Re-imagining the Security of the Hardware using Lookup Table based Obfuscation

By

GAURAV KOLHE

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical and Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Houman Homayoun, Chair

Bevan Bass

Vivek Menon

Committee in Charge

2022

Abstract

The increasing cost of manufacturing the complex Integrated Circuits (ICs) and ever-rising competition to shorten time-to-market have given rise to the trend of fabless manufacturing. Moreover, the addition of various players in the product manufacturing lifecycle has endangered the security of Intellectual Properties (IPs). "Logic locking" and "IC camouflaging" are amongst the most prevalent protection schemes that can thwart various hardware security threats. However, the state-of-the-art attacks such as Boolean Satisfiability Attack (SAT-attack) and approximation-based attacks, question the efficacy of the existing defense schemes.

Recent solutions to protect hardware designs from various hardware security threats have employed a myriad of obfuscation techniques. However, these solutions have mostly focused on specific design elements such as "SAT-hardness". Despite meeting the focused criterion such as "SAT-hardness" for maximizing security, obfuscated designs are still vulnerable to the newly evolving attack vectors. To mitigate this problem and provide a better solution for thwarting SAT-attack and evolving attack vectors, Look-Up Table (LUT)-based obfuscation is studied. This work provides an extensive analysis of LUT-based obfuscation by exploring several factors such as LUT technology, size, number of LUTs, and replacement strategy as they have a substantial influence on the Power-Performance-Area (PPA) and security of the design.

For making the reconfigurable logic obfuscation efficient in terms of design overheads, this work further proposes a novel architecture using LUT. Additionally, a study is conducted with different threat models and attack vectors to show that the security provided by the proposed primitive is superior to that of the traditional ways of LUT-based obfuscation.

While many existing works have focused on mitigating the well-known SAT-attack and its derivatives, there hasn't been much research on preventing Power Side-Channel Attacks (P-SCAs), which have the capability to retrieve the sensitive contents of the IP in a non-invasive manner. Using P-SCA for unlocking the obfuscated circuit, does not require the laborious task of simulating powerful SAT attacks. For maximizing the security, and curbing P-SCA, the work proposes a "defense-in-depth obfuscation" which builds on our existing LUT-based solution. The proposed obfuscation is tailored such that LUT-based obfuscation incurs minimal overheads while providing a full-scale robust solution to secure the hardware.

Additionally, this work invents a security-driven design flow, which uses off-the-shelf industrial Electronic Design Automation (EDA) tools for easier obfuscation of the design. This proposed flow is meticulously crafted in a way such that it is non-disruptive to the current industrial physical design flow. To enable this flow one must overcome some obstacles; as there is a lack of frameworks and unified methods that can validate the security and functionality of obfuscated designs. This work has discussed the challenges of validating the security and functionality of obfuscated designs and further presents the methodology for verifying the functionality of the LUT-based obfuscated IP (pre and post-fabrication). Additionally, Security Evaluation Platform for Hardware Logic Obfuscation using Intelligent Artificial Neural Net (SEPIANN) is proposed to validate the security of the design. This system-level framework instantaneously estimates the obfuscation strength in terms of attack resiliency time eliminating the need to simulate de-obfuscation using the SAT-attack.

Finally, the work shows the application and scalability of the proposed work by fabricating various security and computing cores by obfuscating them with the LUT-based obfuscation.

Acknowledgment

With a heart full of gratitude I would like to thank my advisor, Professor Houman Homayoun, who supported me throughout the entire journey of my Ph.D. , He provided me the opportunity to conduct groundbreaking research and helped me unlock my true potential. His insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I want to extend my gratitude to Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory (AFRL), and the National Science Foundation Center for Hardware and Embedded Systems Security and Trust (CHEST) as the various parts of the research were primarily funded by them.

Moreover, I would also like to acknowledge my mentor, Dr. Vivek Menon, from the Department of Defense, and Dr. Sai Manoj from George Mason University for always guiding me and I would also like to thank Prof. Bevan Baas, Prof. Setareh Rafatirad, and Prof. Soheil Ghiasi who served on my dissertation and qualification exam committee. Your valuable comments and feedback were very significant.

