-	8	13	22	23	50	49.9
C5315	-	4	91	97	15	44	45.6
C6288	-	26	1	27	32	50	50

Table 3.3 shows the detailed results of security evaluation. The previous random and fault analysis-based logic encryptions are included for the comparison. The number of required key gates using a polymorphic logic gate is listed between the second and fourth columns. The fifth column shows the number of required XOR/XNOR gates used in previous random and fault analysis works. It is apparent that our proposed technique embraces more variants for key gates besides XOR/XNOR gates. NAND/NOR and AND/OR based polymorphic gates virtually are more favorable for most benchmark circuits. It can be seen that the required number of the polymorphic logic gates is less than the conventional XOR/XNOR based key gates, which implies the effectiveness of our proposed technique. The last column of Table 3.3 shows the achieved Hamming distance using our technique, where 50% HD is mainly accomplished. Only benchmark circuit C5315 with 45.6% HD is better than both random and fault analysis based methods.

3.2.3.3 Performance Overhead

As we discussed, our proposed polymorphic gates mechanism should display a dramatic advantage in less performance overhead, i.e., area and power-delay product overheads, mainly resulting from the polymorphic gates not adding additional logic gates into original circuits. However, it is expected that our technique would incur certain performance overhead, since SiNW FET is more energy-hungry than its CMOS counterpart due to the unique polarity controllable feature of the emerging device.

Figure 3.18 shows the area overhead of all benchmark circuits with logic locking technique. The number of logic gates corresponds to the results listed in Table 3.3. Similar to the previous work [92], we do not include the overhead of peripheral circuits, such as key-bit generator. Apparently, the polymorphic gate based logic locking has drastically lower area consumption than the other two techniques. When the circuit scale increases, the overhead is merely negligible for our proposed technique. C499 circuit has almost zero area overhead, mainly because the key gate is an XOR/XNOR polymorphic gate, which has less area for SiNW FET than for CMOS.

Figure 3.19 shows the power-delay product (PDP) penalty of all benchmark circuits. It maps to the number of gates added for encryption listed in Table 3.3. Except for the C499 circuit, all benchmark circuits are more favorable to NAND/NOR and AND/OR polymorphic gates. It is obvious that the polymorphic gate based logic locking hardly provokes any overhead on power-delay product, where $<1\%$ overhead applies to every benchmark circuit. On the other hand, random and fault analysis encryption techniques display a considerable power-delay overhead upon original circuits, where $>25\%$ penalty occurs at the majority of benchmark circuits.

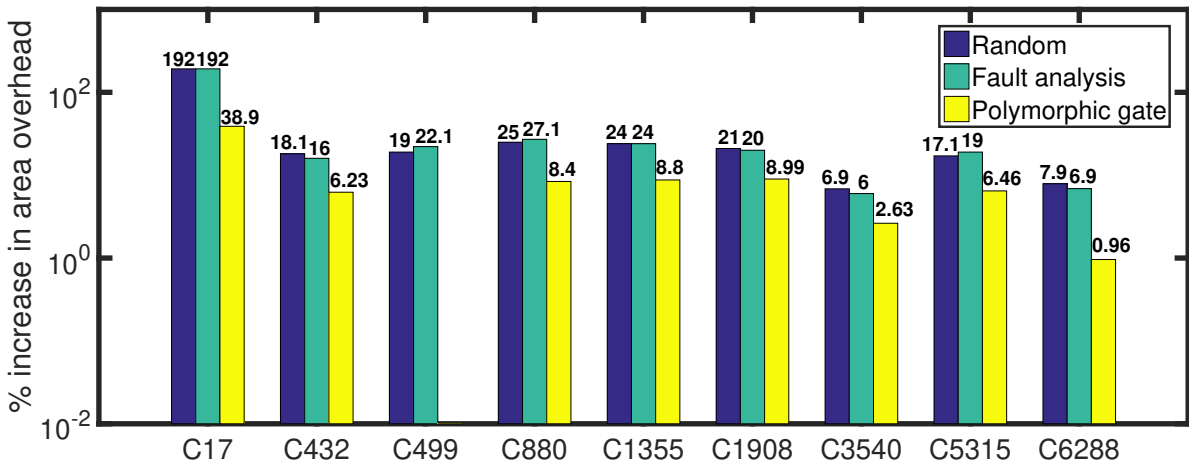


Figure 3.18: Area overhead of random, fault analysis and polymorphic gate based logic locking.

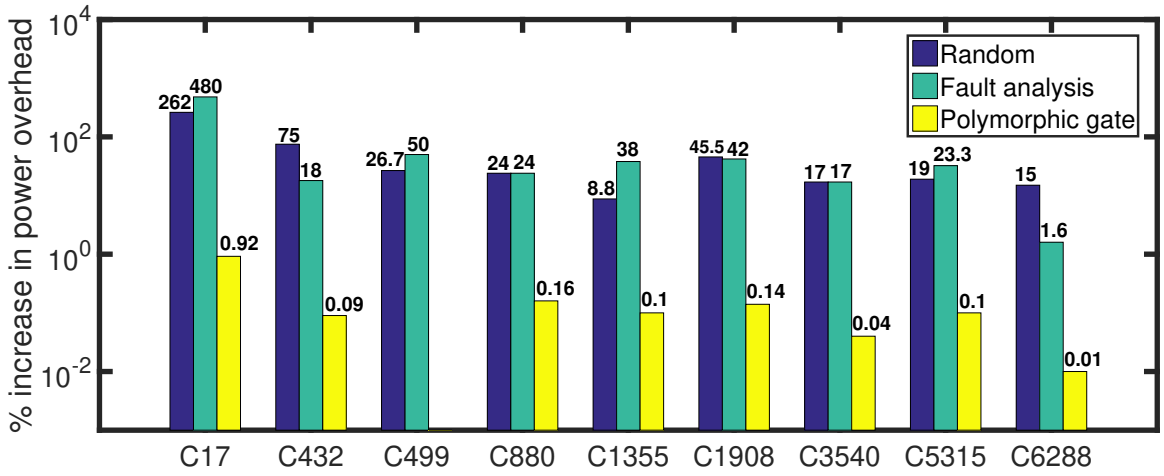


Figure 3.19: Power-delay product overhead of random, fault analysis and polymorphic gate based logic locking.

3.2.4 Discussion

3.2.4.1 Attacker's Perspectives

The goal of an attacker is to expose the secret key of an encrypted circuit. Once the key is revealed, there is no meaning for the encryption since with the correct key the attacker can copy an IC, insert a hardware Trojan, and/or overbuild an IC illegally without designer's license. The most serious attacks and remedies on logic encryption techniques are discussed below.

3.2.4.1.1 Applying Brute Force Attacks

An assailant could expose the valid key of an encrypted circuit by applying all possible cases of the key-bits unless the key-size is long enough. A defender can prevent the brute force attacks via increasing the key length. In the XOR/XNOR insertion approaches, enlarging the key-size means increasing the number of the injected key-gates since each new key-bit is an input of each added key-gate leading to an increase in the performance overhead substantially. In our proposal, besides the smaller performance penalty due to the exchanging gates, a hardware engineer can freely enlarge the key-size to a maximum of two times if the key-bits are not gathered to a line with an inverter for each exchanged gate, as demonstrated in Figure 3.14.

3.2.4.1.2 Can An Attacker Identify The Key Since The SNW Structure Is Different from CMOS Structure?

Unlike traditional logic locking using XOR gates, the polymorphic gates used as 'lock' gates can be easily identified by an attacker in the GDSII of the chip, due to the difference in their lay-out/fabrication features as compared to the rest of the chip (conventional CMOS). Though this

may not completely compromise the security of an IP, it certainly makes it easier for an attacker to identify the 'key' if he/she has access to an unlocked chip, because then the attacker knows exactly where and what to look for. However, the attacker cannot get the key even the structure of the SNW is different from CMOS for two reasons (1) In SNW, the same structure can give different functionalities based on the value of the key. A designer can randomly exchange more gates through the circuit without changing the value of the internal net in the original circuit since the overhead is very small to maximize the ambiguity of the attacker. (2) the only difference between the structure of CMOS and the SiNW is the polarity gate, which is only a few nanometer. So, it is really difficult for an attacker to figure out the difference for a large scale circuit.

3.2.4.2 Key Generations

Previously, Rajendran et al. [92] applied a Physically Unclonable Function (PUF) circuit and Rivest–Shamir–Adleman (RSA) encryption unit to generate the keys for logic encryption. However, area consumption of the two cryptographies might override original netlist with only hundreds of logic gates. For instance, ISCAS'85 C17 to C1355 circuits have less than 400 logic gates. To tackle this issue, we adopted the common encryption technique in system on chip (SoC) design, called dynamic scrambling [151]. The encryption and decryption mechanisms for the key generation are presented in Figure 3.20.

On the rising edge of a Fetch operation (i.e., for a new instruction), the random generator sets a new scrambler configuration. This configuration is saved in a new segment of memory given by a first input first output (FIFO) and the address used is saved with the scrambled data in random memory. Concurrently, each time a new configuration is requested to unscramble data, the configuration is read in the memory at the address given by the random memory and the data is unscrambled. This address is saved in a FIFO that stores all empty memory addresses. When a configuration value is

read, the memory block that holds the value should be overwritten to avoid risk of reuse.

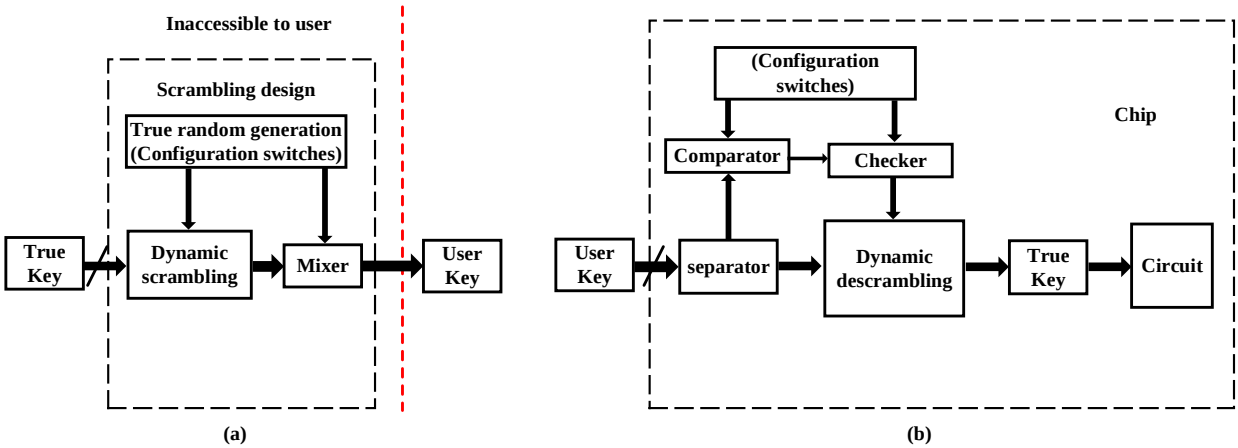


Figure 3.20: Scrambling technique used for (**a**) scrambling and (**b**) descrambling.

More specifically, the designer provides the input of the dynamic scrambling by the secret key of the circuit, and the random generation is used to generate new configuration bits. The scrambling output bits are randomly combined with the configuration bits via a Mixer to produce the user key. The end-user uses this key to decrypt the circuit, where the user key will be separated into the configuration bits and the scrambling output bits. The incoming configuration bits will be compared with the configuration bits in the chip that the designer already burns in a non-volatile memory and the rest of the incoming bits (scrambling output bits) will be fed as inputs to the dynamic unscrambling. In this case, the encrypted chip will only be activated by the secret key if both of the comparator output and the unscrambling key are correct.

3.2.4.3 Testing in an Untrusted Foundry

Since the complexity of the design becomes very large and needs different types of equipment as well as fabricating process involvement [162], many companies design the ICs and then fabricate

them at other companies. Thus, ICs might be imitated by the untrusted company so the company could sell them in the markets illegally or insert a Trojan inside the chips [8]. As a result, the developers have no control over untrusted foundries to protect their designs, where they are susceptible in the face of several attacks [16]. IP owners can protect their design from counterfeit ICs and other attacks in a company during the test using the Secure Split-Test (SST) method before sending the ICs to the trusted facility for configuring its functionality [24]. SST protocol is based on communicating and exchanging generated keys between the foundry and the designer, where only the IP owner can know whether the IC is passing the test successfully or not. An improvement on SST is achieved, namely CSST, which gives a simple communication between the IP owner and the foundry as well as providing more protection than the traditional SST. In this technique, the designer has full control over the chip, and only he or she can understand and analyze the result of the locked chip [163].

3.2.4.4 Beyond SiNW FETs

Besides the proposed SiNW FETs, other emerging transistors might also be employed to protect IP designs. For instance, the recently proposed negative capacitance FET (NCFET) [164] is embedded with the property of tunability. By adding a ferroelectric layer in the gate stack of a MOSFET, NCFET is able to reduce the switching slope to a value less than 60 mV/dec, which shows potential for ultra low-power design. Meanwhile, since it can be configured in a way that may or may not have hysteresis loop, one NCFET can virtually function in two modes: memory cell and Boolean logic cell. The difference between two modes is determined by the thickness of ferroelectric layers, which is on the sub-nanometer scale. Once the NCFET fabrication is done, it is extremely difficult to distinguish which mode NCFET stays because the reverse engineering cannot have the advanced SEMs to identify the devices.

3.3 Proposed Techniques against Advanced Attacks

3.3.1 Approach 1 against Sensitization Attack

Sensitization attack can easily propagate the values of the secret key to outputs since the MUXes in the half and full MUX insertion techniques are inserted close to the outputs. Therefore, the key values can be revealed with less computational time compared to other techniques, e.g. random insertion-based or fault analysis-based encryption. However, the full protected chip, e.g. MUX insertion with HLU and LFSR, makes the execution time of the sensitization attack exponential with the number of key-user since the values of the MUXes are not directly connected to the key user. The relationship between the values of the MUX key and the user key are not linearly. Therefore, this technique can successfully prevent this attack.

3.3.2 Approach 1 against SAT Attack

The full protected chip of half and full MUX insertion can provide so resilience to SAT attack, but since the size of the key user is not long enough, SAT attack may break the design. Our future work is to fully test and develop this technique against SAT attack.

3.3.3 Approach 2 against Sensitization Attack

In our proposed encryption-based SiNW polymorphic gates, even though sensitization attacker cannot propagate the key values of the encrypted circuit to output, he or she may sensitize a path from the output of a polymorphic gate to an output on the working device, and, therefore, the logic function may be determined by applying different patterns to the polymorphic gate's input (using ATPG). Once the logic function is determined, a key bit guess can be made on an unprogrammed

device and the same vectors run again. If the output remains the same, then the key bit guess is correct; otherwise, the opposite value must be the correct assignment. Therefore, one needs to employ 'interference graph' to make the task difficult [2].

3.3.4 Approach 2 against SAT Attack

We tested our SiNW PLG-based traditional logic encryption against SAT attack. Unfortunately, the SAT-attack took less than two seconds to break the c7552 circuit with less than 60 iterations. The only way to strongly prevent this attack is to incorporate untraceable resilient SAT circuit into the encrypted circuit. More information regarding this matter will be discussed in the incoming chapter.

3.4 Summary

In the MUX insertion technique, multiplexer insertion-based logic encryption has been presented. Compared to previous literature, the fault impact analysis approach will not guarantee achieving a 50% Hamming distance for any circuit, and the execution time of its algorithm is very long and unacceptable in practice for a large chip. On the other hand, our methodology can accomplish 50% (or close to it) Hamming distance between the corrupted and corrected outputs, even if one bit in the user key is incorrect, unless all of the right valid key bits are supplied, and it is very fast. Moreover, instead of using both RSA cryptography and PUF, we employed the HLU and the LFSR random generator to protect the secret key and generate random keys with 0.5 probability, respectively. The power, area and delay overheads are gradually decreased for a large circuit that has appropriate output bits, such as C6288 and S9234. In conclusion, our proposed technique can outperform the previous state-of-the-art work in terms of less performance overhead while

achieving a higher security level.

In the hybrid SiNW-CMOS based traditional encryption, we have demonstrated that the usage of emerging transistor, i.e., SiNW FETs, can help improve the logic locking design by preserving lower power and area consumption compared to conventional CMOS technology. Specifically, the SiNW-based polymorphic gates work as logic key units to encrypt the combinational circuits. A smart placement algorithm is formalized and it shows that 50% of Hamming distance between the correct and wrong output bits can be achieved through a security assessment. We showed that, besides the traditional criteria for emerging devices such as area, power, delay and non-volatility, security may serve as a new criterion to thoroughly judge the pros and cons of any emerging devices. Using this new standard, we plan to revisit existing emerging transistors to have a full comparison between emerging technologies and CMOS technology. Meanwhile, we believe that more research outcomes are expected in this area where unique properties of emerging transistors can help in enhancing the circuit security.

CHAPTER 4: STRONG LOGIC OBFUSCATION AGAINST ADVANCED IC REVERSE ENGINEERING ATTACKS

4.1 Strong Encryption Against Reverse Engineering Attacks

4.1.1 PLG-based Fast Traditional Logic Encryption

It is well known that a 2-input XOR gate consumes at least eight transistors in conventional CMOS technology. The large size of a XOR gate consumes more area and is power hungry compared to other basic logic gates, such as AND/OR or NAND/NOR gates. As presented in previous works [1, 68], the power and area overheads are typically high, while the overheads for more secure design, e.g. SLO technique, are even higher than other logic locking techniques when the number of inserted key-gates (XOR/XNOR) is about 5% of the total number of gates in ISCAS'85 benchmark circuits [1]

As mentioned, the large overhead of conventional logic locking may preclude its application on silicon chips. To further reduce the surcharge, instead of incorporating more key-gates (e.g., XOR/XNOR gates [68]) for logic encryption, our technique is to replace some gates in the original netlist with SiNW based PLGs. To fully elaborate our proposed technique, C17 benchmark circuit is adopted. Original C17 circuit consists of six NAND logic gates with five primary inputs and two primary outputs shown in Figure 4.1 (a). As explained at the beginning of this subsection, a SiNW NAND gate can be easily switched into SiNW NOR gate using polymorphic technique. Consequently, we replace two original NAND gates by two PLGs, which each has an extracted locking key as in Figure 4.1 (b). It is important to note that the replacement does not add much overheads.

[1] © 2017 IEEE. Part of this chapter is reprinted, with permission, from [87]

The added inverter wiring is very small compared with the inserted XOR/XNOR as a key-gate. We then define that when K1 and K2 are both zero, it is the original netlist with the correct outputs. Otherwise, the PLG switches to a NOR gate, most likely resulting in wrong outputs.

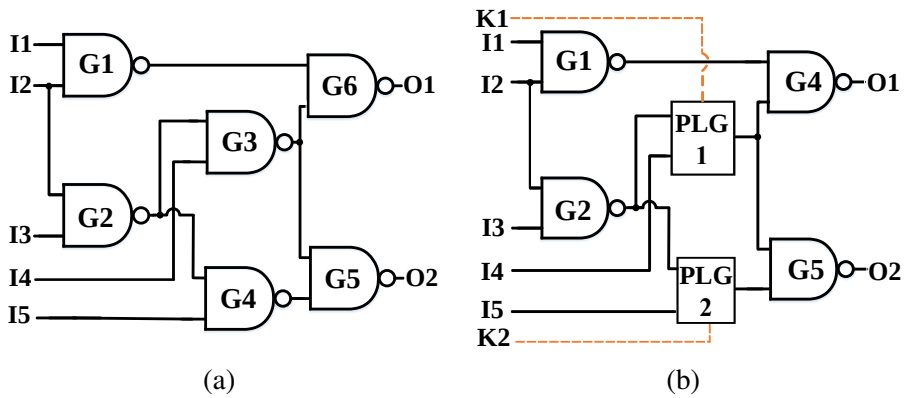


Figure 4.1: PLG based logic locking technique: (a) original C17 circuit, and (b) modified C17 with two NAND/NOR PLGs.

For instance, applying the input pattern of "01000" with $K1 = 0$ and $K2 = 0$, a correct output "00" is produced for the C17 circuit. When $K1 = 1$ and $K2 = 1$, the two PLGs switch from the original NAND gates to NOR gates, consequently resulting in "11" at the outputs. However, when one PLG encrypts with either $K1 = 1$ or $K2 = 1$, wrong outputs "01" or "11" are produced, respectively. Three different key configurations generate two wrong output patterns.

We replace some gates in the original netlist with the three different PLGs, AND/OR, NAND/NOR, and XOR/XNOR. Each type of PLG is designed with a different number of inputs ranging from 2 to 8 to cover/mimic a large number of gate types in the original netlist. We replace 5% from the total number of gates (for small circuits) and 100 gates (for large circuits) in the original netlist with three different types of PLGs. Out of those gates, 75% of them are AND/OR, 20% of them are NAND/NOR, and 5% are XOR/XNOR gates, starting from small gate input sizes (instead of

starting from large gate input size as in [87]) to further reduce the overhead surcharge. If the original circuit does not have a gate type equivalent to one of these three PLGs, its specified ratio will be distributed between the rest of the PLGs. Among all replaced gates in the original circuit with PLGs, we replace the largest ratio of AND/OR gates to increase the ambiguity of an attacker from identifying the location of URSAT scheme since a large number of AND/OR PLGs has been used to build URSAT.

Another two SiNW based PLGs, Inverter/Buffer (I/B) and AND/NAND, have been implemented to build the RSAT block and prevent an attacker from identifying/tracking the RSAT block/its output signal. More information regarding this matter will be clarified in the next section. Note that I/B and AND/NAND PLGs could also be used to help encrypt a netlist through exchanging inverter/buffer and AND/NAND gates in the original circuit with the I/B and AND/NAND PLGs, respectively. Also, note that, *in our previous work [88], we only used three simple PLGs to encrypt a netlist. These PLGs cannot be used to cover/mimic all gate types and sizes in the netlist as in this work, and therefore the work in [88] cannot be used to encrypt all logic circuits. Moreover, the work in [88] is vulnerable to IC reverse engineering attacks.*

4.1.2 Untraceable Light-weight Resilient SAT (URSAT)

We implement a small light-weight circuit using SiNW FET based PLGs, namely URSAT, and add a reconfigurable signal (R-S) to prevent SAT, D-DIP, AppSAT, Bypass, and removal (SPS and TS) attacks. The R-S is a wire coming from an internal net in the encrypted circuit and connected to the key input of the PLG in URSAT. First, we design two additional types of PLG; inverter/buffer (I/B) and AND/NAND PLGs. The I/B PLG is important to reduce the performance penalty, instead of using XOR/XNOR as a key-gate, and the AND/NAND PLG is important to prevent SAT attack. Otherwise, SAT attack will rule out more than one key-bit at each iteration, leading to a reduction

in the iteration number by about $2^n/2$, where n is the number of baseline keys. More information will be covered in section 4.2.2.

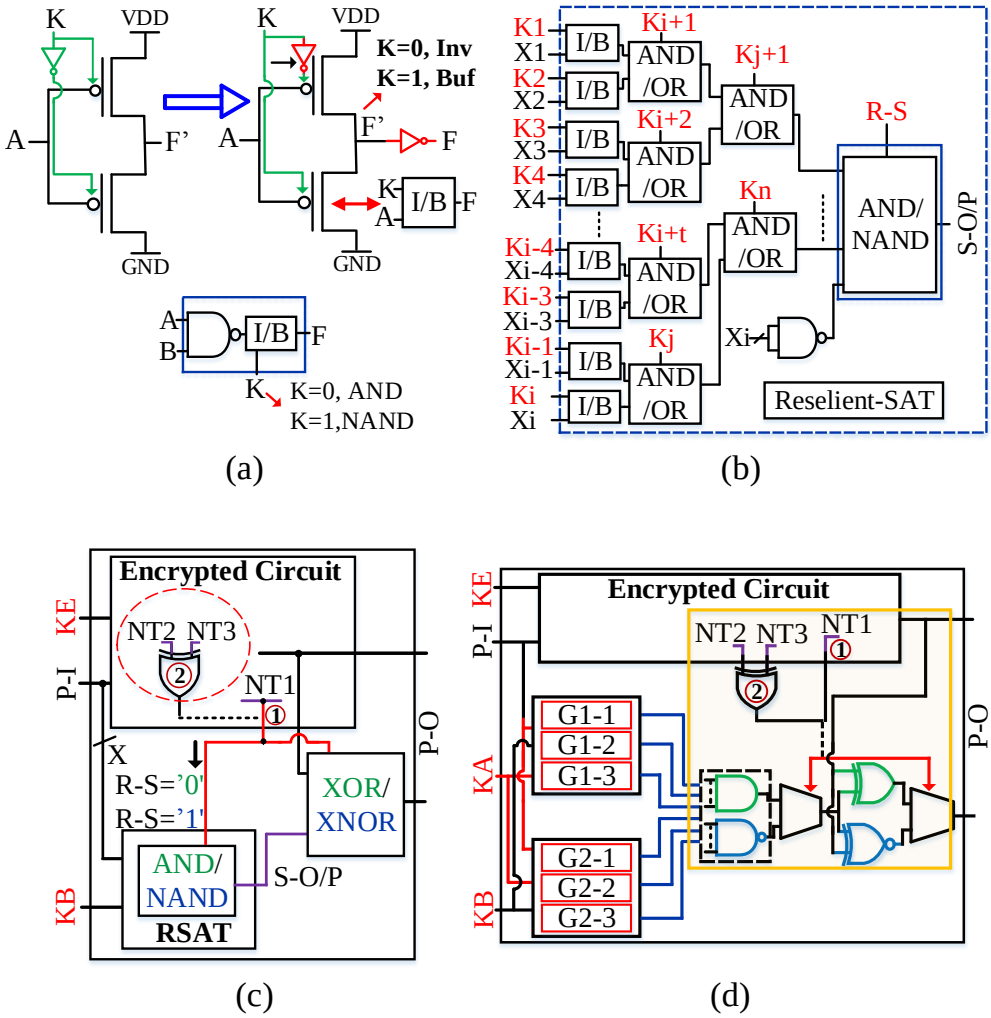


Figure 4.2: (a) Two more PLGs, I/B and AND/NAND PLGs, (b) URSAT circuit, (c) integrating URSAT to the encrypted circuit with R-S based on two different approaches to prevent removal (SPS and TS) attacks, and (d) preventing SPS and TS attacks from tracking and detecting Anti-SAT block.

We combine one CMOS-inverter with SiNW to produce I/B PLG, as shown at the top-left of Figure 4.2 (a). I/B PLG works as an inverter when the key value is '0', otherwise it performs as a

buffer. Afterwards, this obfuscated gate is connected with a NAND gate to produce AND/NAND PLG, as depicted at the bottom of Figure 4.2 (a). AND/NAND PLG performs as an AND gate if the key value is '0', while it works as a NAND gate if the key value is '1'. Also, an AND gate can be connected with I/B PLG to produce AND/NAND PLG, via only changing the location of the CMOS-inverter to the other side, and this will increase the ambiguity of an attacker from discovering the correct key. By using these two obfuscated gates with the previously designed PLGs, we build URSAT circuit against SAT attack, as illustrated in Figure 4.2 (b). A set of I/B PLGs is inserted between the primary inputs (Xi) and the key-bits (Ki). The outputs of these I/B PLGs are fed to 2-input AND/OR PLGs, and then the outputs of the AND/OR PLGs are connected to AND/NAND PLG. The input number of the AND/NAND PLG used at the last stage in the URSAT circuit depends on the number of the coming signals from the outputs of the AND/OR PLGs in the previous stage + one signal, coming from NANDing Xi. Each incoming signal to the AND/NAND PLG comes from a sub-tree of AND/OR PLGs that has only 8 baseline key-bits. Note that the values of the internal signals in URSAT will be changeable due to the switching between AND/OR PLGs, which is impossible to be detected by SPS attack. Increasing the input number of the AND/NAND PLG at the last stage in URSAT is necessary to prevent SPS attack since it can be identified when its inputs have maximum different signal probabilities. The output signal of URSAT (S-O/P) is connected to inserted XOR/XNOR PLG with a primary output wire coming from the encrypted circuit. Also, Xi signals are NAND-ed, and the NAND output signal is connected to the AND/NAND PLG in URSAT. In this case, the iteration number of SAT attack will increase exponentially with the number of baseline URSAT key-bits.

To prevent TS attack from detecting and removing the S-O/P signal, we connect the two key inputs of the AND/NAND and XOR/XNOR PLGs to a random internal signal (NT1) coming from the encrypted circuit, as shown in Figure 4.2 (c) (highlighted as '1' in a red circle). When the value of NT1 is '0', the AND/NAND PLG will configure to an AND gate, and the inserted XOR/XNOR

PLG (between the S-O/P signal and the selected output signal from the encrypted circuit) will configure to a XOR gate (signified by the green color in Figure 4.2 (c)). However, the AND/NAND PLG will work as a NAND gate and the XOR/XNOR PLG will function as a XNOR gate if the NT1 value is '1' (signified by the blue color in Figure 4.2 (c)). As a consequence, the value of the S-O/P signal will change, based on the values of NT1, Xi, and Kn, and hence preventing TS attack. Note that, in URSAT, since I/B and AND/NAND PLGs are not used in the encrypted circuit (portions with KE), if an attacker has insight on URSAT encryption scheme, the attacker can then identify URSAT block from searching for I/B and/or AND/NAND PLGs. The URSAT logic cone can then be traced from I/B and/or AND/NAND PLG outputs and removed from the netlist. Such attacks can easily be prevented by exchanging buffer/inverter and AND/NAND gates in the original circuit with I/B and AND/NAND PLGs, respectively.

Based on the above description, if an attacker feeds the key-bits of the AND/OR PLGs with constant values or recovers the correct keys of these PLGs via using AppSAT attack, then the value of the S-O/P signal will be almost the same as the value of NT1 for a large number of URSAT key combinations. However, it is hard for an attacker to successfully identify the S-O/P signal for two reasons; first, there might be many two-signals in the encrypted circuit that have the same values. Also, to achieve this goal, the attacker has to use the brute force search which is infeasible for a large scale circuit; and second, he or she cannot know exactly which one is for URSAT, especially after the logic synthesizing. However, this assumption might help an attacker decrypt the circuit. To enhance security, an additional gate could be employed to generate a new different URSAT output signal. For example, inserting an additional XOR gate whose two inputs come from two random net signals in the encrypted circuit, and its output signal is connected to the key-input of both AND/NAND and XOR/XNOR PLGs as a R-S, as demonstrated in Figure 4.2 (c) (denoted inside a red dot ellipse in the encrypted circuit as number 2).

This approach could also be used to prevent SPS/TS attacks in Anti-SAT and SARLock techniques

with small overhead. For example, instead of connecting the incoming signals in these techniques to only one gate (last gate), these signals can be connected to an AND and a NAND gates, and the two gate outputs are fed to a multiplexer (MUX) whose output is S-O/P. Then, the S-O/P signal is connected with one of the primary outputs in the encrypted circuit to a XOR gate and a XNOR gate, and the two outputs of the XOR and XNOR gates are fed to another MUX. By connecting the two selector wires of those MUXes to either a random internal signal or an output gate whose two inputs come from two random signals, the S-O/P signal will change based on the value of the selected random signal(s). A simple example of utilizing the aforementioned technique to prevent removing Anti-SAT method that has 24 baseline keys is shown in Figure 4.2 (d). A designer can first partition each of G1 and G2 blocks into three parts, and each of 8 baseline key-bits is connected to one of these parts. In this case, the maximum skew value of each incoming signal to the last AND or NAND gate will be less than 256, instead of being approximately 16×10^6, and hence prevent SPS attack. Meanwhile, the S-O/P signal will change based on the value on the selected random net(s), resulting in prohibiting TS attack. The yellow shadow box in Figure 4.2 (d) illustrates the configuration of the S-O/P signal with the two possible connections for R-S.

It is important to mention that the output signal (F') of the I/B PLG, as demonstrated at the top-left of Figure 4.2 (a), will be slightly degraded when the I/B PLG functions as a buffer. This happens due to the fact that the pMOS is poor in the pull-down network, and the nMOS is poor in the pull-up network. However, the input voltages of the control and polarity gates are compatible with the supply voltage (VDD), resulting in making the I/B PLG fully capable for the implementation of multilevel static logic. More information regarding this matter is provided in [165]. However, in order to produce a strong non-degraded signal, we add a CMOS-inverter after the I/B PLG output and change the location of the CMOS-inverter at the key input from the gate of the lower SiNW-transistor to the upper SiNW-transistor, as demonstrated at the top-right of Figure 4.2 (a)). This new I/B PLG can be used to drive a large fanout or generate many stages of a ring oscillator.

4.1.3 Strong URSAT Using only CMOS-Logic Gates (S-URSAT)

The main idea of URSAT is to make the internal and the output signals of URSAT block inconstant when wrong keys are applied. The generated key-bits from the AND/OR PLGs in URSAT with the R-S will be able to prevent SPS, TS and AppSAT attacks from decrypting the circuit. The URSAT circuit can also be implemented using only CMOS-logic gates; however, the URSAT still has an issue, in which its output signal acts almost the same as the selected internal signal from the encrypted circuit for a large number of key combinations. Also, a large number of the AND/OR PLG key-bits could be revealed by AppSAT attack. This reduces the design resilience against AGR attack.

Instead of depending on the key-bits of PLGs to prevent SPS, TS, and AppSAT attacks, we achieve some modifications on TOA structure to produce strong untraceable resilient SAT (S-URSAT), which is able to thwart AGR as well as SPS, TS, and AppSAT attacks. The modifications are to divide TOA into many stages and connect the outputs of each stage with unique inputs to MUXes. The details of implementing S-URSAT are as follows: (1) XOR gates are inserted between the distinguishing inputs (X) and the baseline key-bits of S-URSAT. (2) Each of 4 XOR gate outputs is connected to AND gate. Then, the output of each AND gate is connected with unique primary input to a MUX, and here the first stage is achieved. Each unique input is a primary input of the original circuit that is used only one time in the whole S-URSAT design. The output signals from the first stage are propagated to next stage, and also 4-input AND gates with MUXes and unique inputs are used, until we get maximum 4-output signals. The MUX can be implemented using different kind and number of gates, e.g. three NAND gates with inverter, or two AND and one OR gates with inverter. The configuration of the key can be modified by only changing the location of the inverter or switching the order of the MUX input signals. (3) Each of 4 distinguishing inputs is ORed, and then a MUX is inserted between each OR gate output and a unique input to create

multi-stages of OR-tree. The final OR output is connected with the last generated outputs from AND tree in (2) to the last AND gate in S-URSAT. The maximum input number of each AND/OR gate in S-URSAT is 4, except the last one in each stage (which may have 5 inputs if there is one wire left from the incoming signals from the previous stage). (4) Finally, the S-URSAT output (S-O/P) is created by connecting the last AND gate output with a unique input to a MUX. Then, the S-O/P is fed with an inverted primary output, coming from the original circuit, to an XOR gate and an XNOR gate, and another MUX is added to get the encrypted output. The inverted output is obtained by flipping the gate at that primary output, which is selected randomly, e.g. if the gate at the selected output is an AND gate, it will be flipped to a NAND gate. Note that the maximum number of inputs of each AND/OR gate can be reduced to 2 inputs, except for the last AND/OR which may have 3 inputs. Accordingly, the key size of S-URSAT will increase. Figure 4.3 illustrates the details of implementing S-URSAT that has 16 baseline key-bits, and the general algorithm for the S-URSAT is demonstrated in algorithm 2.

Note that in URSAT, the key inputs of the AND/NAND and XOR/XNOR PLGs are connected to a reconfigurable signal (R-S), while in S-URSAT, each output of the AND and OR gates in each stage is connected with a unique input to a MUX that has an external key-bit. Consequently, this should be the best approach, compared to the two approaches in URSAT, since obviously the output of the S-URSAT block will be totally independent and changeable when incorrect keys are applied, as will be explained in section 4.2. The encrypted circuit will give the correct output for the combinational input patterns only when the correct key is given, and thus the S-URSAT provides resilience against existing reverse engineering attacks. It is worth mentioning that the XOR and XNOR gates with a MUX can be built with a fewer number of transistors by using only one 3-input XOR gate, where the inputs are the S-O/P, flipped primary output, and the corresponding key. Note that, the S-URSAT circuit cannot be easily identified, especially after the logic synthesizing, since it is fully interconnected with the original circuit and the XOR/XNOR and MUX key-gates can

be implemented using different basic logic gates. Also, the designer can increase the ambiguity of an attacker from identifying the S-URSAT through replacing some inverter/buffer gates in the original circuit with XOR/XNOR and MUX key-gates.

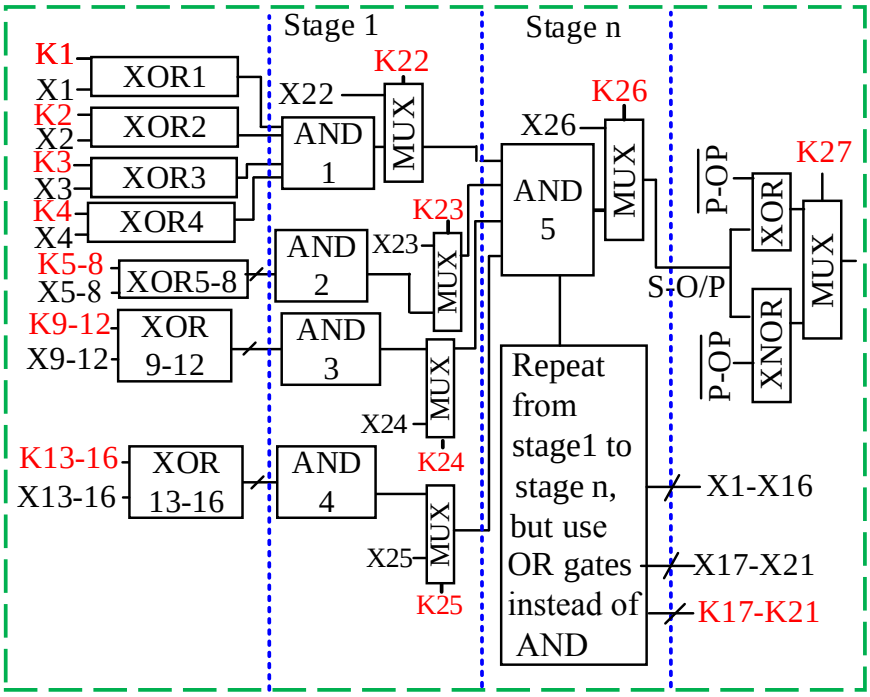


Figure 4.3: Implementing S-URSAT that has 16 baseline key-bits and connecting its output to an inverted output in the original circuit.

Connecting the second input of each MUX in S-URSAT to a unique input is not chosen randomly. To be more specific, we connect the second input of each inserted MUX in each stage to three different locations; (1) random internal net, (2) unique primary output, and (3) unique primary input. We realize that connecting them to unique primary inputs offers more resilience than connecting them to random internal nets or unique primary outputs. Practically, when the second input of each MUX is connected to either a random net or a unique primary output, AppSAT attack successfully decrypts a portion of the encrypted circuits that have a small number of baseline keys ($<$ 10 bits).

Algorithm 2 Strong Untraceable RSAT Algorithm

```
1:  Input: Netlist, keySize;
2:  Output: Encrypted netlist with key;
3:  Randomly select primary inputs equal to the keySize;
4:  for i ← 1 to KeySize do
5:  |   Insert 2-input XOR (input[i], key[i]);
6:  end for
7:  for i ← 1 to NumberOfCreatedOutputWire do
8:  |   if Number of created output wires >4 then
9:  |   |   Create 4-input AND gate;
10: |   |   Insert MUX(unique input, created gate output);
11: |   |   if One created output wire left then
12: |   |   |   Connect that wire to the previous AND gate;
13: |   |   end if
14: |   end if
15: end for
16: Get the last AND/MUX output wire(s);
17: for i ← 1 to NumberOfDips do
18: |   if Number Of Dips or createdOutputWires >1 then
19: |   |   if Number of Dips or createdOutputWires >4 then
20: |   |   |   Repeat 9-13 with OR gates, instead of AND;
21: |   |   else
22: |   |   |   Create OR gate with the reset incoming wires;
23: |   |   |   Insert MUX(unique input, created gate output);
24: |   |   end if
25: |   end if
26: end for
27: Get the last OR/MUX output wire;
28: Connect the OR/MUX wire with the AND/MUX wire(s) to AND gate;
29: Insert MUX(unique input, created gate output);
30: Connect the S-O/P signal with the flipped primary output to XOR and XNOR gates;
31: Insert MUX(XOR output, XNOR output);
```

Also, the values of the selected random nets or the selected primary outputs from the original circuit are not necessary changed when certain input patterns change. This may help removal attack identify/detect the S-URSAT block. On the other hand, AppSAT attack fails to decrypt any encrypted circuit when the second MUX inputs are connected to unique primary inputs. This will also make sure that the internal and output signals of the S-URSAT will change when the input

patterns change since they will directly feed from the primary input patterns when wrong keys are applied. Furthermore, both traditional SAT and D-DIP attacks need at least 2^n + m number of iterations to decrypt a circuit encrypted using S-URSAT, where m changes based on the size of the baseline key-bits and the topology of the original circuit. More information regarding the security analysis of S-URSAT, supported by experimental results, is provided in subsection 4.2.2.

4.1.4 Security Analysis of URSAT and S-URSAT

To validate the security of URSAT, we implement a small URSAT circuit, as shown in Figure 4.4. For simplicity, (1) we configure the I/B and the AND/OR PLGs to be a buffer and an OR when k is ’0’, respectively, and (2) we connect the R-S to ’0’ so the AND/NAND and the XOR/XNOR PLGs function as AND and XOR gates, respectively. As a result, the correct key will be "100". Table 4.1 shows the truth table for URSAT block. For each input pattern with key combinations, the URSAT produces 4 wrong outputs, except when all input patterns are ones, where the NAND gate will cancel all other incoming inputs to AND/NAND PLG. If we add one more AND/OR PLG with two I/B PLGs, the URSAT will cause 16 wrong outputs. In general, the number of produced wrong outputs can be formulated as in Eq. 4.1. The number of wrong outputs that URSAT causes makes AppSAT terminate with a large number of error bits. This also emphasizes that the URSAT works as TOA *only when the correct key is applied* (as also shown in Table 4.1). Note that we used the NAND gate with all primary input X for simplicity purposes. A designer can choose different logic gates and can also integrate them with random signals coming from the encrypted circuit using MUXes.

$$\#\,\overline{Y} = 2^{(2\,*\,\#AND/OR\ PLGs)} \qquad \forall\vec{x}(\vec{x} \in X, \vec{x} \neq 1) \tag{4.1}$$

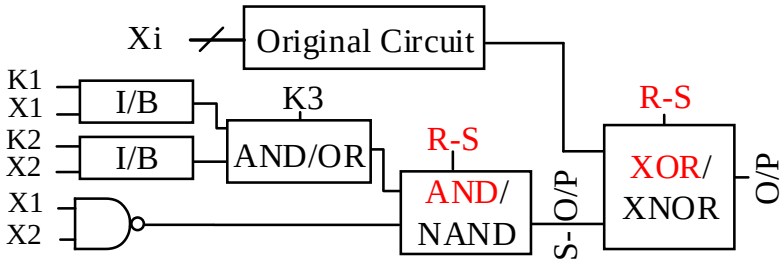


Figure 4.4: A simple example of implementing URSAT

Table 4.1: Truth table for URSAT in Figure 4.4

key / x1x2	000	001	010	011	100	101	110	111
00	0	1	1	1	0	0	0	1
01	1	0	1	1	0	0	1	0
10	1	1	0	1	0	1	0	0
11	0	0	0	0	0	0	0	0

For more secure design, instead of NANDing all of Xi signals together, each 2 Dips can be connected to OR gate, and then the OR outputs with random signals could be connected to MUXes to create a tree of OR gate. Regrettably, the URSAT depends only on the key-bits of the AND/OR PLGs to change the values of the internal signals and produce high corruptibility. A large number of the incorrect recovered key-bits by AppSAT is from the early stages in URSAT. In other words, AppSAT attack may quickly exclude the incorrect key-bits at the AND/OR PLGs that are close to AND/NAND PLG since they produce a very high corruption. This might help removal attack identify URSAT block. Therefore. the URSAT may not be fully resilient against AGR attack. In S-URSAT, unique inputs with MUXes are manipulated to change the internal signals in each stage. Since the input patterns should change during the test, the values of internal signals will be random. Also, the values of the unique inputs may not produce wrong outputs, e.g. when the value of the unique input has few or large zeros. Therefore, the AppSAT attack will terminate with a key

that has a large number of wrong key-bits at the MUXes. This error will increase when the unique inputs and MUX key-bits are increased, as will be experimentally shown in the next section.

4.2 Experimental Results

In this section, we present in details the experimental evaluation to demonstrate the effectiveness of the proposed logic obfuscation technique with URSAT and S-URSAT.

4.2.1 Experimental Setup

Our experiments use combinational (ISCAS'85) [161], sequential (ISCAS'89) [166], and ITC'99 [167] benchmark circuits. We summarize the details of the benchmarks used in our experiment in Table 4.2, where the last column shows the area size of the original benchmarks using 20*nm* CMOS. Synopsys HSPICE is adopted for circuit-level simulation to characterize SiNW based PLGs. We use Java to achieve our proposed logic obfuscation. To show the advantages of SiNW- over CMOS-based logic encryption, we randomly exchange 5% of the total number of gates in the original circuit (C1355-S9234) with SiNW-based PLGs and compare the penalties with randomly inserting 5% CMOS-XOR/XNOR logic gates (see Figure 4.9). Then, we evaluate the penalties of randomly exchanging 100 gates with SiNW-based PLGs for different large benchmark circuits (C5315-B22), each augmented with URSAT, and the penalties of S-URSAT (see Figure 4.10 and 4.11). URSAT and S-URSAT are implemented using only SiNW and CMOS-logic gates, respectively. We evaluate the URSAT and the S-URSAT techniques against SAT, D-DIP, AppSAT, and removal attacks based on the tools utilized [74, 94, 95]. Xilinx ISE Design Suite-14.7 tool is used to observe and track the URSAT/S-URSAT output signals with some other signals in the proposed design (see Figure 4.6 and 4.7). The CPU time is set to 10 hours as in [74]. SAT, D-DIP, and App-

SAT attacks are executed on a server that has Core i7 CPU at frequency 3.5GHz with 32GB RAM. The ABC tools [168] are used to convert the bench circuits to verilog for evaluation purposes. All converted verilog circuits are synthesized using Synopsys Design Compiler with 20*nm* CMOS and 20*nm* SiNW technologies. We set up the timing constraints as follows: (1) input delay = output delay = 0.2 ns, (2) load capacitor $(C_{\mathrm{L}}) = 10\,\mathrm{pF}$, and (3) clock period = 10 ns 50% duty cycle. The area, power, and delay overheads are retrieved accordingly.