In addition, I would like to express my gratitude to my loving parents for their unconditional trust, timely encouragement, and emotional and financial support. It was their love that raised me when I got weary. I am grateful to my sister Ankita. You're a beautiful sister who has always shown deep and unwavering care for me all of my life.

"Where we love is home, home that our feet may leave, but not our hearts."

I could not have completed this dissertation without the support of my friends, who provided stimulating discussions and happy distractions to rest my mind outside of my research. You made me feel at home while being away from home.

Finally, I am also thankful to the University of California, George Mason University, and all of its wonderful staff who helped me with various aspects of clerical work.

Contents

List of Tables

List of Figures

Chapter 1

Specific Aims of the Research

1.1. Aim I - Breaking the trade-off between design overheads and security for LUT-based obfuscation.

Maximizing profits while minimizing the risk in a technologically advanced silicon industry has motivated the globalization of the fabrication and electronic hardware supply chains. However, with the increasing magnitude of successful hardware attacks, the security of hardware IPs has been compromised. Many existing security works have focused on resolving a single vulnerability such as SAT-attack while neglecting the possibility other threats. The lack of the obfuscation method that can provide security against various threat has motivated us to propose a novel approach for securing hardware IPs during the fabrication process and supply chain via logic obfuscation by utilizing LUT-based obfuscation. The first part of this research aims to study and deploy the LUT-based obfuscation that protects against most of the hardware security attacks while breaking the trade-off between security and imposed overheads. Moreover, the work establishes a standard security-driven flow for deploying LUT-based obfuscation, which is non-disruptive to standard Application-Specific Integrated Circuit (ASIC)-design flow.

1.2. Aim II - Providing multi-layer defense mechanism.

While LUT-based obfuscation can thwart most attacks, a successful security scheme usually uses a defense-in-depth mechanism. With the constant evaluation of the attack vectors, reliance on single security primitive is not recommended, and thus in this part, the aim is to realize the "defense-

in-depth" obfuscation using LUT. Moreover, the logic obfuscation community uses a threat model, which assumes the key for restoring the functionality of the obfuscated IP is stored in the tamper-proof memory, where an attacker can't access its content. However, in reality, due to the surge of P-SCA, it is possible to retrieve the content of the tamperproof memory without simulating SAT-attack, and thus, the work focuses on making the obfuscation resilient to P-SCA. The work proposes a multi-layer defense mechanism called Deep-Learning Power Side-Channel Attack Mitigation using Emerging Reconfigurable Devices and Logic Locking (LOCK & ROLL). LOCK & ROLL utilizes Magnetoresistive Random-Access Memory (MRAM)-based LUT called Symmetric Magnetoresistive Random-Access Memory-based Look-up Table (SyM-LUT) with Scan-Enable Obfuscation Mechanism (SOM), which is the key element that provides "defense-in-depth" solutions for restoring the trust in Silicon.

1.3. Aim III - Validating the security of an obfuscated IP.

Finally, numerous defense mechanisms exist for an IP owner who wants to protect his IP against various hardware security threats. As the defense mechanisms' success depends on the underlying characteristics of the IP being secured, evaluating and validating the security of the IP after the integration of obfuscation logic against various attacks is a highly crucial step. However, the security evaluation against SAT-attack can take a few days to months using a contemporary method of simulating the attack. A mechanism that can instantaneously assess the strength of different obfuscation schemes against SAT-attack is the need of the hour. This work proposes the methodology for validating the security and the functionality of the LUT-based Logic-Locked IP Cores.

Outline

The research proposes a full-fledged solution to mitigate hardware security threats. The following chapter introduces the concept of hardware obfuscation, security, and attack vectors/threats. Lookup table-based obfuscation and its empirical evaluation against SAT-attack is covered in Chapter 4. Further, this section introduces a novel architecture that breaks the trade-off in chapter 5 to make LUT-based obfuscation a more viable means of obfuscation. To render maximum security against various threat vectors, chapter 6 uses LUT-based obfuscation to deploy a defense-in-depth

solution in the hardware security domain. Finally, Chapter 7 discusses the methodology to verify the security of the obfuscation instantaneously using neural networks, and Chapter 8 introduces obfuscation flow and the results from the fabricated version of the IPs obfuscated with LUT. The conclusion of the work is drawn in Chapter 9.