Table 4.2: The details of the benchmark circuits used in this work

Benchmarks	Inputs	Outputs	Logic gates	Inv and Buf	Area
C5315	178	123	1413	894	357.62
C7552	207	108	2102	1410	395.74
S5378	214	213	1004	1775	295.23
S9234	247	250	2027	3570	387.3
S13207	700	790	2573	5406	622.15
S15850	611	684	3448	6327	817.74
S35932	1763	2048	12204	4181	2165.3
S38417	1664	1742	8709	13548	2323.59
S38584	1464	1730	11448	7957	2990.22
B14	277	299	4917	430	1335.68
B15	485	519	6540	482	2026.17
B17	1451	1511	21129	1628	6622.58
B18	3307	3293	64603	5310	19354.25
B20	522	512	11051	906	3097.2
B21	522	512	11154	980	3104.5
B22	735	725	15957	1372	4382.1

4.2.2 Security Evaluation for the Newly Proposed Technique

We tested our PLGs based traditional logic obfuscation. Unfortunately, SAT attack took less than two seconds with only a few iterations to break the C5315 circuit. Therefore, each of URSAT and S-URSAT is integrated with the original and obfuscated circuits. In this subsection, we analyze

and evaluate the proposed designs against the most serious reverse engineering threats.

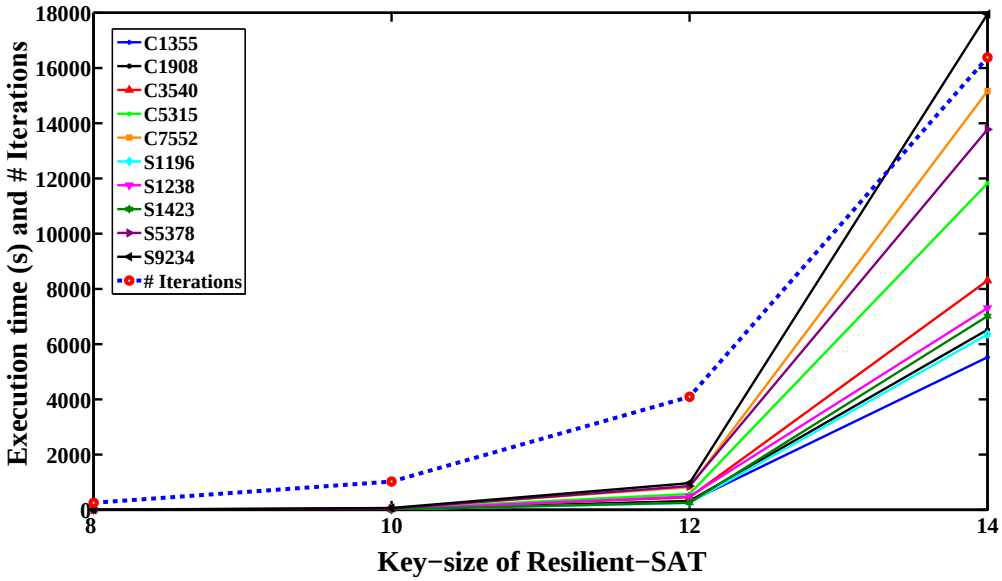


Figure 4.5: Execution times and number of iterations that SAT attack needs to decrypt different circuits + URSAT.

4.2.2.1 SAT, D-DIP, AppSAT, and Bypass attack resilience

We integrate each of the benchmark circuits with URSAT that has different key-sizes (8, 10, 12, and 14 bits). These encrypted circuits are evaluated against SAT attack, and both the number of iterations and the execution time that SAT attack needs are reported. Figure 4.5 shows the number of iterations that the SAT attack needs to break the circuits encrypted using URSAT, where this number is represented by the blue dot line in the Figure. There is only one blue dot line in the Figure, which implies that SAT attack needs exactly the same number of iterations (2^n) to break any encrypted circuit. With a small number of key-size, e.g. < 10 bits, all of the encrypted circuits are broken with a small number of iterations. However, this number grows exponentially when the key-size of the URSAT is increased linearly. Note that the aforementioned key-size number is the

baseline, which is only the number of key-bits that are directly connected with the primary inputs to I/B PLGs. The URSAT has a few more key-bits that are generated from the AND/OR PLGs. The required time for SAT attack to break these encrypted circuits is also reported in Figure 4.5.

We implement AND/OR PLG, instead of AND/NAND PLG at the last stage in URSAT block, and the iteration number of SAT attack reduces to approximately $2^n/2$. We investigate the reason and realize that SAT attack rules out more than one key at each iteration. This is the main reason behind leveraging AND/NAND PLG at the last stage in URSAT block and therefore increases the iteration number of SAT attack exponentially with the number of URSAT key-bits. We also test URSAT against D-DIP based attack, and we realize that such attack needs exactly the same number of iterations that SAT attack needs with about the same execution time.

The C1355 circuit is encrypted (by exchanging 27 different gates in the C1355 with PLGs) and connected with URSAT block that has only 16 key-bits as a baseline. SAT and D-DIP attacks timed out without breaking it within 10 hours. We rerun the experiment for the statistical significance, and both attacks still fail to decrypt it.

Moreover, we connect each of the 16 different circuits with S-URSAT that has 4 different key-sizes, so the total number of the encrypted circuits is 64. These circuits are evaluated against both SAT and D-DIP attacks. The first 8 columns in Table 4.3 illustrate the number of iterations that both attacks need to break the encrypted circuits. Even though the two attacks require different number of iterations to decrypt a similar encrypted circuit, the minimum required number of iterations for each attack is larger than the exponential number of the baseline key-bits by m (# Iterations = $2^n + m$), where m increases based on the key size and the selected inputs from the original circuit. It is important to mention that integrating a traditional encrypted circuit with S-URSAT will barely increase the number of SAT iterations over the produced iterations' number with S-URSAT block alone.

Table 4.3: Number of required iterations for SAT, D-DIP, and AppSAT attacks to break different circuits incorporated with S-URSAT, which has different baseline key-bits, and number of produced error bits for AppSAT. -B and -T in the Table are stand for baseline keys and total keys, respectively.

circuits	#Iterations for SAT				#Iterations for D-DIP				#Iterations for AppSAT and #Error bits							
	8-B	9-B	10-B	11-B	8-B	9-B	10-B	11-B	8-B	Err	9-B	Err	10-B	Err	11-B	Err
	15-T	16-T	19-T	20-T	15-T	16-T	19-T	20-T	15-T		16-T		19-T		20-T	
C5315	274	533	1059	2080	277	543	1037	2080	84	7	96	5	156	4	108	9
C7552	280	520	1070	2088	271	583	1056	2085	96	8	180	8	60	12	72	8
S5378	284	550	1046	2083	276	543	1053	2101	84	7	84	7	60	8	72	10
S9234	292	576	1268	2597	333	788	1270	2879	96	5	84	7	144	9	84	10
S13207	309	570	1095	2226	301	589	1103	2122	132	6	132	5	84	10	60	11
S15850	281	530	1065	2139	289	528	1096	2219	96	5	144	6	60	7	60	9
S35932	286	524	1044	2178	273	542	1106	2106	132	7	120	6	72	12	96	10
S38417	283	528	1058	2092	298	642	1128	2554	156	5	96	7	72	7	144	11
S38584	279	565	1117	2225	259	550	1095	2216	120	7	60	9	120	10	120	9
B14	267	534	1083	2174	275	655	1129	2211	60	7	144	5	60	9	96	7
B15	275	542	1055	2107	304	537	1042	2109	108	4	60	7	108	10	60	10
B17	258	513	1036	2067	263	526	1046	2107	156	5	60	7	96	11	60	10
B18	292	596	1171	2196	291	559	1143	2260	72	5	108	4	108	8	60	11
B20	300	533	1080	2137	310	609	1081	2209	108	6	228	8	72	10	84	10
B21	283	522	1054	2203	309	584	1192	2619	60	5	144	8	60	9	60	12
B22	262	531	1057	2079	359	577	1157	2304	108	6	72	7	132	8	60	9

Table 4.4: Results of AppSAT attack against circuits integrated with URSAT having different baseline key-bits.

circuits	Number of AppSAT iterations and number of produced error bits													
	8-B	Error	9-B	Error	10-B	Error	11-B	Error	12-B	Error	13-B	Error	14-B	Error
	14-T		15-T		17-T		18-T		21-T		22-T		24-T	
C1355	96	6	180	4	108	6	168	5	84	6	132	8	107	8
C5315	103	0	132	5	84	6	156	9	60	8	60	9	100	7
S5378	104	0	132	4	72	9	132	7	132	10	84	4	103	6
S9234	102	0	132	5	60	3	72	7	84	5	72	7	98	9

We also test URSAT and S-URSAT against AppSAT attack. The AppSAT terminates with a high error rate of the recovered key-bits. Since the AppSAT terminates the SAT attack when there are no errors for a certain number of *input patterns*, the URSAT key-bits (including the key-bits of the AND/OR PLGs) and the S-URSAT key-bits (including the key-bits of the MUXes) will

not be recovered correctly, resulting in high error rate. We implement AppSAT with the setup parameters as reported in [95] (settlement threshold – chosen to be 5, the number of inputs sampled – chosen to be 50, and the number of SAT attack iterations – chosen to be 12). The number of AppSAT iterations and the generated error bits in the recovered key, for different circuits integrated with S-URSAT that has different baseline key-sizes, are shown in the last 8 columns in Table 4.3. The AppSAT attack fails to decrypt any encrypted circuit even with a very small number of the baseline key-bits. While Table 4.4 demonstrates the number of AppSAT iterations and the generated error bits, for different circuits integrated with URSAT. The AppSAT attack decrypts most of the encrypted circuits that have only 8 baseline key-bits and fails to decrypt any other encrypted circuits that have larger baseline key-sizes.

Note that the Bypass attack cannot break S-URSAT technique since it produces wrong outputs for a large number of input patterns. When a wrong key is applied, the S-URSAT will be controlled by the unique primary inputs and therefore the Bypass attack should record all of the input patterns that cause wrong outputs. For a large baseline key-size, e.g. 64 bits, the unique primary inputs will be very large, which is infeasible for Bypass attack to record the patterns of such large inputs. The S-URSAT will give the correct I/O pairs only when the correct key is provided. Therefore, the key inputs of the MUXes should be correctly configured and should not be any random keys. The number of the MUX keys can be increased largely by reducing the input-size of the AND and OR gates in S-URSAT to 2, except for the last gate which may have 3 inputs. Also, the URSAT can mitigate Bypass attack since it will produce incorrect outputs at certain input patterns with wrong configurations at the AND/OR PLG key-bits. Therefore, the Bypass attack should record a large number of input patterns, which is impracticable.

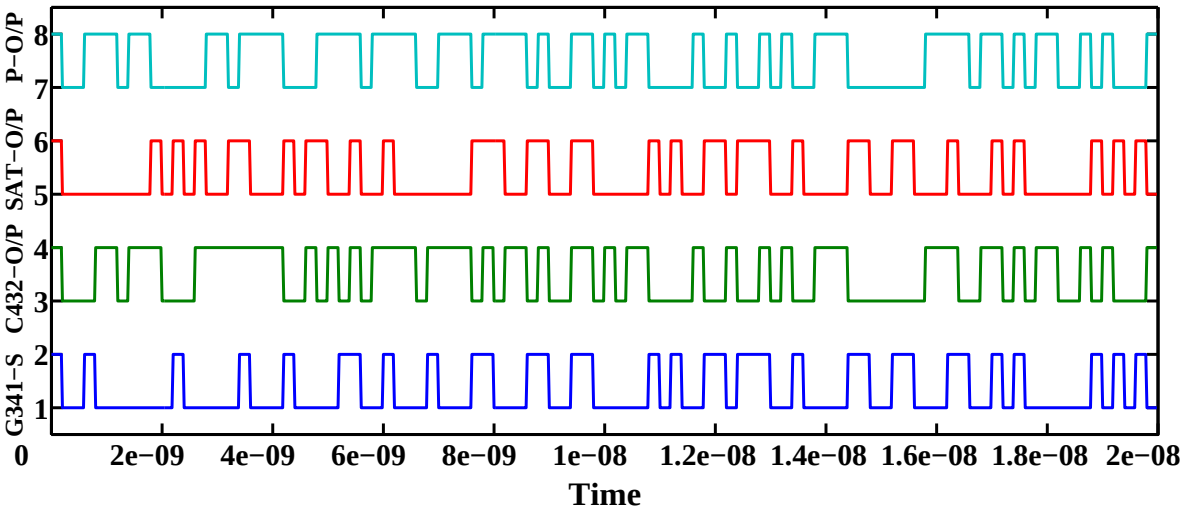


Figure 4.6: Preventing TS attack from tracking and detecting URSAT output.

4.2.2.2 Removal Attack Resilience

4.2.2.2.1 SPS and TS Attack Resilience

As mentioned, the SPS/TS attacks are used to identify and remove the TOA block via tracking the input signals and the output signal at the last gate in TOA block. Once the TOA is removed, the encrypted circuit becomes vulnerable to SAT attacks. These attacks can be prevented by implementing URSAT and integrating it with a netlist. Then, one of the two proposed R-S approaches should be used, e.g. connecting the R-S to a random internal signal coming from the original circuit. The output of the proposed technique, URSAT circuit+R-S, will be changeable but may act almost the same as the selected random signal if an attacker applies a constant key value.

More practically, we incorporate URSAT, which has 32 baseline key-bits, into the combinational benchmark circuit, C432. Then, random input patterns with random wrong keys are applied to observe the URSAT output signal with some other signals, and a sample of the results is shown in Figure 4.6. G341-S is the internal net signal (G341) coming from the original circuit C432.

C432-O/P is one of the C432 primary output signals. SAT-O/P is the output signal of the URSAT block. P-O/P is the output signal of the inserted XOR/XNOR PLG whose two inputs are C432-O/P and SAT-O/P. Figure 4.6 shows that the output of the URSAT block is swinging from zero/one for most of the key combinations. This output signal changes based on the values of the incoming signals to the last PLG in URSAT and the value of G341-S. The G341-S is used to configure the last AND/NAND PLG in the URSAT block as an AND gate and the inserted XOR/XNOR PLG as an XOR gate once its value is zero. Otherwise, the two PLGs will function as a NAND and an XNOR gates, respectively, when the G341-S value is one. As a result, the output of the URSAT block cannot be tracked by SPS/TS attacks. In addition, the values of the incoming signals to the last AND/NAND PLG in URSAT change based on the values of the AND/OR PLG key-bits. If an attacker feeds the key-bits of the AND/OR PLGs with constant values, the output of the URSAT will still change but will act almost the same as the selected internal signal, as we previously reported in [87] Figure 4. In this case, the maximum skew value of each incoming signal to the last AND/NAND PLG will be less than 256 since each one of them comes from 8 baseline key-bits. This skew value is obviously small and cannot be detected by removal attack since the minimum skew value of a signal should be larger than 1024 to be successfully identified by such attack [93]. Also, each incoming signal to the last PLG will be skewed to zero or one if the AND/OR PLGs on each incoming signal path are configured to AND or OR gates, respectively. Therefore, with this small skew value (256) and the unknown relationship between the incoming signals, SPS attack will fail to detect the URSAT block. Note that increasing the baseline key-bits of URSAT will not increase the skew value of any incoming signal to the last PLG. For instance, to enlarge the baseline key-size by 8 bits more, another signal, coming from a new sub-tree of PLG that has 8 baseline key-bits, will be fed to the last AND/NAND PLG. Also note that applying a constant key to the AND/OR PLGs is not easy to be achieved, since the original circuit should be first encrypted by the traditional obfuscation technique and then integrated with URSAT block. Thus, the keys of the AND/OR PLGs in URSAT will be mixed with the keys of the encrypted circuit, and this will

make identifying the keys of AND/OR PLGs harder.

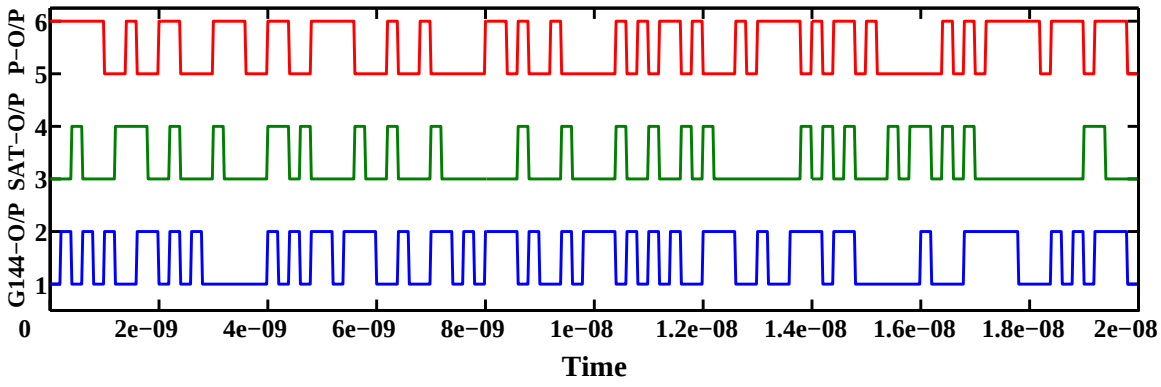


Figure 4.7: Preventing TS attack from tracking and detecting S-URSAT output.

The second R-S approach, which is adding an extra gate, e.g. an XOR gate, might be used to get better security since the output of the R-S (the output of the added XOR gate) will be different from the two selected internal signals from the benchmark circuit (the two inputs of the added XOR gate). Note that a designer can absolutely choose other different gates, e.g. AND, OR etc., rather than the XOR gate.

Traditionally, S-URSAT offers more resilience against SPS and TS attacks than URSAT since its internal and output signals always change when a wrong key is provided. In other words, when random input patterns with incorrect keys are applied, the values of the applied input patterns will convey to the internal signals in S-URSAT and its output signal. Therefore, the values of S-URSAT signals will always change when the input patterns are changed.

Practically, S-URSAT is integrated with the original benchmark circuit, C5315. We set the baseline key-size of the S-URSAT to 64 bits, so the total number of the generated key-bits will be 107 bits. We also use the same constraints as in URSAT (applying random input patterns with random keys). The S-URSAT output signal (SAT-O/P) and two other signals (P-O/P and G144-O/P, which is one

of the C5315 primary outputs) are observed. A sample of the results is shown in Figure 4.7, where these three signals are totally independent. The values of these signals change based on the values of the incoming signals from the previous stages in S-URSAT, the supplied input patterns, and the key-bits of S-URSAT. Therefore, the S-URSAT output signal is hard to be either tracked or detected. Moreover, this technique produces higher corruptions at the primary output than the URSAT when incorrect keys are applied, as evidenced by the P-O/P and the original output signals in Figures 4.6 and 4.7. Note that this implementation does not reduce the resilience of S-URSAT against SAT attacks, as shown in the previous subsection.

4.2.2.2.2 AGR Attack Resilience

The main idea of AGR attack is to recover the correct key-bits that product high corruptibility by AppSAT and then applies the SPS or TS attack to remove the TOA block. For example, in Anti-SAT technique, the AppSAT attack is first applied to get the correct key-bits of the traditional encrypted circuit and the contributed key-bits that are used to obfuscate G1 and G2 blocks. Then, SPS attack is applied to detect and remove Anti-SAT since it will work as one-point function after the AppSAT recovers a part of the total key-bits.

In our work, the first step of AGR attack is to recover the correct key-bits of the AND/OR PLGs in URSAT and the key-bits of the MUXes in S-URSAT by AppSAT attack. Then, The inconstant S-O/P signal in URSAT should be detected and removed by removal attack. However, the S-O/P signal is not easy to be detected since it is controlled by the R-S, which is always changed during the normal operation. The URSAT and S-URSAT will work as a one-point function (TOA) *if and only if* the key-bits of the AND/OR PLGs and the key-bits of the MUXes are successfully recovered. We observe the key-bits of the AND/OR PLGs in URSAT, and we unfortunately realize that the AppSAT attack can successfully recover a large number of the AND/OR PLG key-bits.

Table 4.5: Number of error recovered key-bits by AppSAT attack over the total number of AND/OR PLG key-bits in URSAT and MUX key-bits in S-URSAT.

Tech \ baseline	8-B	9-B	10-B	11-B	12-B	13-B	14-B
URSAT	0/6	0/6	0/7	1/7	2/9	3/9	1/10
S-URSAT	2/7	2/7	5/9	4/9	4/9	4/9	6/11

The AND/OR PLG key-bits are not connected with other control signals coming from the original circuit. As a result, increasing the AND/OR PLG key-bits will not guarantee an increase in the number of error bits in the recovered AND/OR key-bits by AppSAT attack. AppSAT attack can recover more correct key-bits of the AND/OR PLGs if its setup parameters are slightly increased. If AppSAT attack successfully recovers the AND/OR PLG key-bits, the removal attack will be able to identify URSAT *if and only if* he can find the inconstant S-O/P signal that acts the same as the selected random signal, which is hard to be achieved as we explained in section 4.1.2, paragraph 4. Therefore, URSAT may not be fully secure against AGR attack. In S-URSAT, the AppSAT attack fails to recover a large number of the MUX key-bits, and therefore this technique can successfully thwart AGR attack. The manipulated unique primary inputs with the key-bits of the MUXes make AppSAT attacks recover the key-bits at the MUXes with a larger number of error bits compared to the AND/OR PLG error bits in URSAT. As a consequence, the number of the error bits will increase as soon as the number of the MUX key-bits is increased. Experimentally, we integrated C1355 circuit with URSAT and S-URSAT, where each one of them has different baseline key-sizes, ranging from 8 to 14 bits. Table 4.5 illustrates the number of incorrect recovered key-bits by AppSAT over the total number of key-bits that are generated from either AND/OR PLGs in URSAT or MUXes in S-URSAT.

4.2.2.3 Sensitization Attack Resilience

Sensitization attack can be prevented by making interference graph among the inserted key-gates, and therefore muting a key-bit will not lead to propagate the value of the second key-bit to an output [2]. In our proposal, an attacker cannot sensitize the key values of the locked design since all of the key-bits (baseline + shared key-bits) in URSAT and S-UIRSAT converge to one wire. To be more specific, if the attacker wants to propagate a key-bit to an output, the rest of key-bit values should be known since all the keys are connected to the same path. Since these key values are not known, the complexity of such attack will not be better than the brute force.