Chapter 2

Introduction

2.1. Hardware Security

With the rapid advancement in semiconductor technology due to "technology scaling," the dimensions of the ICs have been reducing which has led to the increasing transistor density on a die. Today, the device integration density has continued to double every two years, following "Moore's Law." This trend has led to a rapid increase in the computational power of IC from its advent into manufacturing in the 1960s. However, this growth has not been cheap. The complexity in design increases the time-to-market, manufacturing costs, and recurring and non-recurring test costs at different levels of abstraction. The IP production house has to account for these by-products of Moore's Law to sustain itself in today's competing market. This need has led to heavy reliance on the:

1. Third-party fabrication services for cheaper manufacturing and testing
2. Third-party soft/hard IP core vendors to reduce design time, increase reliability with pre-verification, and hence time-to-market.

Today, the design, manufacturing, and assembly of modern ICs consisting of millions and billions of transistors with intricate fabrication processes are backed by a complex network of global suppliers. For example, one of the United States (US)-based semiconductor companies has $7,300$ suppliers spread over 46 states in the US, and more than $8,500$ suppliers that are geographically located outside of the US. [7]. The globalization of the fabrication process and hardware supply chain

benefits mainly from the diverse and varied skills of human resources and geographical advantages of suppliers.

In the fabless model, the IP is made available to Contract Manufacturers (CM) responsible for mass production. The ICs are then tested, with the good ones being sent to the next stage of the supply chain, whereas the rest is either recycled or dumped as an e-waste. The next phase of the supply chain consists of the Authorized Sellers/Distributors supplying the semiconductor chips to Original Equipment Manufacturers (OEM).

Consumers heavily rely on the company's reputation for providing the end product, which depends on the Original Chip Manufacturer (OCM) reputation. The end applications can range from simple day-to-day hand-held gadgets to critical ones potentially dealing with health, safety, and security. Any vulnerability introduced in the IC supply chain ripples back and forth through the chain regarding economic damages, reputation, and consumer health and safety.

Monitoring the flow of the final products becomes a daunting, impossible task once the IP leaves the design house through brokers, sub-contractors, OEM, and their CM. In recent years, thousands of independent distributors and brokers have set up shop outside the traditional supply chain consisting of simply OCMs and their Authorized Distributors with the help of e-commerce. Consumers are offered products with better availability and possibly lower costs, so they often fall prey to spurious brokers. One of the reasons for such spurious products being present in the market is e-waste recycling. Environmental awareness has resulted in electronics waste (e-waste) no longer ending up in landfills; instead, they have been known to be recycled by removing electronic components from scrap circuit boards sent for "recycling." These used components can then be refurbished and remarked to indicate new and/or higher-performing parts. Some component buyers might also end up buying out-of-production chips with lower performance. The OCM, OEM, and CM remain oblivious to these forged products in the supply chain until/unless they register huge failure rates or warranty claims. At this point, the damages can be irreparable.

An inference to the above discussion is with the dissolution of control over the IC supply chain where most of the security challenges are introduced, such as overproduction, trojan insertion, Reverse Engineering (RE), IP theft, and counterfeiting [8] to name a few. Figure 2.1 shows the various hardware security threats during multiple stages of IP manufacturing.

To curb various threats involved during IP manufacturing, Hardware Obfuscation was intro-

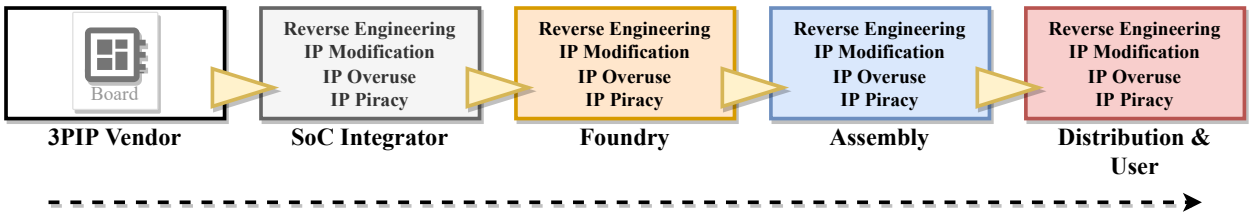


Figure 2.1. Various hardware security threats involved in Fabless manufacturing process.