4.2.2.4 Physical attack resilience

The only possible attacks during the chip fabrication in an untrusted company are the analytical reverse-engineering attacks, but, after the fabrication, the encrypted circuit could possibly be threatened by physical attacks. Below are the possible physical attacks on the encrypted circuit and the defense strategies.

4.2.2.4.1 Read-out sensitive data from key-gates

An attacker can use a scanning electron microscope in Passive Voltage Contrast (PVC) mode to read-out data stored in non-volatile memory cells, and, based on the electrons accumulated in the floating gate, the attacker can identify the value of the stored data [115]. Such attacks will not be able to learn the key of the encrypted circuit from the key-gates, e.g., PLGs, MUX, XOR/XNOR gates, since the key-gates that are implemented using CMOS- and SiNW-transistors are volatile. Once the chip is off, the data in the key-gates will be gone, and therefore there is no way for the attacker to know the key. It is worth mentioning that this attack could be possible on spintronic device-based PLGs since spintronic devices are non-volatile. However, it seems to be very hard

for this attack to read-out the data from these devices since they are much smaller than CMOS (typically spintronic devices are 20-50X smaller than CMOS [160]).

4.2.2.4.2 Identifying the PLGs and removing the URSAT/S-URSAT blocks

An attacker can use advanced reverse engineering equipment and tools to recover the layout of the obfuscated circuit (PLGs + URSAT/S-URSAT) by depackaging, delayering, and imaging the obfuscated circuit. Since SiNW-based PLGs and all other existing PLGs offer different functionalities with the same structure, the attacker cannot identify the original functionality of the PLG. For example, in Figure 3.14, same structure can function as either a NAND or NOR gate based on the key configuration. Moreover, by recovering the layout of the obfuscated circuit, the attacker may distinguish the URSAT/S-URSAT location and remove it. A designer can easily prevent such attack by (1) connecting the URSAT/S-URSAT output signal (S-O/P) to many randomly selected outputs in the encrypted circuit and flipping the gate at half of the selected outputs, and (2) increasing the entanglement between the URSAT/S-URSAT and the encrypted circuit through adding some MUXes. One input of each inserted MUX is a random internal wire coming for the encrypted circuit, and the second MUX input is one of the AND/OR PLG output signal in URSAT (or MUX output signal in S-URSAT) at each stage. This will also increase the corruptions in the encrypted circuit once incorrect key is given. In case of URSAT, at each flipped output, the reconfigurable signal (R-S) should configure the last AND/NAND PLG in URSAT and the inserted XOR/XNOR PLG to either NAND/AND PLG and XOR/XNOR PLG or to AND/NAND PLG and XNOR/XOR PLG, respectively. Figure 4.8 shows the details of preventing an attacker from removing URSAT/S-URSAT.

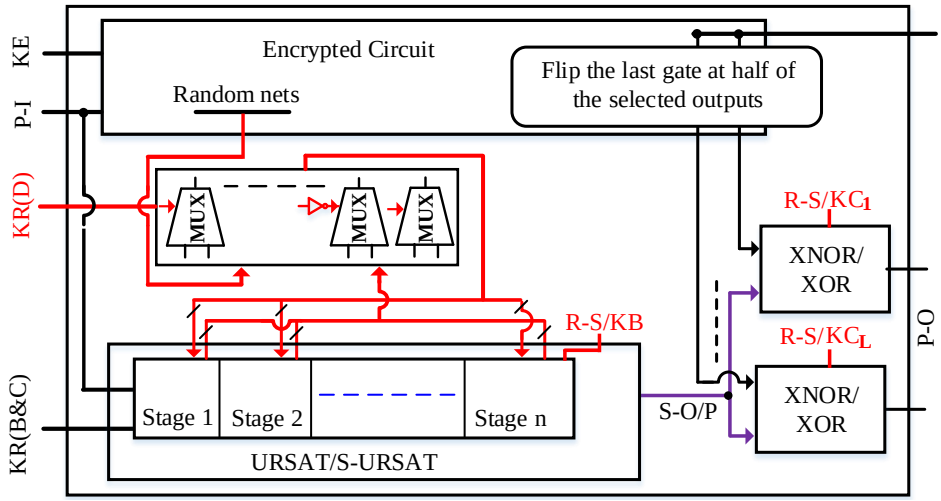


Figure 4.8: Preventing physical attack from removing URSAT/S-URSAT block.

4.2.2.4.3 Power analysis attacks

Side channel analysis attacks (SCAAs) are the most serious threats on the encrypted circuit since they can learn the secret key of the encrypted circuit when the chip is under the test or during the normal operation and without the need to access the memory that stores the secret key. SCAAs can reveal the secret key via observing the power dissipation [41], electromagnetic emissions, timing analysis [169], or photon density [170]. All these attacks are severe due to the same reason, which is *the natural behavior of the CMOS (and SiNW) devices during the operation*. For instance, the device consumes power from VDD only when the output changes from '0' to '1' or '1' to '0', while during the two other transitions (0 to 0 or 1 to 1), there is no power dissipation. The only way to effectively prevent such attacks, in CMOS- and SiNW-based PLGs, is to add extra hardware resources. For example, in CMOS, a resilient circuit, called Current Mode Logic (CML), can be used to produce approximately a constant power consumption at the device's output and here prevent SCAAs. Unfortunately, This method is expensive due to requiring large overhead. Interestingly, SiNW can be used to implement CML circuit by adding a few SiNW-transistors (1-

3 depending of the PLG type) [171]. In the new generation of spintronic devices (GSHE, DWM, STT, and MTJ), the read-from and write-to these devices are asymmetric, where the write current is much larger than the read current. Therefore, implementing PLGs using such devices will be much more vulnerable to side channel attacks than CMOS (or SiNW) [132]. Note that the read current from these devices cannot be increased since the stored data in these devices might be changed accidentally, while reducing the write current will not guarantee storing the new data correctly. The differences between the two-write and two-read ('0' or '1') currents make the device vulnerable to SCAAs. Based on our knowledge, only ASLD can prevent these attacks due to naturally generating identical current during the switching.

4.2.3 Area, Power, and Delay Penalties

Figure 4.9 (a) shows the area overhead of ISCAS'85 and ISCAS'89 benchmark circuits that are encrypted by exchanging 5% of the total number of gates in each original circuit with PLGs, and inserting 5% random XOR/XNOR key-gates. The number of the produced keys for each circuit is on the top of each bar. Even though the area overhead of the proposed approach is not very small, it is still much better than incorporating XOR/XNOR key-gates. The area overhead is somewhat large for the following reason: we exchange the higher percentages of AND/NAND/OR/NOR gates in the original circuits with their corresponding PLGs since the original circuits have higher percentages of those gates. Replacing such gate types requires higher performance overhead than other gates, e.g. XOR/XNOR gates. This is due to the fact that the XOR/XNOR SiNW PLG requires only four transistors while it consumes at least eight transistors in CMOS. The average area overhead of replacing with PLGs is less than inserting XOR/XNOR gates by about $4\times$. Figure 4.9 (b) shows the power overhead of combinational and sequential benchmark circuits for exchanging 5% and inserting 5% of the total number of gates in the original circuit. It is apparent that PLGs based logic obfuscation barely increases the power overhead. The delay overhead of exchang-

ing 5% PLGs is very small for most of the benchmark circuits compared to the random inserting XOR/XNOR gates, as illustrated in Figure 4.9 (c).

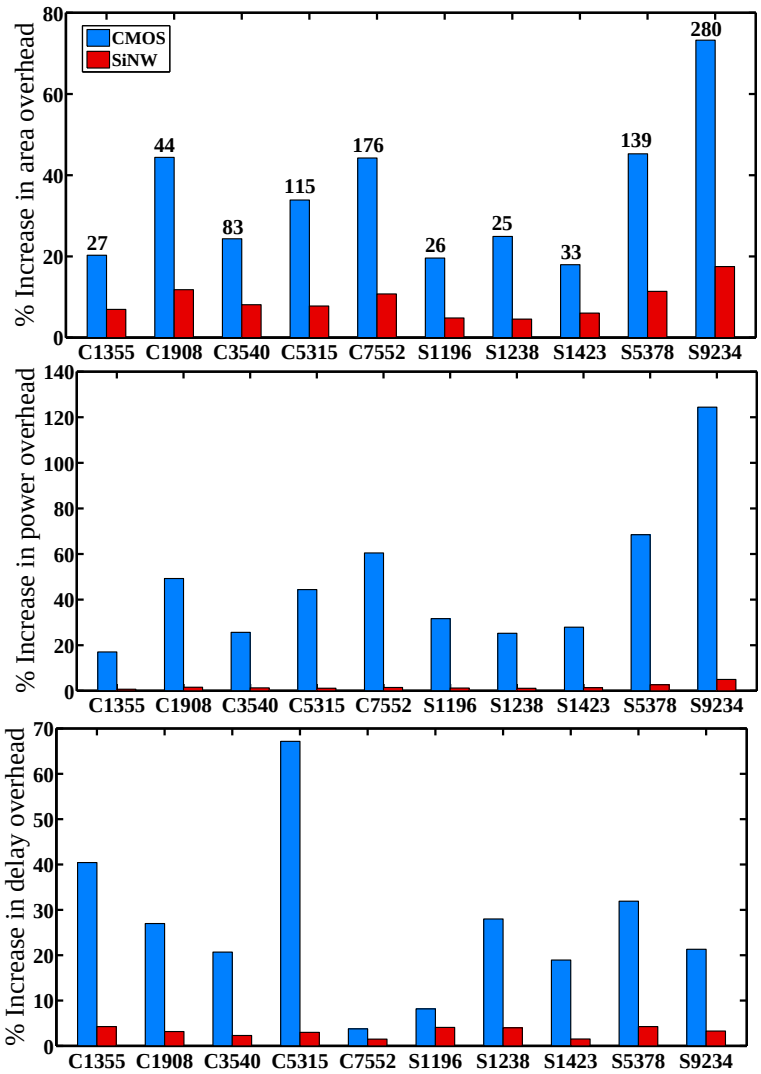


Figure 4.9: A comparison between the penalty of exchanging 5% of the total gate number with PLGs and randomly inserting 5% XOR/XNOR gates: (a) Area, (b) power, and (c) delay overheads.

It is worth mentioning that the selected benchmarks in Figure 4.9 have small size. Replacing 5% from the total number of gates in the original circuit is not good for large scale circuits since the produced key-bits will be very big and this requires a large memory size.

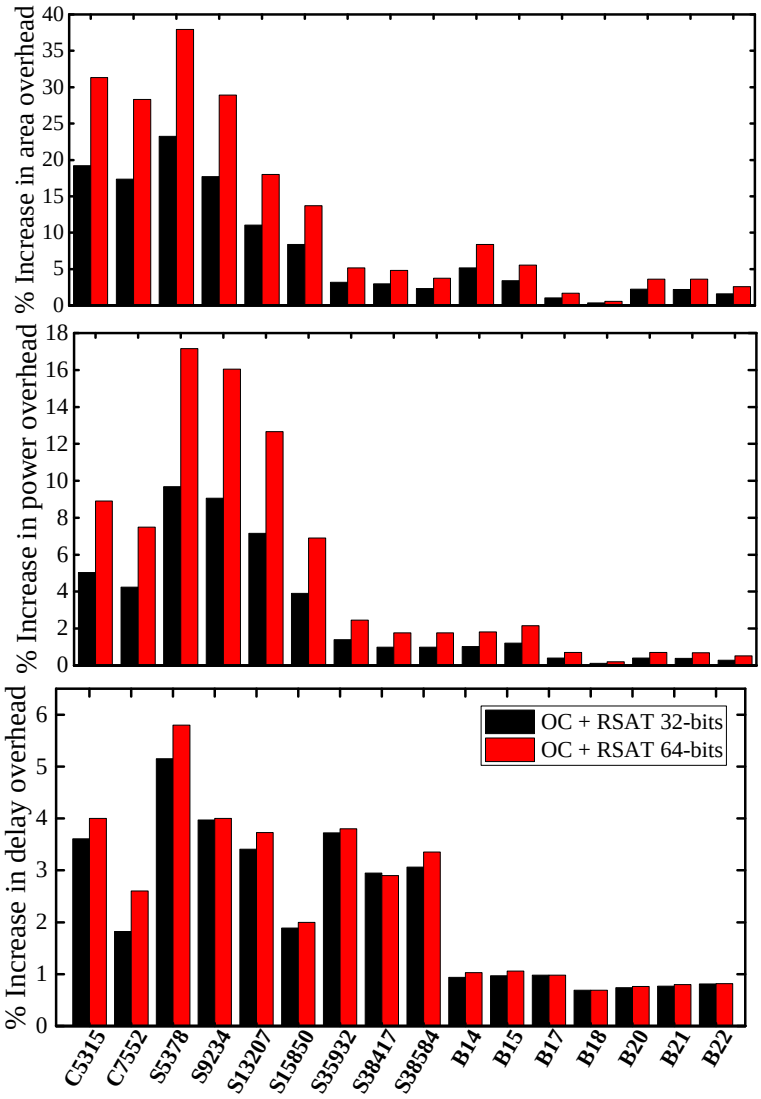


Figure 4.10: The penalty of exchanging 100 gates of the total gate number with PLGs + URSAT: (a) Area, (b) power, and (c) delay overheads.

Therefore, for larger circuits, we evaluate the total area, power, and delay overheads of exchanging 100 gates + URSAT block. Figure 4.10 shows the performance penalty of exchanging 100 gates with PLGs + URSAT block that has different baseline key-sizes (key-size = 32 and 64 bits). The average area, power, and delay overheads for encrypted benchmarks + 64 URSAT key-bits for the five mid-size benchmarks, S35932-B15, are 5.53%, 1.99%, and 2.43%, while for the five larger-

size benchmarks, B17-B22, are 2.41%, 0.57%, and 0.81%.

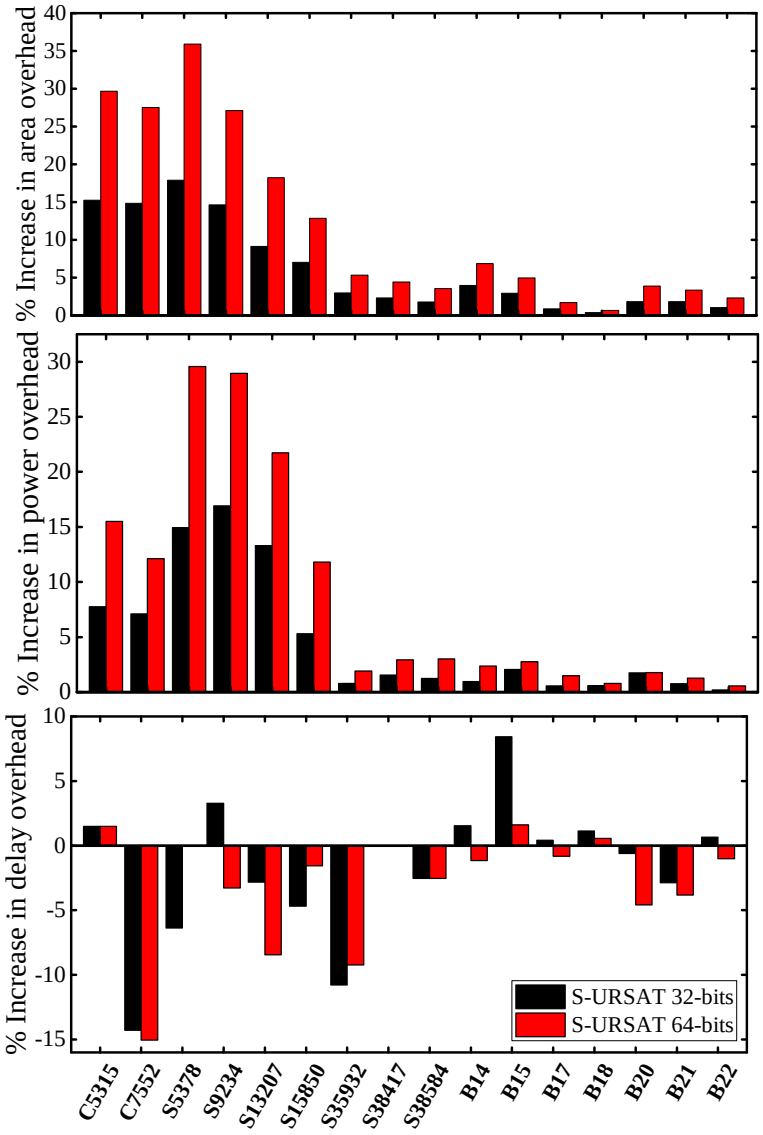


Figure 4.11: The penalty of integrating S-URSAT with different key-sizes to different circuits: (a) Area, (b) power, and (c) delay overheads.

Note that the total number of key-bits in this approach is the produced key-bits from the exchanging gates plus the URSAT key-bits. For instance, the total number of key-bits for encrypted S9234 + 32 baseline key-bits of URSAT is: 100 (from exchanging 100 different logic gates in the original

circuit with PLGs) + 32 (baseline key-bits of URSAT) + 28 (key-bits of AND/OR PLGs, where each of the 28 PLGs has 2 inputs, except the last one which has 5 inputs (R-S)) = 160 key-bits.

Moreover, a designer can get a better secure circuit with small overhead by implementing the S-URSAT technique using only CMOS-logic gates. We evaluate the area, power, and delay overheads of S-URSAT technique that has different baseline key-sizes (KA = 32 and 64). Figures 4.11 (a), (b), and (c) show the area, power, and delay penalties of S-URSAT, respectively. The average area, power, and delay overheads for 64 baseline key-bits of S-URSAT for the five mid-size benchmarks, S35932-B15, are 5.03%, 2.60%, and -2.26%, while for the five larger-size benchmark circuits, B17-B22, are 2.37%, 1.18%, and -1.93%. The Figure shows that S-URSAT offers delay improvements in most of the encrypted benchmark circuits. As known, adding extra hardware resources, i.e., logic gates, to a circuit will always increase the required area and power dissipation but not necessarily increase the delay since during the compilation the Design Compiler does a lot of optimizations on the design. Also, the input and output signals of the added design, S-URSAT, are directly connected with the input and output signals of the original circuit. During the compilation, this helps to improve the performance in many circuits since the delay is calculated as the longest path from the input to the output. In general, the delay depends on the topology of the circuit, the locations of the added design, and the type of the optimization algorithm that is used by the Compiler. For instance, even though B18 is much larger than C7552, the delay improvement in C7552 is much better than the one in B18. Moreover, the setup slack for all circuits is positive, which means there is no timing violation in any circuit. Finally, there is a relationship between the switching power and delay. When the delay reduces, the power should increase. This appears very clear in some of the encrypted circuits, i.e., B14, B17, and B22. In these circuits, when the delay improved largely by adding S-URSAT with 64 baseline key-bits compared to adding S-URSAT with 32 baseline key-bits, the power dissipation of adding S-URSAT with 64 key-bits increased by more than 2× (about 2.6× on average). Note that the total number of key-bits for this approach is

the baseline key-bits of the S-URSAT (KA) plus the key-bits of the MUXes (KB). As mentioned, the number of S-URSAT key-bits can be increased by reducing the input size of each inserted AND and OR gates to 2 inputs, except the last gate.

4.3 Discussions

4.3.1 URSAT and S-URSAT Vs Other Techniques

It is worth to elucidate why URSAT and S-URSAT are resilient against removal and AppSAT attacks compared to SARLock and Anti-SAT. SARLock and basic Anti-SAT are implemented to produce one-function output in order to maximize the number of SAT iterations ($2^n - 1$); however, they are vulnerable to removal and AppSAT attacks. The obfuscated Anti-SAT, which is achieved via adding 2 n additional key-gates (n XOR/XNOR and n MUX key-gates), gives some resilience against removal, Bypass, and AppSAT attacks. However, it is less resilience against SAT attack than basic Anti-SAT, e.g. SAT attack needs less iteration number and execution time to break obfuscated Anti-SAT. Also, it is vulnerable to AGR attack due to the following reasons: (1) the Anti-SAT output is directly connected with a primary output to an XOR gate (this will limit switching the Anti-SAT output signal since no input signal(s) with key-bits are used to switch the Anti-SAT output signal as in S-URSAT), (2) the n XOR/XNOR key-gates that are inserted at the inputs of G1 and G2 blocks could not produce high corruptibility at Anti-SAT output due to the natural behaviour of TOA, and (3) the n MUX key-gates that are inserted between the internal wires of the original circuit and Anti-SAT do not provide strong resilience. A large number of the MUX outputs are connected to internal signals in the original circuit, which will not affect/change any signal in Anti-SAT when a wrong key is applied. The rest of the MUX outputs are connected to internal signals in Anti-SAT, which can produce a high corruptibility. However, AGR attack can easily recover the key-bits of the MUXes.

The S-URSAT produces *a strong non-corruptibility output*, which is resilient against AppSAT and removal attacks. More precisely, when a wrong key is provided, the internal signals and the output signal of S-URSAT will be fed by unique primary inputs. Thus, these signals will always change and cannot be tracked, as explained in Figure 4.7. The S-URSAT also gives correct output patterns for a few number of input patterns with incorrect keys. As a consequence, when AppSAT attack finds a key that provides the correct output patterns for specific random input patterns, the attacker will terminate. This key cannot satisfy all of the input/output pairs because the S-URSAT does not work as a one-function output, and the recovered key will have a number of error bits. When the key-bits of S-URSAT increase, this error will increase largely even when the setup parameters of AppSAT are increased. More practically, we incorporated S-URSAT having 64 baseline key-bits (so totally 107 bits) to C5315 circuit. We increased the number of setup parameters in AppSAT attack to 10× (threshold – 50, inputs sampled number – 500, and the SAT iteration number – 120). The AppSAT attack terminated after about 16 hours with 6000 iterations and 47% error rate (50 incorrect recovered key-bits). In URSAT, when wrong keys are applied, the PLGs will be configured as OR gates and the internal signals in URSAT will change, while the output signal of URSAT will always change whether the key-bits of PLGs change or not since it is controlled by the R-S. Furthermore, when the PLGs configure to OR gates, the URSAT will also produce correct corresponding output patterns for certain input patterns, and the AppSAT attack will terminate with a key. This recovered key cannot satisfy all I/O pairs, resulting in producing error bits. However, increasing the setup parameters of AppSAT helps recovering more correct key-bits at AND/OR PLGs, and this benefits AGR attack *only* if the removal attack can successfully detect S-O/P signal, which is not easy to be achieved, as explained in subsections 4.1.2 and 4.2.2.2.

In order for AppSAT attack to get the correct key of a circuit encrypted using URSAT/S-URSAT, the setup parameters should be increased linearly when the key size of the encrypted circuit is increased. However, the execution time and the iteration number of AppSAT attack will also

increase exponentially but less than the required one by SAT attack. Practically, C1355 circuit was incorporated with URSAT and S-URSAT, each has different baseline key-sizes. Then, the setup parameters of AppSAT were linearly increased by the same number of the original setup (12 50 5) if the AppSAT attack fails to get the correct key. The required number of iterations and the execution time of AppSAT to get the correct key are shown in Figure 4.12, where x in the Figure represents the values of the setup parameters for AppSAT. The AppSAT attack requires less number of iterations compared to SAT and D-DIP attacks; however, the attack still needs exponential number to break either URSAT or S-URSAT. Also, the attack needs larger number of iterations and longer execution time to break S-URSAT than URSAT. This is because unique inputs from the original circuit are utilized in each S-URSAT stage, besides the MUX key-bits. To successfully recover the correct key from URSAT/S-URSAT, the values of the original setup parameters (x) should be first increased twice when the number of baseline key-bits of URSAT/S-URSAT is only 8 bits. Then, these values should be increased one time once the number of key-bits of URSAT is increased by 2 while they should be increased approximately twice when the key-bit number of S-URSAT is increased by 2, as illustrated in Figure 4.12.