duced. **Hardware Obfuscation** is a technique that makes understanding or RE of a design difficult. To protect hardware IP from these threats, the design must be unintelligible, even in decrypted form. Hardware obfuscation provides the option to effectively hide and disable the design while facilitating structural testing and static/dynamic parameter analysis during manufacturing. This convenience makes obfuscation a desirable method for security and an active field of research.

Numerous obfuscation techniques, namely logic locking and camouflaging, have been proposed to thwart existing hardware security threats. Increased interest within the logic locking and IC camouflaging research community has persuaded Mentor Graphics, a major Computer-aided design (CAD) tool provider, to release TrustChain, which is a CAD framework that supports logic locking and camouflaging as a means of curbing various hardware security threats [9]. However, recently introduced attacks have exploited vulnerabilities in various available logic obfuscation schemes. SAT-attack is one of the most effective de-obfuscation/de-camouflaging techniques. Such attacks vectors can RE a target design even when state-of-the-art logic locking and camouflaging protection mechanisms [10, 11] are used.

For the working of SAT-attack, it requires a reverse-engineered netlist along with an oracle copy. An oracle copy is the activated, working copy of the obfuscated design. Although IC RE for netlist extraction is a slow and expensive process, it has become more practical today with the advent of advanced imaging and probing techniques such as focused ion beam (FIB) and scanning electron microscopy (SEM). To RE the design, an attacker needs to perform delayering, high-resolution imaging or X-raying, and image processing to retrieve the netlist from a fabricated IC. If the adversary is a foreign government or competitive ill-intended organization, acquiring this expensive imaging equipment is possible. Therefore, sensitive designs, like military-grade ICs, need to be kept secure from such threats.

To curb the SAT-attack, recent works in the domain of reconfigurable obfuscation make use of

diverse approaches like increasing the number of reconfigurable blocks [12] to reinforce the security against the SAT-attack. However, on closer study, it is observed that security can be compromised despite considering all substantial and compelling factors for reconfigurable logic obfuscation.

Many studies try to mitigate SAT-attack using different techniques/methods. Yet, the red-teaming efforts in the hardware security community have continued to find the vulnerability in the proposed techniques. Some techniques are impractical due to their overheads, while others are vulnerable to evolving attacks in the community. This study aims to develop a full-scale solution that can prevent existing attacks while providing maximum security in case of zero-day attacks. Moreover, this work discusses the methods to verify functionality and security in case of obfuscated design.

The work discusses the background, basic terminology in the hardware security community, and the current problem in the following chapter.

Chapter 3

Background

3.1. Logic Locking

Logic Locking is a type of hardware obfuscation that aims to conceal the functionality of the design by inserting additional logic gates. These gates can be key-programmable (KPG) XOR/XNOR gates or KPG MUXes for interconnection. The ambiguity is created in the circuit due to the newly added gates. The programmable key for restoring the original functionality of the IP is stored in the tamper-proof memory. The strength of the logic obfuscation depends on the number of gates inserted, and the location of the gate insertion [13–15]. Logic obfuscation is weak when the inserted key gates are isolated, or their effect can be muted. If mutable gates are employed, the attacker can determine the key bits within a second. However, it can be strengthened by inserting key gates such that their effects are not mutable. One of the logic locking techniques, Strong Logic Locking, insert gates at the position where input cannot be easily sensitized to output. Figure 3.1 shows Strong Logic Locking.