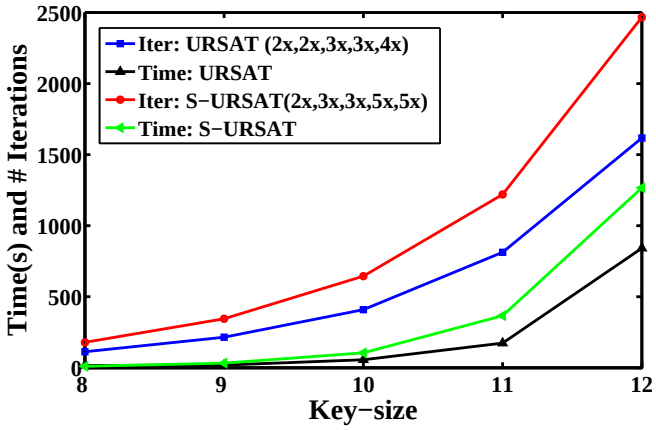


Figure 4.12: Execution times and number of iterations that AppSAT attack needs to decrypt C1355 circuit + URSAT/S-URSAT, where x refers to the base setup parameters of AppSAT.

Based on the above discussions, S-URSAT is more resilient than URSAT since SAT and D-DIP attacks need more number of iterations to break S-URSAT. Moreover, only the AND/OR PLG key-bits in URSAT control its internal signals, while both the unique primary inputs and the MUX key-bits at each stage in S-URSAT control the internal signals and the output signal of S-URSAT. As a result, AppSAT fails to break any circuit encrypted with S-URSAT, and removal attack cannot detect the S-URSAT signals since these signals are totally controlled by the unique inputs and the MUX key-bits. Table 4.6 summarizes existing logic locking techniques, where OH, L, M, S, V-S symbols refer to overhead, large, medium, small, and very small, respectively. To conclude, in order for a defense mechanism to successfully prevent reverse engineering attacks, it should satisfy the following conditions:

1 Producing high corruptibility on applying certain input patterns with incorrect keys to prevent AppSAT and Bypass attacks.

2 Making the internal and output signals of the technique changeable when incorrect keys are applied to prevent removal attacks.

3 Producing low corruptibility for a large number of input patterns to prevent both traditional SAT and D-DIP attacks.

4.3.2 Is the encrypted design using S-URSAT alone fully secure?

Like other existing techniques, e.g. TOA, SARLock, and Anti-SAT, the original netlist still needs to be encrypted using traditional logic encryption techniques, and then S-URSAT block should be incorporated to produce fully secure chip. Note that, even though S-URSAT provides higher corruptibility than other existing techniques, e.g. TOA, SARLock, and Anti-SAT, it is still not sufficient to produce fully secure chip, especially against physical attacks that may identify and remove S-URSAT. Since the traditional logic encryption techniques using CMOS logic-gates are

typically costly and slow down the performance speed, as clarified in Figure 4.9, a designer can encrypt the original circuit using PLG-based traditional Logic encryption and then integrating the S-URSAT block.

Table 4.6: A comparison between our designs and other techniques

attacks / Techs	Sen [2]	SAT [74]	D-DIP [94]	AppSAT [95]	ByPass [97]	SPS/TS [93]	AGR [93]	FALL [100]	CycSAT [106]	TimingSAT [108]	SMT [110]	OH
Random [68]	✓	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	M
FLL [69]	✓	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	M
SOL [2]	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	M
MUX-Locking [147]	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	M
SiNW-CMOS [88]	✓	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	V-S
One-point* [85]	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	S
TTlock* [98]	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	S
Anti-SAT* [3]	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	S
SFLL* [4]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	L
DLL [107]	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	M
Cyclic* [105]	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	M
RAND-PLG (RP)	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	V-S
URSAT+RP	✗	✗	✗	✗	✗	✗	P	✗	✗	✗	✗	S
S-URSAT+RP	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	S

✓: Successful attack, ✗: Unsuccessful attack, P: Only recover partial of the key, e.g., key-bit values of AND/OR PLGs in URSAT. Now, after recovering the key-values of the AND/OR PLG by AppSAT, the attacker needs to detect the S-O/P signal that will be acting the same as the selected random net, but it is hard to be achieved, as explained in section 4.1.2, paragraph 4. *: These technique should be integrated with the encrypted circuit, e.g. random, FLL, and SOL, to prevent other attacks, e.g. some types of physical attacks. Therefore, the overhead will be large.

4.4 Summary

We have demonstrated that the usage of emerging transistors, such as SiNW FETs, can help improve the logic locking design by preserving lower power consumption and smaller area than conventional CMOS counterpart. Specifically, the SiNW-based polymorphic logic gates (PLGs) function as logic key units to encrypt the combinational circuits. In addition to traditional criteria such as power, delay, and area, security may serve as a new criterion to examine the pros and

cons of emerging transistor and/or memory technologies. The proposed technique offers high level security against IP piracy, reverse engineering, and tracking signal based attacks via exchanging some logic gates in the original circuit with PLGs and incorporating URSAT. Since URSAT technique may not be fully secure against AGR attack, we implement S-URSAT using only CMOS-logic gates to strongly prevent such attack. The combined S-URSAT and PLG-based encryption increases the security level of the design to robustly thwart all existing attacks.

CHAPTER 5: SECURE AND RESILIENT HARDWARE DEVICES USING ASLD

To build a hardware design that is fully protected from IC attacks (during the fabrication in an untrusted foundry) and very resilient against PA attacks (after the fabrication process), a designer needs to add extremely large hardware resources, as follows: (1) inserting sufficient number of key-gates to produce encrypted design, (2) adding addition circuit that resists key information leakage to produce resilient design, and (3) incorporating non-volatile memory to store the secret key of the encrypted circuit. Figure 5.1 explains the required modification on the IC design flow to generate a secure and resilient chip [1][2].

5.1 Hardware Device after the Fabrication: Attacks and Defenses

In chapter 3 and 4, we explained in details how to prevent IC attacks. Now, after the fabrication, the secret key of a secure hardware device can be revealed through observing the power dissipation, electromagnetic emissions, or time analysis [44]. PA attack is more popular and powerful than others. The first PA attack was presented by Kocher et al. [45]. There are three types of PA attack; simple power analysis (the data are directly taken from the power dissipation), differential power analysis (the related information is obtained from large number power traces, and then statistical analysis is used to identify the differences in these traces), and correlation power analysis (CPA) (the real power is correlated with the predicted power model to get the leakage data). CPA is more advanced and faster than other [46].

In CMOS technology, the PA attacks can reveal the secret key of a secure hardware device, e.g. cryptography, via measuring and analyzing the power consumption of the data dependence (intermediate data at specific operations) during the voltage transitions. For instance, the device consumes power from VDD only when its output switches from ‘0’ to ‘1’, and from ‘1’ to ‘0’. While during the two other degenerated transitions (‘0’ to ‘0’ or ‘1’ to ‘1’), no power is required [67].

In the spintronic devices, the operations of write-to and read-from MTJ are asymmetric. For instance, if the previously stored data in the MTJ is ‘0’ and the new data is ‘1’, the write current will go from low to high. In contrast, when the previously stored data is ‘1’ and the new data is ‘0’, the write current will go from high to low. For the two other states (‘0’ to ‘0’ and ‘1’ to ‘1’), the current will be constant [132].

These differences between the write and read currents make the device vulnerable to PA attacks. The new generation of spintronic devices, Hybrid Spintronic-CMOS devices including Magnetic Tunnel Junction (MTJ), have been utilized to overcome Moore’s law limitation as well as preserve higher performance with a lower cost. This technique is achieved by adding a transistor on the top of each nanomagnet to reduce the power dissipation and speed up the device. However, the differential power at the output of the Hybrid Spintronic-CMOS device and asymmetric read/write operations in MTJ will increase. Therefore, this increases the device vulnerability to PA attacks since the power dissipation at the output of the device will be significantly different.

There are many PA attack countermeasures, but the most two popular ones are the masking [64] and hiding [65] techniques. The masking method targets to randomly mask the intermediate values, which requires additional hardware resources and significant overhead. For the hiding method, even though the performance penalty is not that much for different types of operations, there is a trade-off between security and power dissipation. Increasing noise power can boost the protection of the design. However, the signal to noise ratio will be reduced [66]. Employing a Sensing

Amplifier Based Logic (SABL) [67] is a more empirical technique to prevent PA attacks. Moreover, Current Mode Logic (CML) [172] can be employed to protect the circuit from PA attacks by producing approximately constant power dissipation at the output. Unfortunately, the above-mentioned techniques require tremendous power and area overhead for a high protection level.

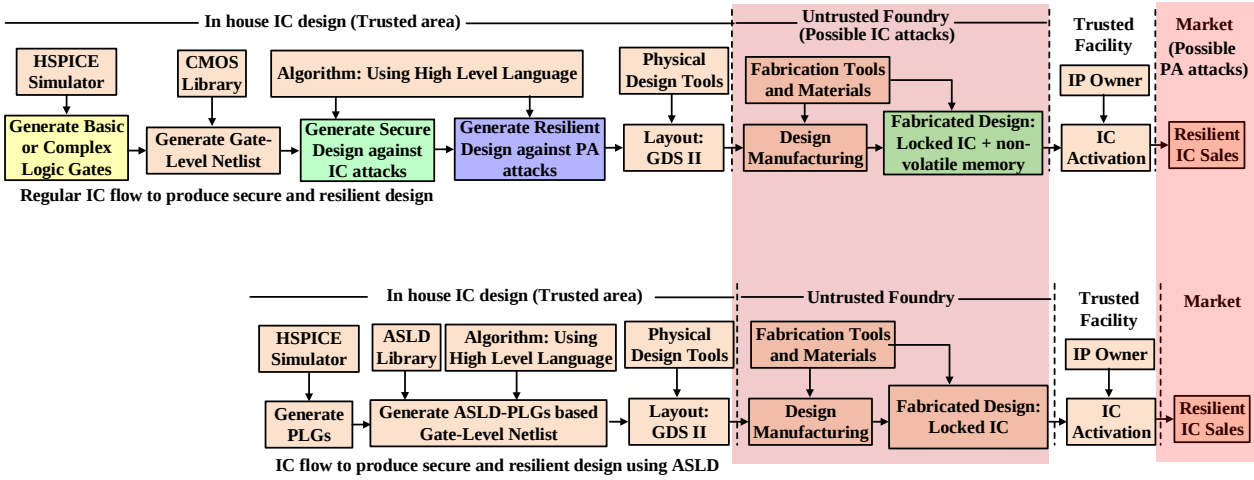


Figure 5.1: Main IC flow steps to generate secure and resilient design using CMOS and ASLD technologies: The yellow and blue boxes donate the locations of using SABL (or CML) and masking (or hiding) techniques, respectively, and the two green boxes describe the locations of using logic encryption method where each of these four places requires extra resources.

Typically, SABL or CML technique is applied at the time of generating the logic gates, and the masking or hiding technique is deployed at the gate level netlist. The yellow and blue regions at the top part of Figure 5.1 show where the extra resources are required for implementing SABL or CML and masking or hiding, respectively. While in our proposal, shown at the bottom part of Figure 5.1, the IC is protected from IC and PA attacks without adding any additional hardware resources.

5.2 Evaluations and Analyses of ASLD

5.2.1 ASLD as Logic Gates and Data Storage

5.2.1.1 Fundamental operation of ASLD

Emerging devices have been used in many applications to outperform CMOS technology in terms of power dissipation and performance. Several emerging devices have been fabricated over the past few years, such as FinFETs [117], carbon nanotube [118], and spin transfer torque [119]. The first proposal for spin-based logic in semiconductors was introduced by Dery et al. in [133]. The proposed design contains non-local spin signals, which require an amplification circuit to enlarge these signals and produce sufficient current to switch the nanomagnets. An improvement in this work [134], namely the All Spin Logic device, was achieved to switch a nanomagnet, which could be the input for the next stage, through the non-local spin signal. Thus, no extra hardware (amplification circuit) is needed. The ASLD model contains four main components, including the nanomagnetic (ferromagnetic) unit, interface element, isolator, and non-magnetic channel. The nanomagnet is used as a storing element. The interface element between the nanomagnetic unit and non-magnetic channel amplifies the injected spin current from the nanomagnet to the non-magnetic channel. The isolator separates the input and output ports.

Figure 5.2 shows the ASLD in details. Figure 5.2 (a) represents the basic ASLD. A basic ALSD comprises the nanomagnetic unit to store the binary data, an isolation layer between the input (with low spin polarization factor) and output (high spin polarization factor) ports, and one non-magnetic channel. Figure 5.2 (b) shows a simple ASLD with two magnets. These two magnets are polarized in the same direction and connected through a non-magnet channel. The channel is made from nickel or copper due to high spin-flip length. The maximum length of the channel

relies on spin-flip length, which is used to identify the maximum distance that the spin current travel (more details about the channel length is described in [173]). On applying negative VDD, the charge current will flow from GND to VDD, and the electrons will flow from VDD to GND. Electron Spins in the same direction of M1 will pass while others will be blocked by M1 (electrons get filtered). Since the output of M1 has high spin polarization and the output of M2 has low spin polarization, M1 will dominate the spin current, and the passed spins will accumulate in the channel.

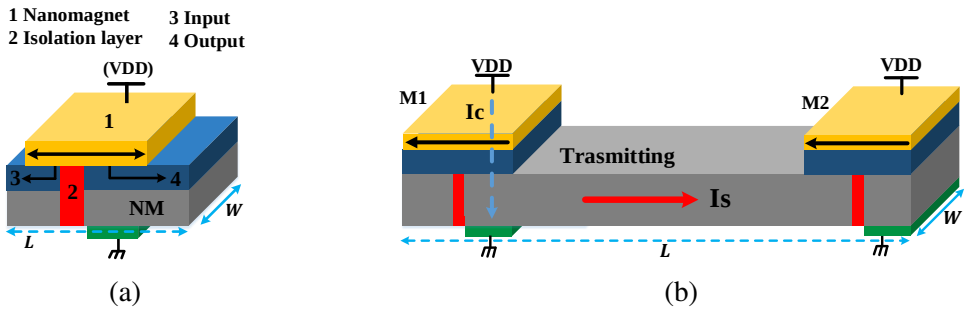


Figure 5.2: ASLD technique (a) Layout of ASLD (b) a simple layout of ASLD with two magnets.

Meanwhile, M2 will receive a large spin current from M1. The direction of M2 will not be changed because both M1 and M2 have the same magnetization direction. Therefore, the whole design will work as a buffer. In contrast, on applying positive VDD, the electrons will flow from the ground to the M1. As a consequence, spins with the opposite direction of the magnet will accumulate in the channel. Meanwhile, only the spins with the same direction of M1 will pass out of M1 while the spins with the opposite direction will be moved through the channel to switch the direction of M2. Therefore, the device will work as inverter [173]. Note that the Landau-Lifshitz-Gilbert (LLG) should be applied at the output of each implemented ASLD to model the magnetic field of the output magnetics.

5.2.1.2 ASLD as PLGs

The main idea of using the polymorphic logic gates (PLG) is to obtain different logic gates with the same structure. For instance, a PLG in [87] using SiNW technology can perform as an AND or OR gate when the VDD is '1' or '0' logic, respectively, which is useful in many applications, such as aerospace purposes and intelligent system. For IP protection, the designer can conceal the functionality of the circuit by inserting PLGs. The functionality will not be exposed even if an assailant can get the encrypted design by reverse engineering because only the designer knows the correct configuration signals for the PLGs.

The ASLD can naturally perform as a majority gate (MG) operation. The principle of the MG is that the value of the primary output relies on the values of the majority inputs. As a result, the ASLD can implement any logic gate. For instance, a designer can easily get an N-input NOR gate by making the directions of the fixed and output magnet layers apart. By inverting the direction of the fixed layer, the design operates as an N-inputs NAND gate. To get AND and OR gates, one more magnet layer has to be added at the primary output.

As a result, the ASL device allows us to change the functionality of the circuit with the same structure and without any extra hardware by making one of the primary input as an external key. As shown in Figure 5.3 (a), the structure of ASLD can perform as four different gates with the same circuit, such as AND, OR, NAND, and NOR, using only four magnets. Where A and B are the primary inputs, Key (K) and VDD are used to change the functionality of the circuit. The design can be switched from AND to OR by only changing the value of the key from '0' to '1', and set VDD positive. On applying negative VDD, the design can work as a NAND or a NOR gate if the value of the key is '0' or '1', respectively. Instead of making the VDD as a key to get a NAND or a NOR gate, one can apply only positive VDD and add one more magnet at the output of the AND or the OR gate, respectively. This way will avoid us using the VDD as a key.

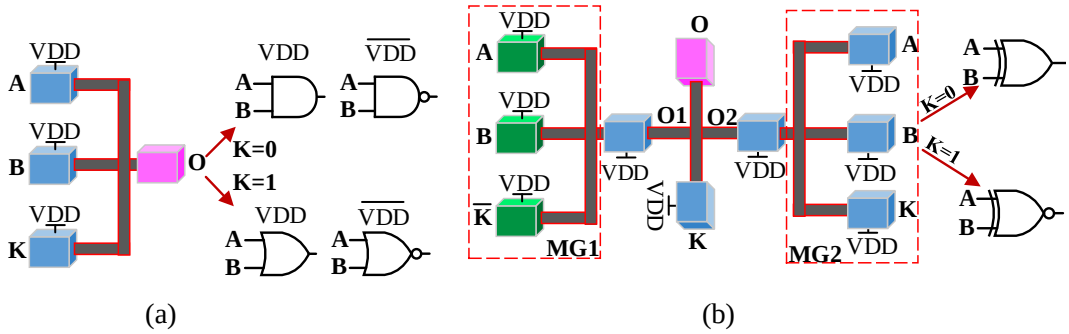


Figure 5.3: ASLD technique as PLGs: Layout of (a) AND/OR/NAND/NOR and (b) XNOR/XNOR PLGs with the same structure. Note that "A" and "B" are primary inputs and "K" is the key-input.

Table 5.1: Truth table of the ASLD-based PLGs in Figure 5.3

ASLD configurations			
	K	VDD polarity	operation
Fig. 5.3 (a)	0	pos	AND
Fig. 5.3 (a)	1	pos	OR
Fig. 5.3 (a)	0	neg	NAND
Fig. 5.3 (a)	1	neg	NOR
Fig. 5.3 (b)	0	pos	XOR
Fig. 5.3 (b)	1	pos	XNOR

Two options have already be introduced by Augustine et al. [173] to design a universal Full Adder (FA). The first one requires 44 magnetic units (using many NAND gates as majority gates), which is very expensive. The second way requires two-stage of MGs to implement a 3-input XOR gate as a sum of FA. The idea that the author used to implement the sum of the FA is to implement 3-input MG and then implement 5-input MG whose inputs are two inverted outputs, coming from the first 3-input MG, and the 3-input of the first implemented MG. In this case, the design will work as a three-input XOR gate (sum of FA). We Previously [89] realized that the structure can also function as an XOR or an XNOR gate by only making one of the primary input as an external key with magnetic free layer, in which it will be an XOR gate if the key is '0' or an XNOR gate

when the key is ‘1’. Even though this XOR/XNOR PLG requires a small number of magnets, its energy consumption and delay are more than twice the normal MG. This is because the output of the first MG should be stored temporarily. Then, this output is inverted, and two copies of it feed as two new inputs with the three primary inputs to another 5-input MG. Therefore, it will delay the whole stage in the circuit by more than twice since the output of this XOR/XNOR needs two clock cycles to reveal. More details regarding this implementation are clarified in [89].

Apart from using this approach, we use two 3-input MGs based on ASLD to implement XOR/XNOR PLG, as shown in Figure 5.3 (b). We implement the first MG as AND/OR MG (denoted inside a red dotted rectangle as MG1 in 5.3 (b)) and the second one as NOR/NAND MG (denoted inside a red dotted rectangle as MG2 in 5.3 (b)). Afterward, the outputs of those two MGs are inverted by adding one more ASLD with negative VDD at the output of each MG. Finally, we insert another ASLD to build the new XOR/XNOR PLG with a valid key-bit. The design will perform as an XOR gate if the value of the key (K) is ‘0’ (hence the first inverted MG will be NAND and the second is OR), while it will function as an XNOR gate if the key value is ‘1’ (hence the first inverted MG will be NOR and the second will be AND). The summarization of our simulation results, for different ASLD devices based PLGs, is described in Figure 5.4. To implement the normal basis LUT, we first design the decoder with the logic gates that we designed. Then, the output of the decoder is AND-ed with the data bits of the LUT to generate the output of the LUT. The ASLD is nonvolatile, and therefore the shift register can be realized by connecting multiple nanomagnets in series. Using the same topology introduced by Yuhao Wang et al. [128], we add redundant bits shift to perform circularly shifting. More details regarding the implementation of encrypted circuits using ASLD is provided in the next section.

5.2.2 PA Attack Prevention Using ASLD

In traditional CMOS, different input data to the same operations consume different powers due to the switching statement (‘0’ to ‘1’ or ‘1’ to ‘0’), while the MTJ-based spintronic logic unit produces four power differences in the current values. These values make the design vulnerable to PA attacks. On the other hand, the principle operation of ASLD is based on the magnetization direction of the spin current that flips the orientation of the magnet at the target or not, which always consumes constant power. More specifically, the ASLD device always consumes approximately the same power whether the supply voltage is positive or negative according to the targeted flipping of the magnet. This could be considered as a unique property in hardware security applications. A designer can implement a very resilient lightweight cryptographic algorithm, such as AES, DES, ECC, etc, using ASLD with high performance (small area with low switching power) against PA attacks without the need to add extra hardware resources. The PA attacks cannot figure out the correct secret key of an implemented cryptography designed using ASLD since the switching of the magnetization direction at each output stage depends on the direction of the generated spin current during the operation, as mentioned in the previous subsection. The spin current value that is required to switch the magnet is the same for both directions.

To verify the above-mentioned theory analysis, the Current-In-Plane (CIP) non-local spin valve model [174] is adopted to characterize ASLD. A simple two-input XOR/XNOR PLG, as illustrated in Figure 5.3 (b), is designed using the HSPICE simulator. We supply 2.664 mA of current for the duration of 8 ns to each nanomagnet with three sequences of input data (input A and input B normalized moments with a key-bit) to the design. The sequence of the output data response (output normalized moment) with the output current value is observed and reported, as shown in Figure 5.4. The simulation results of the design emphasize that the value of the output current has not changed during the four possible switching transient statements (‘0’ to ‘1’, ‘1’ to ‘0’, ‘0’ to

‘0’, and ‘1’ to ‘1’) at the output of the XOR/XNOR PLG. This could provide a highly resilient design against PA attacks. Note that the output current waveform in Figure 5.4 shows some spikes, which are generated due to the coupling noise, that is because ALSD does not have a capacitor at the output as in CMOS technology.

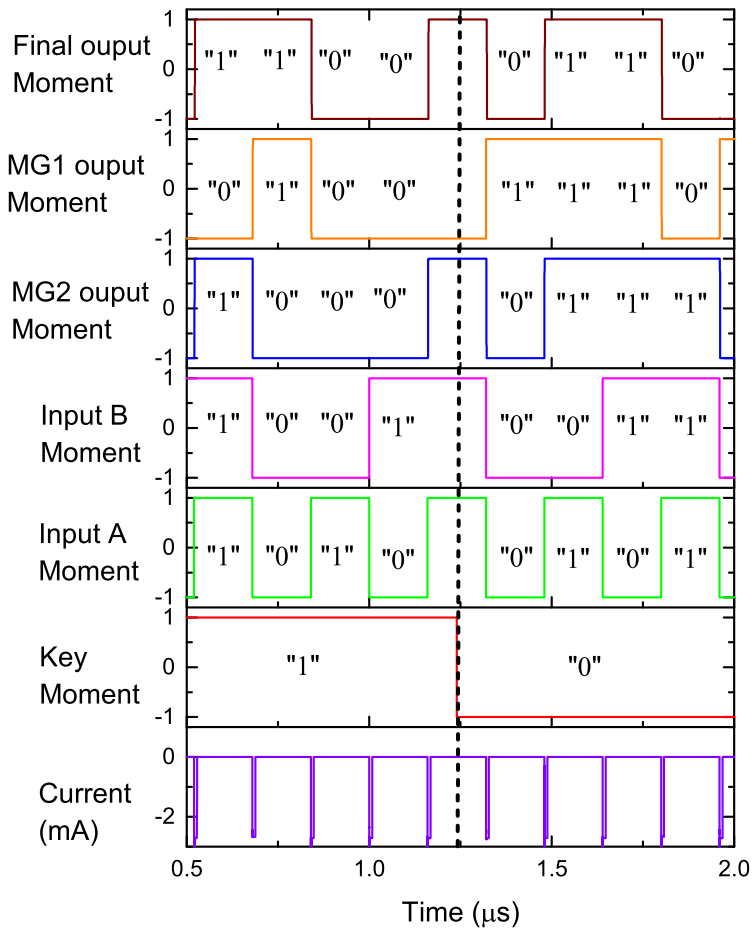


Figure 5.4: The ASLD simulation results show unchangeable output current during the switching of the XOR/XNOR PLG based ASLD.

The ASLD has two limitations; first, the spin current cannot be transferred to a long-distance or distributed to many paths or outputs (fan-out) since the value of the spin current is very small.

Second, the ASLD is slow compared to CMOS. The switching delay can be reduced via increasing the supplied voltage; however, the power consumption will be increased as well. In other words, the energy dissipation will be large. Since we are looking to design resilient AES with lower power and smaller area, we keep the small supplied voltage and utilize the pipeline technique to improve the energy of the design. In the next section, we provide some techniques to overcome these two limitations based on the work in [175].

5.3 Strong Logic Encryption and PA attack-Resilience

5.3.1 Robust Encryption against IC Attacks

Traditional logic encryption can only be achieved by adding key-gates (XOR/XNOR, MUX, and/or AND/OR gates). The overhead of power and area is more than 31% and 20%, respectively, when the number of inserted key-gates is only about 5% of the total number of gates [68]. Encrypting a circuit with a small number of keys, about 128 bits, will make strong reverse engineering attacks, such as sensitization, SAT, and D-DIP attacks, decrypt the encrypted circuit easily. These attacks can be prevented if the key-size is increased to a few thousand. However, increasing the key-size to such a number prevents the chip from being fabricated on real silicon due to the enormous power and area overhead. Besides this overhead, a large non-volatile memory should be added to store the secret key of the encrypted circuit.

Interestingly, by using ASLD technology, the circuit can be encrypted without adding any extra resources since each simple 2-input gate based ASLD can provide free key-bit. Note that each of the generated keys is already connected to a nanomagnetic unit. Therefore, there is no need to add extra memory since the keys can easily be stored in these nanomagnetic units. Unfortunately, implementing a large scale circuit using ALSDs suffers two problems. First, the spin current is too

small, so it cannot be distributed to large fan-out or transferred for a long distance. Second, the ASLD is slower than CMOS device. Even though the switching time can be reduced by increasing the current supply through the resistor, this will increase the power dissipation. These two problems have been overcome by adopting the pipeline technique and connecting all nanomagnetic units at each stage together.

The pipelining technique is commonly wide used in many applications to speed up the design, especially in modern computer systems and architectures. In traditional CMOS design, the pipeline can be achieved by incorporating flip-flops or registers between the stages, which increases power and area overhead significantly since each D flip-flop requires at least 12 transistors. In ASLD, the pipeline can be done by inserting ASLD buffer-gates only at the required paths in the stages, which incurs little power and area penalties since each added ASLD buffer-gate is very small compared to D flip-flop. Therefore, we leverage the pipeline to speed up the implemented circuit based ASLD with low overhead. Furthermore, the charge current in ASLD flows from VDD to GND (or GND to VDD based on the supply voltage) and its value does not decrease during the switching operation because the resistor value of the magnet is very small (few Ω). Consequently, the end terminal of the device can be used to connect all ASLDs in the same stage with one power supply. Since not all of the pipeline stages toggle at the same time (e.g. the output of a certain ALSD is input to the next stage), two clock phases are added, in which odd stages are performed first and then even stages sequentially. Also, note that the spin current is very small and cannot be transferred to a long-distance or drive in many paths (fanout). Therefore, we add ASLD as a buffer-gate once the gate output is connected to many paths, or the distance between two ASLDs is longer than the Spin flip length of the channel (λ_{sf}) (more details regarding this matter is in [176]). The added ASLD buffers and the usage of the pipeline technique render the circuit work at a low current supply (2.664 mA) with much higher energy-efficiency.

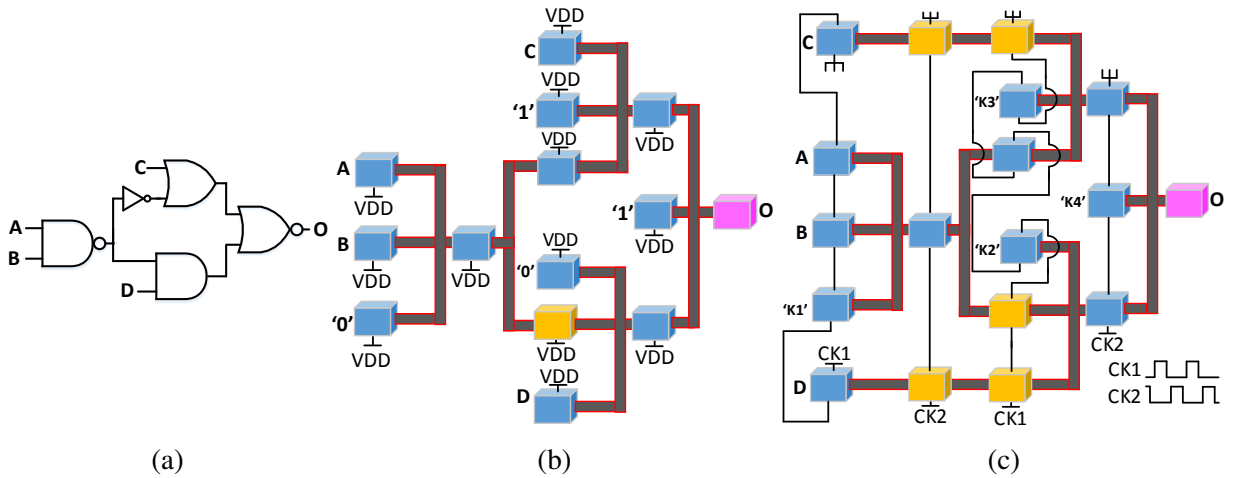


Figure 5.5: Example of implementing an encrypted circuit using ASLD (a) original circuit in CMOS (b) implementation of the circuit using ASLD with inserting buffer once the output has many fanout (c) incorporating two phase-clocks and applying the pipeline to generate efficient encrypted circuit with valid keys.

Figure 5.5 gives an example to summarize the aforementioned improvements on a circuit designed using ASLD. A simple logic circuit designed using CMOS is shown in Figure 5.5 (a). The circuit has four different logic gates (AND, NAND, OR, and NOR gates) with an inverter. The implementation of this circuit using ASLD is shown in Figure 5.5 (b). The output of the ASL NAND-gate is connected to two paths. The upper path has an inverter, which neglects the need for adding a buffer. In the second path, a buffer is inserted before the ASL AND-gate to enhance the spin current (the inserted ASL buffer-gate is donated in orange color in Figure 5.5 (b)). To apply the pipeline technique, the end terminals of all devices in each stage are connected to one clock-phase, and buffers are inserted in the required locations, as illustrated in Figure 5.5 (c), where all of the necessary inserted buffers are donated in the orange color. For each odd stage, clock-phase1 (CK1) is employed while for each even stage clock-phase2 (CK2) is added. Therefore, the ASLD based encrypted circuit has 4 key-inputs without adding any extra hardware since each 2-inputs ASL-logic gate offers one free key-bit. Such optimized strategy helps speed up the design with a very small performance penalty. The security analysis of this design is discussed in sub-section 5.4.3.

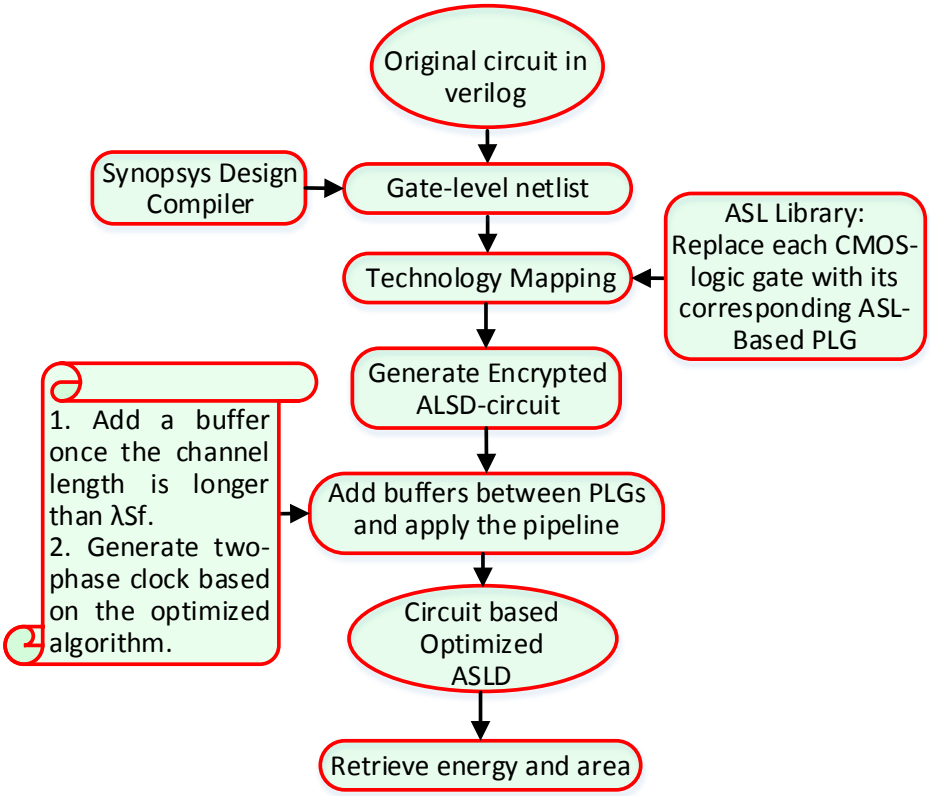


Figure 5.6: Implementing efficient ASLD based encrypted netlist

The flow-chart in Figure 5.6 shows the steps of implementing efficient circuit against IC and PA attacks. The first step is to synthesize the original verilog circuit using Synopsys Design Compiler. Then, Java has been used to generate ASLD based the encrypted circuit by replacing each CMOS-logic gate with its corresponding ASL-PLG. All ASL-PLGs have been stored in the ASLD library. The ASLD library has various PLGs (e.g. inverter/buffer, AND/OR, NAND/NOR, XOR/XNOR PLGs), each PLG has different input-sizes. The key-size of the encrypted circuit is equal to the number of gates in the original circuit since each designed ASLD-PLG has one key-input. Afterward, buffers are inserted between the stages of the PLGs. A buffer is inserted once the distance between the sender and the receiver ASLDs is longer than the Spin-flip length of the channel (λ_{sf}) [176], or the output signal goes to more than two paths. To apply the pipeline with the gener-

ated multi-phase clock, the gates at each level in the design have been specified. Then, a two-phase-clock is generated at each stage (level), and buffers are inserted in different necessary locations. The two-phase-clock generation is achieved using the global optimization algorithm [177].

5.3.2 PA Attack Resilience

As mentioned, the secret key of a hardware device can be revealed by PA attacks due to the differential power during the switching at the normal operation. PA attacks should choose a good location in the IC to attack. We take AES cryptography as an example since it is very secure to transfer data among users. AES has two vulnerable locations to PA attacks [44]; one is at the last round (LR) and the second is at the first round (FR), as shown in Figure 5.7. The attacker at the LR can calculate the ciphertext as SR(SSB_LR) $\oplus$ KL and then recover the original key via reversing the key-schedule routine. The more critical place to be attacked is at the FR since the original key is directly XOR-ed with the PT, and the target data can be computed as SSB_FR(K0$\oplus$PT). Where KL is the generated round key at LR, and K0 is the original key. Therefore, we assume during our evaluation on ASLD-based AES that the point of the PA attacks is at the FR. The PT, SSB, and K0 are 16 bytes. Measuring the whole 16 bytes one time to recover the key will be very time-consuming. However, if one byte of SSB_FR and K0 is revealed, the rest of the bytes can be recovered accordingly using the divide and conquer method, in which one byte is observed at a time. Note that the main requirements of implementing AES are XOR-gate, shift register operations, and LUT. These components are already elucidated in detail in subsection 5.2.1. It is important to mention that the key-bits of the cryptography can be increased significantly by adding the generated key-bits from the logic gates in the cryptography to its secret key-bits as an assistant key (generated from the PLGs during the traditional logic encryption). The assistant key will not increase the cryptography criteria, e.g. the strict avalanche criterion (SAC) [178], but will make the cryptography very strong against brute force and PA attacks.

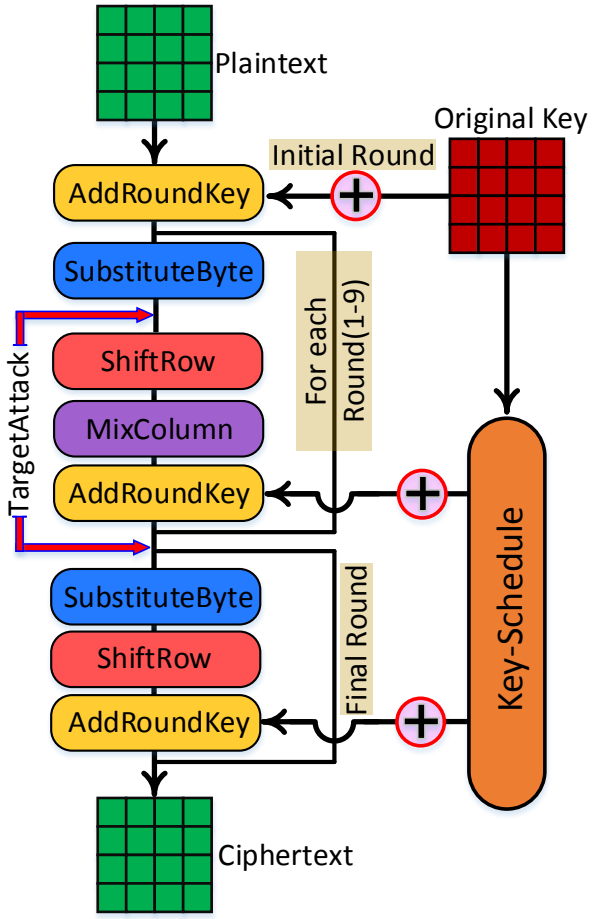


Figure 5.7: Block diagram of the AES with two points of PA attack.

5.4 Results

In this section, we elucidate the simulation results in detail to demonstrate the effectiveness of our proposed technique leveraging ASLD as PLGs to produce a secure and resilient circuit against IC and PA attacks. We also show a comparison between ASLD and CMOS in the required area and energy dissipation for the entire design.

5.4.1 Experimental Setup

The combinational ISCAS'85, sequential ISCAS'89, and ITC'99 benchmark circuits are used in our experiments. The CIP non-local spin valve model is leveraged to design different PLGs (inverter/buffer, AND/NAND/OR/NOR, and XOR/XNOR) with different input-sizes, the shift register, and the LUT. All these components are built and simulated using the Synopsys HSPICE simulator. We use the Synopsys Design Compiler to synthesize the Verilog code of the original netlist. We develop an in house modified ASLD library to perform technology mapping with the technology parameter in [179]. We use Java to produce the ASL-based encrypted circuit, where the program takes each of the original benchmark circuits with all ASL-based PLGs as inputs. The evaluation against different reverse engineering and PA attacks is achieved using the tools in [74, 94, 180]. For evaluation, we firstly convert each ASL-based encrypted circuit to Verilog gate-level circuit using Java, where each PLG is replaced by two gates with a multiplexer, e.g. inverter/buffer PLGs are replaced with CMOS-inverters, buffers, and multiplexers. Then, the Berkeley ABC tool is used to convert the encrypted circuits from Verilog to bench to evaluate them against SAT and D-DIP attacks. SAT and D-DIP attacks are ran on a cluster that has CPU 3.5GHz Core i7 and RAM 32GB. All benchmark circuits are synthesized using Synopsys Design Compiler with our developed ASLD and 14nm CMOS libraries.

5.4.2 Performance and Device Parameters

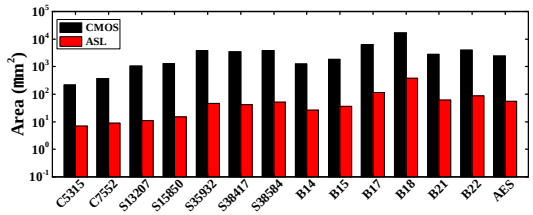
Even though the required voltage to switch the magnetization direction of ASLD (or other spintronic devices) is very small, the switching time is typically longer than CMOS. In ASL, the switching time can be reduced by increasing the supply voltage, but the power dissipation will also increase accordingly. Therefore, the metrics to evaluate a design implemented using the spintronic device are based on the energy (Power delay produce (PDP)) and the area. To accelerate

the proposed technique and enhance energy performance without increasing the supply voltage, we employ the pipeline technique. We use the ASLD supply input current of 2.664 mA. The setup parameters of the ASLD based physics framework are shown in Table 5.2.

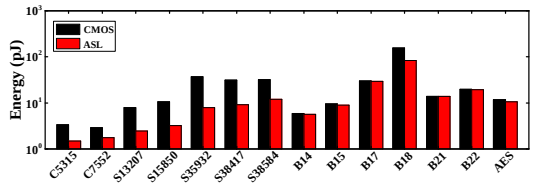
Table 5.2: Simulation parameters of ASLD

Parameter	setup value
Effective damping Alpha (α)	0.01
Spin-flip length of the channel (λ)	500 nm
Interconnection resistivity (ρ_N)	7 Ω-nm
Nano-magnet area (A)	30X10 nm^2
Interconnection element	Cu
Anisotropy field H_K	12.2 KOe
Saturation magnetization (M_S)	800 emu/cm^3
Magnet volume (V)	45X15X1 nm^2
Spin polarization ratio (P)	0.1/0.90

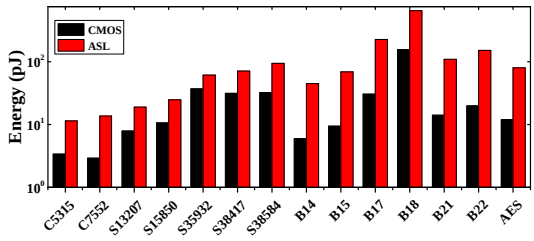
We evaluate the required energy and area of different benchmark circuits and compare them with CMOS. The key size has been set to be equal to the total number of gates in each netlist to get the maximum required energy and area. Figure 5.8 (a) shows a comparison between *original CMOS-benchmark circuits* and ASL-based encrypted circuits. On average, the required area size using ASLD is about 40X less than the required one by CMOS technology. We also compare the required energy by CMOS and ASLD for different benchmark circuits at 45 MHz and 100 MHz, and the results are as shown in Figure 5.8 (b) and (c), respectively. When the frequency operation is at 45 MHz, the ASLD outperforms CMOS by 0.56X (56%). However, CMOS is better than ASLD by 4.3X when the frequency operation is at 100 MHz. Therefore, ASLD is very good for applications that not required a very high-frequency operation.



(a)



(b)



(c)

Figure 5.8: A comparison between circuits implemented using CMOS and ASLD (a) Area, (b) energy at 45 MHz, and (c) energy at 100 MHz.

5.4.3 Security Analysis of the Proposed Techniques

In this section, we will briefly discuss the resilience of our proposal against PA and IC attacks. This approach produces a secure and resilient design and does not require any hardware resources, e.g. non-volatile memory, logic key-gates, and SABL. An attacker can perform different techniques to get the secret key of an encrypted circuit, as in the following:

5.4.3.1 Leveraging brute force attack

The secret key of an encrypted IC can be revealed via an attacker on supplying all of the key combinations if the length of the key is not large. Such an offense can be prohibited by enlarging the size of the secret key. In conventional CMOS technology, increasing the size of the secret key requires more inserted key-gates (because one of each inserted-gate input has to be a key-bit). This will largely increase the performance overhead, especially for small scale benchmarks, where the total overhead of a small encrypted circuit might override the original circuit-size. In our proposal, implementing a circuit using ASLD can give a key-bit for any 2-inputs logic gate without any extra hardware. Therefore, one can freely increase the size of the key to maximum N times, where N is the number of gates in the original circuit. For a large scale circuit, the locations and the required size of the key-bits can be specified by the designer. All other extra key inputs should be connected to VDD or GND based on the configuration of the original gate.

5.4.3.2 Sensitization attack

Sensitization attack can be prevented by making a relationship among the key-gates. If an attacker wants to propagate a key-bit to an output, a large number of dependent key-bits should be known. In tradition encryption using CMOS technology, each inserted key-gate has only two inputs; one represents the key-bit, and the second is connected to an internal net. In ASLD-based encryption, each key-bit is connected with at least two other inputs to an original gate. Therefore, generating a pattern to propagate a key-value to output will be much harder. Also, the selected key-bits should be chosen in sequence to make such attack works almost the same as the brute force. For instance, the simple encrypted circuit using ASLD in Figure 5.9 has 5 key-bits. The key-bit of G1 and G2 is connected to one key-bit (K1), and the key-input of G5 is connected to VDD to show how a designer can reduce the key-size. The key-bit of each of the rest gates is connected to an external

key. Now, to propagate the value of K1, K3, or K5 to output 2 (O2), the two other keys should be known. Similarly, if we want to propagate K1, K2, or K4 to O1, also the two rest keys should be known. Also, it is much harder to generate a pattern that can propagate a key-value to an output since each key-gate has at least two inputs besides the key-input. Note that most of the real-complex ICs have gates with large fan-in. This increases the ambiguity of such attack much harder to find a key-bit.