Another obfuscation technique thwarts the RE attempt using camouflaging, which doesn't require extra KPG gates. This technique blocks the RE attempt by replacing existing gates in the design with camouflaged gates. In one embodiment of IC camouflaging, the layouts of logic gates are designed to look identical, resulting in an incorrect extraction. For example, in Figure 3.2 Standard Cell Versus Camouflaged Cell, the layout of regular NAND cell and NOR cell look different and are hence easy to RE. However, the layout of camouflaged NAND cell and NOR cell look identical and difficult to differentiate. When deceived into incorrectly interpreting the functionality of the

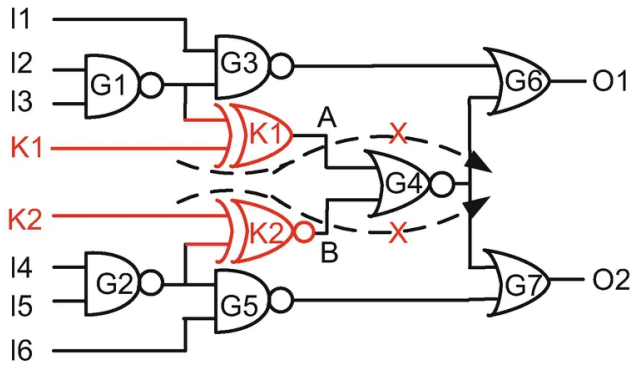


Figure 3.1. Strong Logic Locking [2]. The attacker can not sensitize either K1 or K2 to a primary output O1 or O2.

camouflaged gate, the attacker may obtain a reverse-engineered netlist that is different from the original.

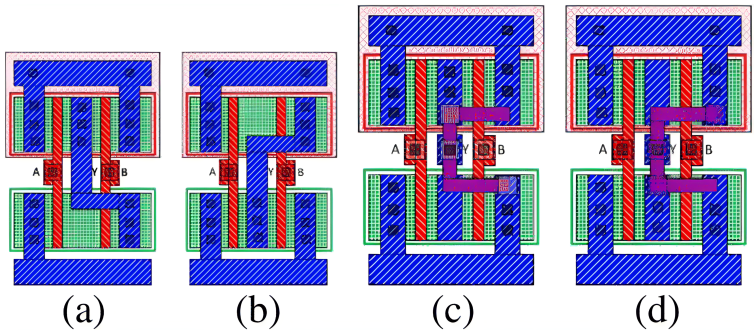


Figure 3.2. Standard Cell Versus Camouflaged Cell [3] (a) NAND Cell (b) NOR Cell (c) Camouflaged NAND Cell (d) Camouflaged NOR Cell

After proposing various techniques for hardware security, attacks such as justification or sensitization [16] were developed to RE the design using heuristic techniques; however, with more advanced logic obfuscation, the community has shown success in extracting the keys using Automatic Test Generation Pattern (ATPG) and SAT-attack [11]. The novel SAT-attack is discussed in the following section.

3.2. Overview of the SAT-Attack

Boolean Satisfiability Attack (SAT-attack) [11] is used for determining the correct key of the obfuscated or logic-locked circuit. The SAT-attack is an "oracle-guided attack" where the threat model assumes that an attacker has access to the activated IC and the locked netlist, which can be retrieved through invasive RE [17]. The SAT solver input is a Boolean formula in conjunctive normal

form (CNF) obtained from the transformation of the obfuscated netlist. The CNF representations of various logic gates are represented in Table 3.1.

Table 3.1. Tseytin Transformation of Basic Logic Gates [1]

Gate	Operation	CNF (sub-expression)
$C = \text{AND}(A,B)$	$C = A.B$	$(\overline{A} \vee \overline{B} \vee C) \wedge (A \vee \overline{C}) \wedge (B \vee \overline{C})$
$C = \text{NAND}(A,B)$	$C = \overline{A.B}$	$(\overline{A} \vee \overline{B} \vee \overline{C}) \wedge (A \vee C) \wedge (B \vee C)$
$C = \text{OR}(A,B)$	$C = A + B$	$(A \vee B \vee \overline{C}) \wedge (\overline{A} \vee C) \wedge (\overline{B} \vee C)$
$C = \text{NOR}(A,B)$	$C = \overline{A + B}$	$(A \vee B \vee C) \wedge (\overline{A} \vee \overline{C}) \wedge (\overline{B} \vee \overline{C})$
$C = \text{BUFF}(A,B)$	$C = A$	$(A \vee \overline{C}) \wedge (\overline{A} \vee C)$
$C = \text{NOT}(A,B)$	$C = \overline{A}$	$(\overline{A} \vee \overline{C}) \wedge (A \vee C)$
$C = \text{XOR}(A,B)$	$C = A \oplus B$	$(\overline{A} \vee \overline{B} \vee \overline{C}) \wedge (A \vee B \vee \overline{C}) \wedge (A \vee \overline{B} \vee C) \wedge (\overline{A} \vee B \vee C)$
$C = \text{XNOR}(A,B)$	$C = \overline{A \oplus B}$	$(\overline{A} \vee \overline{B} \vee C) \wedge (A \vee B \vee C) \wedge (A \vee \overline{B} \vee \overline{C}) \wedge (\overline{A} \vee B \vee \overline{C})$
$C = \text{MUX}(S,A,B)$	$C = A.\overline{S} + B.S$	$(S \vee \overline{A} \vee C) \wedge (S \vee A \vee \overline{C}) \wedge (\overline{S} \vee \overline{B} \vee C) \wedge (\overline{S} \vee B \vee \overline{C})$

SAT attack iteratively eliminates the incorrect keys based on specific input patterns, called Distinguishing Inputs (DIP) [11]. DIPs are found using a miter circuit, which was initially used to check the two hardware designs' equivalence. The miter circuit is built using two copies of the locked netlist obtained after invasive RE efforts, with their outputs XORed together. The input is common to both instances of the locked netlists, which are part of the miter circuit, but each locked netlist is programmed with a different key. The DIP is found when the output of the two circuits differs, making the XOR gate or miter circuit "1" output. These two different outputs indicate that one of the keys is wrong, and the input that helped us distinguish between the keys is termed distinguishing input. The DIP is applied to the oracle circuit to identify the correct output. The SAT-attack in every iteration tries to find a new DIP until no new DIP exists. This algorithm reduces the search space for finding the correct key with fewer queries (iteration) that need to be done with the oracle circuit. The following section discusses the process of SAT-attack in further details.

3.3. Working of the SAT-Attack

Every key input combination is considered a candidate key before invoking the SAT solver to find the key. Let's denote the Set of Candidate Keys (SCK). If one can find an input x_d, and two distinct key values K_1 and K_2 in SCK such that $C(x_d, K_1, Y_1) \neq C(x_d, K_2, Y_2)$, the input x_d would be a DIP [11]. This is because the selected input can prune the SCK and find at least one incorrect key

which can be removed from SCK. In addition, each time a new DIP is found, the SCK search space for function F_{DI} should be updated. This could be achieved by forcing the F_{DI} to check each pair of new keys, K_1 and K_2, against all DIPs. A Complete- DIP-set is a set of DIP inputs that reduce the SCK to the Set of Valid Keys (SVK). SCK reduces to SVK when one no longer can find a DIP using the updated F_{DI}. At this point, if a key is valid across the complete-DIP-set, it is the correct key for all other inputs [11].

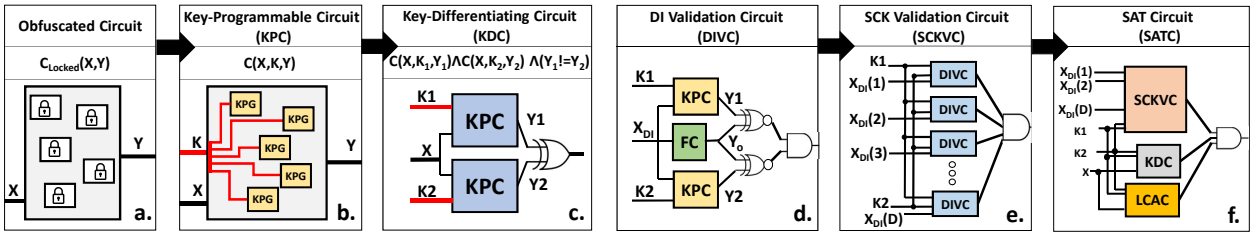


Figure 3.3. (a) Transforming an obfuscated circuit to (b) Key-Programmable Circuit and (c) Key Differentiating Circuit. (d) DIVC circuit for validating that two input keys produce the correct output with respect to a previously discovered DIP. (e) SCKVC circuit for validating that both input keys are in SCK set and produce the correct output for all previously discovered DIPs. (f) SAT circuit for finding a new DIP.