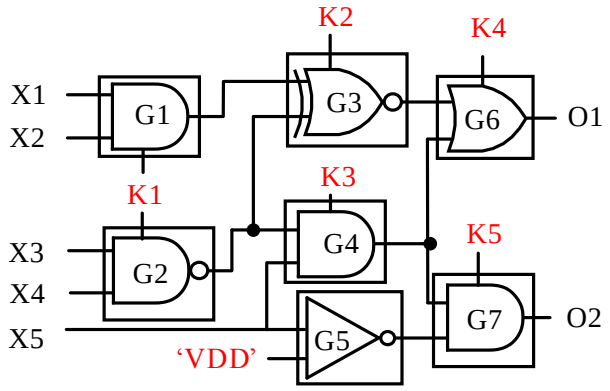


Figure 5.9: Preventing sensitization attack from propagating key values to outputs

5.4.3.3 SAT and D-DIP attacks

The best-proposed technique to prevent these two attacks is to incorporate large TOA block with some modifications on the original netlist. However, it requires a very large overhead and offers a trade-off between removal and SAT attack resilience. Also, for high protection level, its key size should be long enough, e.g. > 512, and it should keep working at low corruptibility. Otherwise, it will be subject to FALL attack [100].

If the key-size of the traditional encrypted circuit can be increased to > 3k, these two attacks will fail to decrypt such complex circuit. Unfortunately, this technique cannot be achieved using

Table 5.3: Results of SAT and D-DIP attacks against circuits encrypted using ASLDs with different baseline key-size, where Bs and TO are bits and timeout, respectively.

circuits	Execution time of SAT					Execution time of D-DIP				
	128 Bs	256 Bs	512 Bs	1024 Bs	2048 Bs	128 Bs	256 Bs	512 Bs	1024 Bs	2048 Bs
C5315	1.7	2.33	7.96	364.98	TO	2.12	6.03	47.44	347.5	TO
C7552	2.05	4.7	28.5	993.8	TO	2.16	6.37	74.54	1487.82	TO

CMOS since the area and power overhead will be enormous, especially for a small-scale circuit. The ASLD-based encrypted circuit is naturally resilient to both SAT and D-DIP attacks since the key-size of the traditional encrypted circuit can be enlarged up to the total number of gates in the original circuit. Practically, we convert the two smallest ASLD-based encrypted circuits, C5315 and C7552, to CMOS encrypted bench circuits to evaluate them against SAT and D-DIP attacks. Table 5.3 shows the execution time that both attacks need to decrypt these two circuits. When the key-size is smaller than or equal to 1k, both attacks can successfully decrypt them. However, both fail to break either benchmark when the key-size is set to 2k within three days. We repeated the experiment for statistical value, and both attacks still failed to decrypt either one. Note that the proposed ASL-based encrypted circuit is already resilient against other attacks (removal, approximate SAT, and ByPass attacks) since there is no additional block, e.g. TOA, has been integrated with the proposed technique.

5.4.3.4 PA attack resilience

We test our proposal using the CPA since it is more accurate and requires less number of traces to figure out the correct key. The simulation results of the ASLD XOR-gate are not enough to prove that the proposal can prevent PA attacks. To further evaluate the resilience of our technique, we target the intermediate data in the SBOX at the first round since it is the only part that directly treats with the original secret key, as shown in Fig. 5.7. We assume that the attacker can get access to the

primary inputs of the proposed AES, supply the input data (PT), and observe the power dissipation. We select one byte from the output result of the SBOX as a point of attack.

The power model can be created using either the Hamming weight (HW); (it is a simple model to calculate the estimated power dissipation for each set bits (count the number of set bits in the data word)), or the hamming distance (HD); it is the relationship between the previous and the new data in the data set (the data set before and after the SBOX). Each of these power models has different properties. For stronger evaluation, we test the proposed AES against PA attacks using both the HW and HD models. Each model is built based on the work in [180], where 1% Additive White Gaussian Noise (AWGN) has been included. We apply each of the possible PT values, each with all possible combinational key values to create the two hypothetical power models. Afterward, we apply the correct key of AES and measure real power consumption. We sample the real power with 4ns sampling period. The power models are compared with the real power. Then we calculate the correlation between the sampled power and the two generated hypothetical models utilizing Equation 5.1. The highest resilient AES against PA attacks can be achieved if the correct key trace is mixed with the incorrect key traces while it is considered very weak if the correct key is stepped away from the incorrect keys.

$$Corr(p*c) = \frac{\sum_{k=1}^{256}(t_{k,p,c} - \overline{t_{p,c}}).(h_{k,p,c} - \overline{h_{p,c}})}{\sqrt{\sum_{k=1}^{256}(t_{k,p,c} - \overline{t_{p,c}})^2 . \sum_{k=1}^{256}(h_{k,p,c} - \overline{h_{p,c}})^2}} \tag{5.1}$$

Figure 5.10 (a) shows the correlation outcomes of applying the correct key and incorrect keys on the proposed AES using the HW, while Figure 5.10 (b) shows the outcome results based on the

Where:

- $p * c$: number of PT (256) * number of traces (17).
- k: number of guess key (256).
- t: real power.
- h: hypothetical power model.

HD. Note that implementing ASLD-based AES with different key-sizes will not affect on the PA attack-resilient AES since we took all possible cases on the selected byte as the point of attack and the guessed key. In fact, implementing an IC using ASL-based encryption makes it secure against IC attacks (during the fabrication) and PA attacks (after the fabrication).

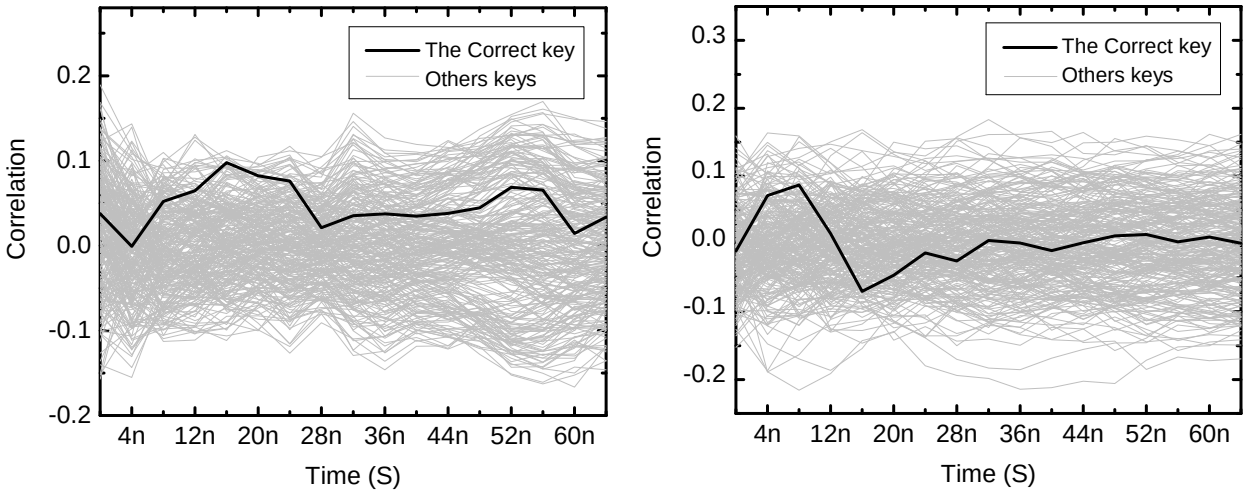


Figure 5.10: CPA attack vs the execution time: (a) correlation-based on the HW (b) the HD.

The secret-key size of a cryptography implemented using ASLD can be enlarged via adding assistant key-bits that are generated from the logic gates in the cryptography. The assistant key-bits

increase the resilience of the cryptography against PA and brute force attacks significantly. It is worthwhile to mention that the previous works [181, 181] have proposed to implement AES without taking into consideration the vulnerability of the design against PA attacks. Preventing such attacks requires a large surcharge for a high protection level of AES designed using only CMOS technology. Meanwhile, implementing resilient AES against PA attacks using hybrid Spintronic-CMOS devices, including MTJ has not been presented yet.

5.5 Summary

In this paper, we have investigated the usage of ASLD that can help improve in hardware security application domains. We extract and evaluate the unique properties of ASLD, which can be utilized to implement different PLGs and provide unchangeable power dissipation without external resources during the transient statements. We also leverage the pipeline technique to enhance the performance of the design, instead of increasing the power supply. The proposal has been evaluated against both reverse engineering and power analysis attacks. Our analyses, supported by the evaluation results, emphasize that ASLD-based encrypted circuit can provide a highly secure and resilient design against IC and PA attacks with a smaller area and lower energy. In our technique, the untrusted foundry cannot break the encrypted design, during the fabrication using reverse engineering tools, and the secret key cannot be learned, after the chip fabrication based on the leakage information. As a result, the proposed technique could be a good candidate for the internet of thing applications.

CHAPTER 6: CONCLUSION and FUTURE WORK

The increasing cost of nanoscale device manufacturing has resulted in the proliferation of "fabless" integrated circuit (IC) design houses which outsource the production of ICs to overseas foundries. While there are many benefits to the de-verticalization of semiconductor manufacturing, it also poses some threats. In particular, untrusted foundries increase the risks of hardware Trojan, IC piracy, counterfeiting, and reverse engineering: resulting in intellectual property (IP) theft and overproduction [25, 182, 183]. Each of these threats can cause significant financial losses to the semiconductor industry [40]. To thwart such attacks, many countermeasure techniques have been presented, such as metering, split manufacturing, watermarking, and camouflaging. However, these methods can only prevent certain hardware attacks.

Interestingly, it has been shown that logic encryption can successfully mitigate the most severe IC hardware attacks that can possibly be leveraged by an untrusted overseen company since logic encryption assumes all parts in IC design flow untrusted, except the in-house design. Moreover, after the encrypted circuits successfully pass the untrusted foundry, the key inputs should be driven by on-chip tamper-proof memory [72]. Unfortunately, the embedded secret key of the encrypted circuit, including the smart devices, e.g., cryptographies, can be revealed by power analysis (PA) attacks [41, 45]. Therefore, extra hardware resources should be added to mitigate PA attacks.

In chapter 3, we present two traditional logic encryption techniques, half/full MUX insertion and hybrid CMOS-SiNW. The MUX insertion based logic encryption offers high security (50% HD) for whatever the circuit size is. Also, random key generation circuit has been designed using LFSR with hardware units to increase the security level of the encrypted circuit, in which the locations of the corrupted outputs will change on applying wrong keys. To get 50% HD when a wrong key is provided with very small overhead, instead of incorporating key-gates, three simple

PLGs, designed using SiNW, are exchanged with corresponding gates in most of the ISCAS'85 combinational circuits.

In chapter 4, SiNW PLG-based traditional encryption is augmented with a small block, whose output is untraceable, namely URSAT. This technique can prevent most of the existing IC attacks, such as sensitization, SAT, and AppSAT, and removal attacks. Although URSAT is secure against many attacks, it may not offer strong resilience against the combined AppSAT-removal attack. To strongly thwart this attack, we propose S-URSAT designed using only CMOS logic gates. The compounded S-URSAT and PLG-based encryption increases the security level of the design to robustly thwart all existing attacks.

Chapter 5 presents comprehensive secure hardware circuits that withstand all IC attacks (during the fabrication) and PA attacks (after the fabrication). We firstly design and implement optimized ASLD-based PLGs that require very small energy dissipation and are much faster than traditional ASLD gates. These ASLD-based PLGs are used to produce encrypted circuit by only replacing the traditional CMOS logic gates with ASL based PLGs. The implemented circuit using ASLDs can be encrypted with a very complicate key without any adding extra resources. Therefore, the design is extremely resilient to IC attacks. The improvements on ASLD-based encrypted circuit have been achieved by employing optimal ASL parameters, using the pipeline technique, and adding ASL-based buffers at necessary locations. Moreover, we show that ASLD has another unique feature, which is identical power dissipation through the switching operations. This feature enables us to implement resilient designs against PA attacks with a small performance penalty. Finally, each presented technique in all of our works has been experimentally evaluated in terms of performance overheads and security guarantees.

Most of our work are based on protecting combinational logic circuits. However, the modern real word ICs are typically a mix of combinational circuits, final state machines, scan chains, and se-

quential circuits. In the future work, we suggest the research directions focus on building a fully protected hardware design that has all of the aforementioned components. Some of other interesting future work is to develop a high performance and secure monolithic 3D architecture design against hardware attacks with small cost using split manufacturing technique. Another possible future work is to produce a comprehensive secure system by combining the software and hardware protection techniques. This area is really interesting since the hardware developers are focusing on protecting the hardware design without taking into account the software possible attacks. Similarly, for the software developers. The most promising technique to prevent all hardware and software attacks could be achieved if the hardware and software security methodologies are mixed somehow together.

APPENDIX : COPYRIGHT PERMISSIONS

Logic Obfuscation against IC Reverse Engineering Attacks Using PLGs

Conference Proceedings: 2017 IEEE International Conference on Computer Design (ICCD)

Author: Qutaiba Alasad

Publisher: IEEE

Date: Nov. 2017

LIST OF REFERENCES

[1] J. A. Roy, F. Koushanfar, and I. L. Markov, "Ending piracy of integrated circuits," *Computer*, vol. 43, no. 10, pp. 30–38, Oct 2010.

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

[3] Y. Xie and A. Srivastava, "Mitigating sat attack on logic locking," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer Berlin Heidelberg, 2016.

[4] 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 the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 1601–1618. [Online]. Available: http://doi.acm.org/10.1145/3133956.3133985

[5] Karen Mercedes Goertzel, Booz Allen Hamilton, "Integrated circuit security threats and hardware assurance countermeasures," *REAL-TIME INFORMATION ASSURANCE*, 2013.

[6] S. Hu, Y. Jin, K. Heffner, and M. Tehranipoor, "Guest editorial: Hardware/software cross-layer technologies for trustworthy and secure computing," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 144–145, July 2016.

[7] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design Test*, vol. PP, no. 99, pp. 1–1, 2013.

[8] M. Yasin and O. Sinanoglu, "Transforming between logic locking and ic camouflaging," in *2015 10th International Design Test Symposium (IDT)*, Dec 2015, pp. 1–4.

[9] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," in *IEEE, 102(8):1283-1295, 2014*, 2014.

[10] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, "Hardware security: Threat models and metrics," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2013, pp. 819–823.

[11] S. Brown and Z. Vranesic, "Fundamentals of digital logic with verilog design (mcgraw-hill series in electrical and computer engineering," in *McGraw-Hill Science/Engineering/Math; 1 edition*, August 23 2002.

[12] G. Smith, "Developments of multi-cad models," *IC CAD Market Trends, WWW.garysmithEDA.com*, 2015.

[13] R. Jacob Baker, "Cmos circuit design, layout, and simulation," in *IEEE Press Series on Microelectronic Systems*, 2008.

[14] J. Mazurek, "Making microchips: Policy, globalization, and economic restructuring in the semiconductor industry." in *Mit Press*, 1999.

[15] Defense Science Board (2005), "Defense science board (dsb) study on high performance microchip supply," in *March 16*, 2015.

[16] J. Rajendran, O. Sinanoglu, and R. Karri, "Regaining trust in vlsi design: Design-for-trust techniques," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1266–1282, Aug 2014.

[17] U. Guin, D. DiMase, and M. Tehranipoor, "Counterfeit integrated circuits: Detection, avoidance, and the challenges ahead," *Journal of Electronic Testing*, vol. 30, no. 1, pp. 9–23, Feb 2014. [Online]. Available: https://doi.org/10.1007/s10836-013-5430-8

[18] U. Guin and K. Huang and D. DiMase and J. M. Carulli and 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, Aug 2014.

[19] Y. M. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *16th USENIX Security Symposium (USENIX Security 07)*. Boston, MA: USENIX Association, Aug. 2007. [Online]. Available: https://www.usenix.org/conference/16th-usenix-security-symposium/active-hardware-metering-intellectual-property-protection

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

[21] Yousra Alkabani, Farinaz Koushanfar, and Miodrag Potkonjak, "Remote activation of ics for piracy prevention and digital right management," in *2007 IEEE/ACM International Conference on Computer-Aided Design*, Nov 2007, pp. 674–677.

[22] 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, Jan 2010.

[23] U. Guin and M. M. Tehranipoor, "Obfuscation and encryption for securing semiconductor supply chain," in *Hardware Protection through Obfuscation*. Cham: Springer International Publishing, 2017, pp. 317–346.

[24] G. K. Contreras and M. T. Rahman and M. Tehranipoor, "Secure split-test for preventing ic piracy by untrusted foundry and assembly," in *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, Oct 2013, pp. 196–203.

[25] U. Guin, Q. Shi, D. Forte, and M. M. Tehranipoor, "Fortis: A comprehensive solution for establishing forward trust for protecting ips and ics," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 21, no. 4, pp. 63:1–63:20, May 2016. [Online]. Available: http://doi.acm.org/10.1145/2893183

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

[27] Bi, Yu and Shamsi, Kaveh and Yuan, Jiann-Shiun and Gaillardon, Pierre-Emmanuel and Micheli, Giovanni De and Yin, Xunzhao and Hu, X. Sharon and Niemier, Michael and Jin, Yier, "Emerging technology-based design of primitives for hardware security," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 1, pp. 3:1–3:19, Apr. 2016.

[28] Y. Bi and K. Shamsi and J. S. Yuan and Y. Jin and M. Niemier and X. S. Hu, "Tunnel fet current mode logic for dpa-resilient circuit designs," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[29] W. P. Griffin and A. Raghunathan and K. Roy, "Clip: Circuit level ic protection through direct injection of process variations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 5, pp. 791–803, May 2012.

[30] IHS, "Reports of counterfeit parts quadruple since 2009, challenging u.s. defence industry and national security," *[Online]. Available: http://www.ihs.com/images/IHS-iSuppli-Reports-CounterfeitParts-Quadruple-Since-2009.pdf*, April 2012.

[31] F. E. (2011), "Estimating the global economic and social impacts of counterfeiting and piracy," in *A Report Commissioned by the Business Action to Stop Counterfeiting and Piracy (BASCAP)*, 2011, [Online]. Available: www.iccwbo.org/Data/Documents/Bascap/Global-Impacts-Study-Full-Report.

[32] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, “Hardware trojan attacks: Threat analysis and countermeasures,” *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug 2014.

[33] S. Narasimhan, D. Du, R. S. Chakraborty, S. Paul, F. G. Wolff, C. A. Papachristou, K. Roy, and S. Bhunia, “Hardware trojan detection by multiple-parameter side-channel analysis,” *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2183–2195, Nov 2013.

[34] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, “Trustworthy hardware: Identifying and classifying hardware trojans,” *Computer*, vol. 43, no. 10, pp. 39–46, Oct 2010.

[35] H. Salmani, *Design Techniques for Hardware Trojans Prevention and Detection at the Layout Level*. Cham: Springer International Publishing, 2018, pp. 93–107. [Online]. Available: https://doi.org/10.1007/978-3-319-79081-7_7

[36] R. Torrance and D. James, “"the state-of-the-art in semiconductor reverse engineering",” in *in Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE, June 2011, pp. 333–338.*, 2011.

[37] Chipworks, ““intel’s 22-nm tri-gate transistors exposed”,” in *[Online].Available: http://www.chipworks.com/blog/technologyblog/2012/04/23/intels-22-nmtri-gate-transistors-exposed/*, 2012.

[38] D. A. R. P. Agency, “"trust in integrated circuits (tic)",” in *[Online]. Available: https://www.fbo.gov/spg/ODA/DARPA/CMO/BAA07-24/listing.html*, 2007.

[39] S. Bhunia and M. Tehranipoor, “"hardware ip piracy and reverse engineering",” in *Morgan Kaufmann is an imprint of Elsevier Inc. Book, chapter 7*, 2019.

[40] SEMI. (2008), “"innovation is at risk as semiconductor equipment and materials industry loses up to $4 billion annually due to ip infringement.",” *In*, 2008.

[41] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '99. Berlin, Heidelberg: Springer-Verlag, 1999, pp. 388–397. [Online]. Available: http://dl.acm.org/citation.cfm?id=646764.703989

[42] A. Schlosser, D. Nedospasov, J. Kramer, S. Orlic, and J.-P. Seifert, "Cryptographic hardware and embedded systems," in – *CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings. Springer Berlin Heidelberg, 2012, ch. Simple Photonic Emission Analysis of AES.*, 2012, pp. 41–57.

[43] P. Rohatgi, "Cryptographic engineering," in *Boston, MA: Springer US, 2009, ch. Electromagnetic Attacks and Countermeasures, pp. 407–430.*, 2009.

[44] K. J. Smith, "Methodologies for power analysis attacks on hardware implementations of aes," *Thesis. Rochester Institute of Technology.*, 2009.

[45] P. Kocher, "Design and validation strategies for obtaining assurance in countermeasures to power analysis and related," in *NIST Physical Security*, 2005.

[46] F. Zhang and Z. J. Shi, "Differential and correlation power analysis attacks on hmac-whirlpool," in *2011 8th ITNG*, 2011.

[47] R. Karri, K. Wu, P. Mishra, and Yongkook Kim, "Fault-based side-channel cryptanalysis tolerant rijndael symmetric block cipher architecture," in *Proceedings 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Oct 2001, pp. 427–435.

[48] A. Aghaie, M. Mozaffari Kermani, and R. Azarderakhsh, "Fault diagnosis schemes for low-energy block cipher midori benchmarked on fpga," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 4, pp. 1528–1536, April 2017.

[49] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A lightweight concurrent fault detection scheme for the aes s-boxes using normal basis," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, E. Oswald and P. Rohatgi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 113–129.

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

[51] F. Koushanfar and G. Qu, "Hardware metering," in *Proceedings of the 38th Annual Design Automation Conference*, ser. DAC '01. New York, NY, USA: ACM, 2001, pp. 490–493. [Online]. Available: http://doi.acm.org/10.1145/378239.378568

[52] F. Koushanfar, G. Qu, and M. Potkonjak, "Intellectual property metering," in *Proceedings of the 4th International Workshop on Information Hiding*, ser. IHW '01. London, UK, UK: Springer-Verlag, 2001, pp. 81–95. [Online]. Available: http://dl.acm.org/citation.cfm?id=647597.731874

[53] R. Jarvis and M. McIntyre., "Split manufacturing method for advanced semiconductor circuits,," in *US Patent 7,195,931.*, 2007.

[54] S. Garg and J. Rajendran, *Split manufacturing*. Springer International Publishing, 1 2017, pp. 243–262.

[55] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 1, pp. 6:1–6:23, May 2016. [Online]. Available: http://doi.acm.org/10.1145/2906147

[56] K. Vaidyanathan, B. P. Das, E. Sumbul, R. Liu, and L. Pileggi, "Building trusted ics using split fabrication," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, May 2014, pp. 1–6.