Figure 3.3 (b) shows a reverse-engineered netlist, where all obfuscated cells or camouflaged cells from Figure 3.3 (a) are replaced with KPG cells. KPC denotes this circuit). To build the F_{DIP}, two copies of the KPC are used, their non-key inputs (X) are tied together, and their outputs are XORed. This circuit produces logic 1 when the output of two instantiated KPCs for the same input X, but different keys K_1 and K_2 are different. As suggested in Figure 3.3 (c), this circuit is denoted as KDC or miter circuit.

The candidate keys in the SCK can produce the correct output for all DIPs that have previously been discovered and tested on the KPC circuit. To test the keys for one DIP, the circuit in Figure 3.3 (d) is instantiated. In this figure, FC is the working copy of the chip, and its output is used for testing the correctness of both KPCs for a given DIP and two key values. This circuit is denoted as Distinguishing Input Validation Circuit (DIVC). To test the keys for all DIPs, as illustrated in Figure 3.3 (e), the DIVC circuit is duplicated D times, with D being the number of current DIPs tested and the output of all DIVC circuits ANDed together. The resulting circuit is a validation circuit for SCK the set denoted as Validation Circuit for Set of Candidate Keys (SCKVC).

If two keys, K_1 and K_2, produce the correct output for all previously tested DIPs (SCKVC evaluates to true) but produce different results for a new input X_{test}, then X_{test} is a DIP that

further prunes the SCK. This, as illustrated in Figure 3.3 (f), could be tested by using an AND gate at the output of SCKVC and KDC circuits. The resulting circuit forms a SAT solvable circuit denoted by SATC. When SATC evaluates to true, the KDC has tested a pair of keys, K_1 and K_2, that produce two different results for an input X_{test}, and the SCKVC circuit has confirmed that both K_1 and K_2 belong to the SCK set. Hence, the input X_{test} is yet another DIP. Each time a new DIP is found, the SCKVC should be updated by adding another DIVC circuit to test the newly discovered DIP. This process is continued until SAT solver no longer finds a solution to the final SAT circuit. In this case, any key remaining in the SCK set is the correct key for the circuit. Every time the SAT solver is executed on the SAT solver side, it learns a new set of conflict clauses. It is essential to store the learned clauses and use them in the following invocation of the SAT solver to prevent SAT solver from re-learning these clauses. Hence, as illustrated in Figure 3.3 (f), a Learned-Clause Avoidance Circuit (LCAC) is added to the SATC to check for learned conflict clauses.

As illustrated in Algorithm 1, the SAT-attack follows the SATC construction process explained in Figure 3.3. The SCKVC circuit does not contain any logic in the first iteration since there is no previously tested DIP. Hence, it is set to 1 (true). The KDC circuit is built based on its definition using the equation in Figure 3.3.c. ANDing the KDC and SCKVC circuits builds the SATC circuit. SAT_F function is a call to SAT solver. Considering the to-be-assigned variables in the SATC circuit are X, K_1, and K_2, the SAT solvers return an assignment to these variables and a list of conflict clauses (CC) learned during SAT execution. SAT_F returns UNSAT if no such assignment exists. The while loop is controlled by the return status of the SAT solver. In every pass through the while loop, a new DIP is found. Hence, the SATC circuit should be modified (lines 7-10). The parts of the SATC circuit that are updated are the SCKVC and LCAC. After finding each DIP, an additional DIVC is added to SCKVC to validate the keys generated in the following invocation of SAT solver for the newly found DIP. In addition, the freshly learned CCs are added to LCAC. The C_F is a call to the functional circuit that returns the correct output for each newly found DIP. Finally, the SATC circuit is formulated at line 10 for the following invocation of SAT solver.