[57] Y. Xie, C. Bao, and A. Srivastava, "Security-aware design flow for 2.5d ic technology," in *Proceedings of the 5th International Workshop on Trustworthy Embedded Devices*, ser. TrustED '15. New York, NY, USA: ACM, 2015, pp. 31–38. [Online]. Available: http://doi.acm.org/10.1145/2808414.2808420

[58] J. Valamehr, T. Sherwood, R. Kastner, D. Marangoni-Simonsen, T. Huffmire, C. Irvine, and T. Levin, "A 3-d split manufacturing approach to trustworthy system development," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 4, pp. 611–615, April 2013.

[59] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking techniques for intellectual property protection," in *Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175)*, June 1998, pp. 776–781.

[60] SypherMedia, "Syphermedia library circuit camouflage technology," in *http://www.smi.tv/solutions.htm.*, 2015.

[61] M. Shiozaki, R. Hori, and T. Fujino., "Diffusion programmable device: The device to prevent reverse engineering." in *IACR Cryptology ePrint Archive*, 2014, p. 109.

[62] J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, and B. J. Wang., "Camouflaging a standard cell based integrated circuit." in *US Patent no.8151235,*, 2012.

[63] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Camoperturb: Secure ic camouflaging for minterm protection," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016.

[64] A. Moradi *et al.*, "Pushing the limits: A very compact and a threshold implementation of aes," in *Advances in Cryptology – EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, K. G. Paterson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[65] P. Liu *et al.*, "A low overhead dpa countermeasure circuit based on ring oscillators," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2010.

[66] X. Li *et al.*, "Energy-efficient side-channel attack countermeasure with awareness and hybrid configuration based on it," *IEEE Transactions on VLSI Systems*, 2017.

[67] K. Tiri *et al.*, "A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Solid-State Circuits Conference*, 2002.

[68] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *2008 Design, Automation and Test in Europe*, March 2008, pp. 1069–1074.

[69] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 410–424, Feb 2015.

[70] S. Dupuis, P. Ba, G. Di Natale, M. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*, July 2014, pp. 49–54.

[71] Y. Lee and N. A. Touba, "Improving logic obfuscation via logic cone analysis," in *2015 16th Latin-American Test Symposium (LATS)*, March 2015, pp. 1–6.

[72] P. Tuyls *et al.*, "Read-proof hardware from protective coatings," in *Cryptographic Hardware and Embedded Systems - CHES 2006*. Springer Berlin Heidelberg, 2006.

[73] Y. Liu, K. Huang, and Y. Makris, "Hardware trojan detection through golden chip-free statistical side-channel fingerprinting," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2014, pp. 1–6.

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

[75] F. Koushanfar, "Provably secure active ic metering techniques for piracy avoidance and digital rights management," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 51–63, Feb 2012.

[76] F. Imeson, A. Emtenan, S. Garg, and M. Tripunitara, "Securing computer hardware using 3d integrated circuit (IC) technology and split manufacturing for obfuscation," in *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX, 2013, pp. 495–510. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/imeson

[77] D. Kirovski and M. Potkonjak, "Local watermarks: methodology and application to behavioral synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 9, pp. 1277–1283, Sep. 2003.

[78] A. L. Oliveira, "Robust techniques for watermarking sequential circuit designs," in *Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361)*, June 1999, pp. 837–842.

[79] J. Baukus, L. Chow, R. Cocchi, P.O., and B. Wang., "Building block for a secure cmos logic cell library,," *US Patent no. 8111089.*, 2012.

[80] J. Baukus, L. Chow, R. Cocchi, and B. Wang., "Method and apparatus for camouflaging a standard cell based integrated circuit with micro circuits and post processing,," *US Patent no. 20120139582.*, 2012.

[81] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, "Provably secure camouflaging strategy for ic protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 8, pp. 1399–1412, Aug 2019.

[82] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 709–720. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516656

[83] A. Vijayakumar, V. C. Patil, D. E. Holcomb, C. Paar, and S. Kundu, "Physical design obfuscation of hardware: A comprehensive investigation of device and logic-level techniques," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 64–77, Jan 2017.

[84] 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, Feb 2019.

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

[86] A. Rezaei, Y. Shen, and H. Zhou, "Rescuing logic encryption in post-sat era by locking and obfuscation," in *Cryptology ePrint Archive, Report 2019/1463*, 2019, https://eprint.iacr.org/2019/1463.

[87] Q. Alasad and J. Yuan, "Logic obfuscation against ic reverse engineering attacks using plgs," in *2017 IEEE International Conference on Computer Design (ICCD)*, Nov 2017, pp. 341–344.

[88] Q. Alasad, J.-S. Yuan, and Y. Bi, "Logic locking using hybrid cmos and emerging sinw fets," *Electronics*, vol. 6, 2017.

[89] Q. Alasad, J. Yuan, and D. Fan, "Leveraging all-spin logic to improve hardware security," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, ser. GLSVLSI '17. New York, NY, USA: ACM, 2017, pp. 491–494. [Online]. Available: http://doi.acm.org/10.1145/3060403.3060471

[90] R. S. Chakraborty and S. Bhunia, "Harpoon: An obfuscation-based soc design methodology for hardware protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, Oct 2009.

[91] R. S. Chakraborty and S. Bhunia, "Security against hardware trojan through a novel application of design obfuscation," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, Nov 2009, pp. 113–116.

[92] J. Rajendran and Y. Pino and O. Sinanoglu and R. Karri, "Logic encryption: A fault analysis perspective," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2012, pp. 953–958.

[93] M. Yasin and B. Mazumdar and O. Sinanoglu and J. Rajendran, "Security analysis of anti-sat," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2017, pp. 342–347.

[94] Yuanqi Shen, and Hai Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, ser. GLSVLSI '17. New York, NY, USA: ACM, 2017, pp. 179–184. [Online]. Available: http://doi.acm.org/10.1145/3060403.3060469

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

[96] Y. Shen, A. Rezaei, and H. Zhou, "A comparative investigation of approximate attacks on logic encryptions," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2018, pp. 271–276.

[97] 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 *CHES*, 2017.

[98] M. Yasin, A. Sengupta, B. Schafer, Y. Makris, O. Sinanoglu, and Jeyavijayan Rajendran (JV), "What to lock?: Functional and parametric locking," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, ser. GLSVLSI '17. New York, NY, USA: ACM, 2017, pp. 351–356. [Online]. Available: http://doi.acm.org/10.1145/3060403.3060492

[99] K. Shamsi, T. Meade, M. Li, D. Z. Pan, and Y. Jin, "On the approximation resiliency of logic locking and ic camouflaging schemes," *IEEE Transactions on Information Forensics and Security*, 2019.

[100] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019, pp. 936–939.

[101] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," in *arXiv preprint arXiv:1811.12088 (2018)*, 2018.

[102] A. Sengupta, M. Nabeel, M. Ashraf, and O. Sinanoglu, "Customized locking of ip blocks on a multi-million-gate soc," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '18, 2018.

[103] M. El Massad, S. Garg, and M. Tripunitara, "The sat attack on ic camouflaging: Impact and potential countermeasures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2019.

[104] H. Zhou, "A humble theory and application for logic encryption," *Cryptology ePrint Archive, Report 2017/696.(2017)*, 2017. [Online]. Available: https://eprint.iacr.org/2017/696.pdf

[105] Shamsi, Kaveh and Li, Meng and Meade, Travis and Zhao, Zheng and Pan, David Z. and Jin, Yier, "Cyclic obfuscation for creating sat-unresolvable circuits," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, ser. GLSVLSI '17. New York, NY, USA: ACM, 2017, pp. 173–178. [Online]. Available: http://doi.acm.org/10.1145/3060403.3060458

[106] Hai Zhou, Ruifeng Jiang, and Shuyu Kong, "Cycsat: Sat-based attack on cyclic logic encryptions," in *ICCAD: Proceedings of the International Conference on Computer-Aided Design*. ACM, 2017.

[107] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against ic counterfeiting and overproduction," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2017, pp. 1–6.

[108] A. Chakraborty, Y. Liu, and A. Srivastava, "Timingsat: Timing profile embedded sat attack," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2018, pp. 1–6.

[109] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "Smt attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the sat attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 1, pp. 97–122, Nov. 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7335

[110] K. Zamiri Azar, H. Mardani Kamali, H. Homayoun, and A. Sasan, "Threats on logic locking: A decade later," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '19. New York, NY, USA: ACM, 2019, pp. 471–476. [Online]. Available: http://doi.acm.org/10.1145/3299874.3319495

[111] 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, Oct 2017.

[112] H. Wang, Q. S. Shi, A. Nahiyan, D. Forte, and M. M. Tehranipoor, "A physical design flow against front-side probing attacks by internal shielding," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, 2020.

[113] H. Wang, Q. Shi, D. Forte, and M. M. Tehranipoor, "Probing assessment framework and evaluation of antiprobing solutions," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 6, pp. 1239–1252, June 2019.

[114] S. Skorobogatov and C. Woods, "In the blink of an eye: There goes your aes key," *in IACR Crypt. ePrint Arch., no. 296*, 2012.

[115] F. Courbon, S. Skorobogatov, and Ch. Woods, “Direct charge measurement in floating gate transistors of flash eeprom using scanning electron microscopy,” *Proceedings of the 32nd International Symposium for Testing and Failure Analysis*, 2016.

[116] Swarup Bhunia and Mark Tehranipoo, “Physical attacks and countermeasures,” *Elsevier Inc, Chapter 10 of book "Hardware Security: A Hands-On Learning Approach"*, pp. 246–282, 2019.

[117] D. Hisamoto *et al.*, “Finfet-a self-aligned double-gate mosfet scalable to 20 nm,” *IEEE Transactions on Electron Devices*, vol. 47, no. 12, pp. 2320–2325, Dec 2000.

[118] P. Zhao, R. M. Feenstra, G. Gu, and D. Jena, “Symfet: A proposed symmetric graphene tunneling field-effect transistor,” *IEEE Transactions on Electron Devices*, vol. 60, no. 3, pp. 951–957, March 2013.

[119] K. Roy, M. Sharad, D. Fan, and K. Yogendra, “Computing with spin-transfer-torque devices: Prospects and perspectives,” in *2014 IEEE Computer Society Annual Symposium on VLSI*, July 2014, pp. 398–402.

[120] Z. Chen, H. Zhou, and J. Gu, “R-accelerator: An rram-based cgra accelerator with logic contraction,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2655–2667, Nov 2019.

[121] Z. Chen and H. Zhou and J. Gu, “A rram-based coarse grain reconfigurable array for neural network accelerators,” in *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, Oct 2018, pp. 1–2.

[122] Z. Chen, H. Zhou, and J. Gu, “R-accelerator: A reconfigurable accelerator with rram based logic contraction and resource optimization for application specific computing,” in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, Oct 2018, pp. 163–170.

[123] X. Fong, Y. Kim, K. Yogendra, D. Fan, A. Sengupta, A. Raghunathan, and K. Roy, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.

[124] S. Angizi, Z. He, and D. Fan, "Parapim: A parallel processing-in-memory accelerator for binary-weight deep neural networks," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 127–132. [Online]. Available: https://doi.org/10.1145/3287624.3287644

[125] S. Angizi, Z. He, A. S. Rakin, and D. Fan, "Cmp-pim: An energy-efficient comparator-based processing-in-memory neural network accelerator," in *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3195970.3196009

[126] A. C. Shaahin Angizi, Zhezhi He and D. Fan, "Hybrid spin-cmos polymorphic logic gate with application in in-memory computing," in *IEEE Transactions on Magnetics (TMAG)*, 2019.

[127] A. Jaiswal *et al.*, "Energy-efficient memory using magneto-electric switching of ferromagnets," *IEEE Magnetics Letters*, vol. 8, 2017.

[128] Y. Wang, L. Ni, C. H. Chang, and H. Yu, "Dw-aes: A domain-wall nanowire-based aes for high throughput and energy-efficient data encryption in non-volatile memory," *IEEE Transactions on Information Forensics and Security*, 2016.

[129] S. A. A. S. R. Zhezhi He, Li Yang and D. Fan, "Sparse bd-net: A multiplication-less dnn with sparse binarized depth-wise separable convolution," in *ACM Journal on Emerging Technologies in Computing Systems*, 2019.

[130] S. Angizi, Z. He, A. Awad, and D. Fan, "Mrima: An mram-based in-memory accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2019.

[131] F. Parveen, S. Angizi, and D. Fan, "Imflexcom: Energy efficient in-memory flexible computing using dual-mode sot-mram," *J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 3, Oct. 2018. [Online]. Available: https://doi.org/10.1145/3223047

[132] M. Khan *et al.*, "Side-channel attack on sttram based cache for cryptographic application," in *IEEE International Conference on Computer Design (ICCD)*, 2017.

[133] H. Dery *et al.*, "Spin-based logic in semiconductors for reconfigurable large-scale circuits," *Nature*, vol. 447, pp. 573–376, 2007.

[134] B. Behin-Aein *et al.*, "Proposal for an all-spin logic device with built-in memory," *Nature*, vol. 5, pp. 266–270, 2010.

[135] Colli, A. and Pisana, S. and Fasoli, A. and Robertson, J. and Ferrari, A. C., "Electronic transport in ambipolar silicon nanowires," *physica status solidi (b)*, vol. 244, no. 11, pp. 4161–4164, 2007.

[136] A. K. Geim and K. S. Novoselov, "The rise of graphene," *Nature Materials*, vol. 6, pp. 183–191, 2007.

[137] R. Martel, V. Derycke, and C. Lavoie, J. Appenzeller, K. K. Chan, J. Tersoff, and Ph. Avouris, "Ambipolar electrical transport in semiconducting single-wall carbon nanotubes," *Phys. Rev. Lett.*, vol. 87, 2001.

[138] Appenzeller, J. and Knoch, J. and Tutuc, E. and Reuter, M. and Guha, S., "Dual-gate silicon nanowire transistors with nickel silicide contacts," in *Electron Devices Meeting, 2006. IEDM '06. International*, 2006, pp. 1–4.

[139] M. De Marchi, D. Sacchetto, S. Frache, J. Zhang, P. Gaillardon, Y. Leblebici, G. De Micheli, "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire fets," in *Electron Devices Meeting (IEDM), 2012 IEEE International*, Dec 2012, pp. 8.4.1–8.4.4.

[140] P.-E. Gaillardon and S. Bobba and M. De Marchi and D. Sacchetto and G. De Micheli, "Nanowire systems: Technology and design," *Philosophical Transactions of the Royal Society of London A*, vol. 372, no. 2012, 2014.

[141] A. Stoica, R.S. Zebulum, D. Keymeulen, M.I. Ferguson, and V. Duong , "Taking evolutionary circuit design from experimentation to implementation: some useful techniques and a silicon demonstration," *IEE Proceedings-Computers and Digital Techniques*, vol. 151, no. 4, pp. 295–300, 2004.

[142] A. Stoica, R.S. Zebulum, and D. Keymeulen, *Polymorphic Electronics*. Springer, 2001.

[143] R. Ruzicka, "New polymorphic nand/xor gate," in *Proceedings of 7th WSEAS International Conference on Applied Computer Science*, vol. 2007. Citeseer, 2007, pp. 192–196.

[144] V. Garg and V. Arunachalam, "Architectural analysis of rsa crypto system on fpga," *International Journal of Computer Applications*, vol. 26, no. 8, 2011.

[145] D. E. Holcomb and W. P. Burleson and K. Fu, "Power-up sram state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, Sept 2009.

[146] M. Mustapa and M. Niamat, "Temperature, voltage, and aging effects in ring oscillator physical unclonable function," in *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*, Aug 2015, pp. 1699–1702.

[147] Qutaiba Alasad, Yu Bi, and Jiann Yuan, "E2lemi:energy-efficient logic encryption using multiplexer insertion," *Electronics*, vol. 6, no. 1, 2017. [Online]. Available: http://www.mdpi.com/2079-9292/6/1/16

[148] E. Erkek and T. Tuncer, "The implementation of asg and sg random number generators," in *System Science and Engineering (ICSSE), 2013 International Conference on*, July 2013, pp. 363–367.

[149] Hathwalia, Shruti and Yadav, Meenakshi, "Design and analysis of a 32 bit linear feedback shift register using vhdl," in *International Journal of Engineering Research and Applications*, vol. 4, 2014, pp. 99–102.

[150] P. Alfke, "Efficient shift registers, lfsr counters, and long pseudo-random sequence generators," in *Technical Report, Xilinx, Inc. Application number 052*, July July 7, 1996, [Online]. Available: http://www.xilinx.com/support/documentation/application_notes/xapp052.pdf.

[151] J. Dubeuf and D. Hély and R. Karri , "Run-time detection of hardware trojans: The processor protection unit," in *2013 18th IEEE European Test Symposium (ETS)*, May 2013, pp. 1–6.

[152] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou, "Cyclic locking and memristor-based obfuscation against cycsat and inside foundry attacks," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 85–90.

[153] A. Rezaei, J. Gu, and H. Zhou, "Hybrid memristor-cmos obfuscation against untrusted foundries," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2019, pp. 535–540.

[154] K. Huang and R. Zhao, "Magnetic domain-wall racetrack memory-based nonvolatile logic for low-power computing and fast run-time-reconfiguration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 9, pp. 2861–2872, Sep. 2016.

[155] F. Parveen, Z. He, S. Angizi, and D. Fan, "Hybrid polymorphic logic gate with 5-terminal magnetic domain wall motion device," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2017, pp. 152–157.

[156] Y. Zhang, B. Yan, W. Wu, H. Li, and Y. Chen, "Giant spin hall effect (gshe) logic design for low power application," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2015, pp. 1000–1005.

[157] S. Patnaik, N. Rangarajan, J. Knechtel, O. Sinanoglu, and S. Rakheja, "Advancing hardware security using polymorphic and stochastic spin-hall effect devices," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018.

[158] A. Roohi, R. Zand, and R. F. DeMara, "Logic-encrypted synthesis for energy-harvesting-powered spintronic-embedded datapath design," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '18, 2018.

[159] S. Angizi, H. Jiang, R. F. DeMara, J. Han, and D. Fan, "Majority-based spin-cmos primitives for approximate computing," *IEEE Transactions on Nanotechnology*, vol. 17, no. 4, pp. 795–806, July 2018.

[160] Q. Alasad *et al.*, "Resilient aes against side-channel attack using all-spin logic," in *Proceedings of the on Great Lakes Symposium on VLSI 2018*, ser. GLSVLSI '18, 2018.

[161] Hansen, Mark C. and Yalcin, Hakan and Hayes, John P., "Unveiling the iscas-85 benchmarks: A case study in reverse engineering," *IEEE Des. Test*, vol. 16, no. 3, pp. 72–80, Jul. 1999. [Online]. Available: http://dx.doi.org/10.1109/54.785838

[162] M. Yasin and J. Rajendran and O. Sinanoglu and R. Karri, "On improving the security of logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2015.

[163] M. T. Rahman and D. Forte and Q. Shi and G. K. Contreras and M. Tehranipoor, "Csst: Preventing distribution of unlicensed and rejected ics by untrusted foundry and assembly," in *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct 2014, pp. 46–51.

[164] S. Dasgupta and A. Rajashekhar and K. Majumdar and N. Agrawal and A. Razavieh and S. Trolier-Mckinstry and S. Datta, "Sub-kt/q switching in strong inversion in pbzr0.52ti0.48o3 gated negative capacitance fets," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 1, pp. 43–48, Dec 2015.

[165] M. D. Marchi, D. Sacchetto, J. Zhang, S. Frache, P. E. Gaillardon, Y. Leblebici, and G. D. Micheli, "Top-down fabrication of gate-all-around vertically stacked silicon nanowire fets with controllable polarity," *IEEE Transactions on Nanotechnology*, vol. 13, no. 6, pp. 1029–1038, Nov 2014.

[166] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Circuits and Systems, 1989., IEEE International Symposium on*, May 1989, pp. 1929–1934 vol.3.

[167] F. Corno, M. Reorda, and G. Squillero, "Rt-level itc'99 benchmarks and first atpg results," *Design Test of Computers, IEEE*, 2000.

[168] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool," in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds., 2010.

[169] J. J. Paul Kocher and B. Jun, "Introduction to differential power analysis and related attacks, 1998," in *Available at http://www.cryptography.com/dpa/technical*, 1998.

[170] S. Tajik, H. Lohrke, J.-P. Seifert, and C. Boit, "On the power of optical contactless probing: Attacking bitstream encryption of fpgas," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[171] Y. Bi, K. Shamsi, J. Yuan, F. Standaert, and Y. Jin, "Leverage emerging technologies for dpa-resilient block cipher design," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 1538–1543.

[172] M. W. Allam and M. I. Elmasry, "Dynamic current mode logic (dycml): a new low-power high-performance logic style," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 3, pp. 550–558, March 2001.

[173] C. Augustine *et al.*, "Low-power functionality enhanced computation architecture using spin-based devices," in *2011 IEEE/ACM International Symposium on Nanoscale Architectures*, 2011.

[174] K. Y. Camsari, S. Ganguly, and S. Datta, "Modular approach to spintronics," in *2015 Scientific Reports*, vol. 5, 2015.

[175] Z. Pajouhi *et al.*, "Exploring spin-transfer-torque devices for logic applications," *IEEE Transactions on CAD*, 2015.

[176] M. Sharad, K. Yogendra, K.-W. Kwon, and K. Roy, "Design of ultra high density and low power computational blocks using nano-magnets," in *International Symposium on Quality Electronic Design (ISQED)*, 2013.

[177] J. M. Renders *et al.*, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways," in *IEEE Proceedings on EC*, 1994.

[178] S. E. Tavares and H. M. Heys, "Avalanche characteristics of substitution-permutation encryption networks," *IEEE Trans. Comput.*, 1995.

[179] M. G. Mankalale and S. S. Sapatnekar, "Optimized standard cells for all-spin logic," *J. Emerg. Technol. Comput. Syst.*, 2016.

[180] Q. Tian and S. A. Huss, "Power amount analysis: Another way to understand power traces in side channel attacks," in *2nd ICDIPC*, 2012.

[181] S. Mathew, S. Satpathy, V. Suresh, H. Kaul, M. Anders, G. Chen, A. Agarwal, S. Hsu, and R. Krishnamurthy, "340mv–1.1v, 289gbps/w, 2090-gate nanoaes hardware accelerator with area-optimized encrypt/decrypt gf(24)2 polynomials in 22nm tri-gate cmos," in *2014 Symposium on VLSI Circuits Digest of Technical Papers*, June 2014, pp. 1–2.

[182] P. Subramanyan and N. Tsiskaridze and W. Li and A. Gascón and W. Y. Tan and A. Tiwari and N. Shankar and S. A. Seshia and S. Malik, "Reverse engineering digital circuits using structural and functional analyses," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 1, pp. 63–80, 2014.

[183] B. Shakya, N. Asadizanjani, D. Forte, and M. Tehranipoor, "Chip editor: Leveraging circuit edit for logic obfuscation and trusted fabrication," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2016, pp. 1–8.