The while loop is executed until no other DIP is found. At this point, any key in the SCK set is the correct key. To obtain a correct key, the DIVC circuit is modified to take a single key denoted as KeyGenCircuit. Hence, KeyGenCircuit has input K, and its output is valid if K satisfies all previous

Algorithm 1: SAT-based Attack Algorithm [11]

SAT_Attack ($Circuit\ C_{obf}, Circuit\ C_{org}$);
Input : Obfuscated Circuit, Oracle Circuit
Output: Key to unlock the functionality of the obfuscated circuit
$i \leftarrow 0$;
$F_1 \leftarrow$ C(X, K_1, Y_1) $\wedge$ C(X, K_2, Y_2);
while SAT$(F_i \wedge (Y_1 \neq Y_2))$ **do**
    X_d[i] $\leftarrow$ sat_assignment ($F_i \wedge (Y_1 \neq Y_2)$);
    Y_d[i] $\leftarrow$ eval(X_d[i]);
    $F_{i+1} \leftarrow F_i \wedge$ C(X_d[i], K_1, Y_d[i]) $\wedge$ C(X_d[i], K_2, Y_d[i]);
    $i \leftarrow i+1$;
end
$Correct\_Key \leftarrow$ sat_assignment($F_i \mid K_1$);

constraints imposed by previously found DIPs. A simple call to a SAT solver at this point returns a correct key assignment. If the SAT solver does not produce a valid key, the obfuscation, locking, or camouflaging technique is invalid. Note that the SAT-attack in each iteration, as explained in Algorithm 1 and illustrated in Figure 3.4, reduces the SCK by constraining the SATC with new clauses added to the SCKVC and LCAC. But it does not explicitly check to find the keys in SCK.

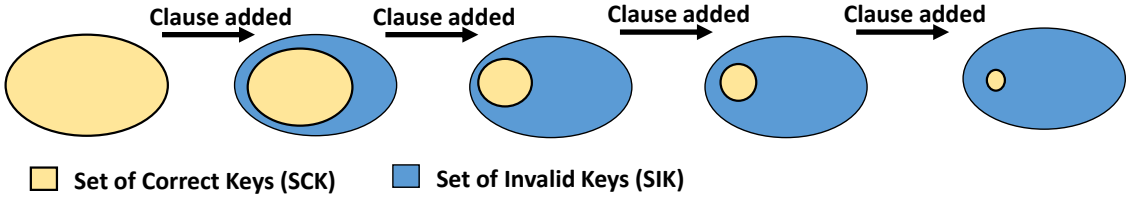


Figure 3.4. SCK set reduces in each pass through the while loop in Algorithm 1 as a new DI is discovered and is added to SATC circuit.

The discussion above shows how the logic locking obfuscation can be broken using SAT-attack. However, an additional pre-processing step is required for the camouflaged gate, where the reverse-engineered netlist with camouflaged gates is replaced with a MUX. The input to the MUX is the choice of gates that a camouflaged gate can implement, and the select input to MUX is treated as a key input. For example, to represent a camouflaged gate from Figure 3.2, one can use 2:1 MUX, with the input of MUX being NAND gate and NOR gate, and the select line can choose between the correct functionality implemented by the camouflaged gate.

3.4. Resisting SAT-Attack

Many works in the area of hardware security seek to increase the number of iterations required to retrieve the correct key for unlocking the circuit to thwart the SAT-attack. Stripped Functionality Logic Locking (SFLL) is the state-of-the-art method requiring exponential SAT iterations to find the correct key. However, the continued success of red-teaming efforts led by the hardware security community, work in [18], and [19] exploits the vulnerability in the SFLL method implementation and shows that the obfuscation key can be found within several minutes. Moreover, obfuscation similar to SFLL [20], such as SAR-Lock [21] and Anti-SAT [22], obfuscations fall into categories of one-point function, which evaluates to correct output upon applying a specific input pattern. The one-point obfuscation-based techniques require SAT-attack to apply many inputs to find the correct key. However, the output corruptibility offered by such techniques is very low, which means that the circuit works almost identical to the oracle circuit even when the wrong key is used [23].

In the quest to resist the SAT-attack, the community also formulated obfuscation techniques that resisted the obfuscated netlist's transformation to SAT problems. Delay-based locking or cyclic obfuscation are among a few of the examples that defy the SAT-attacks. However, shortly after introducing these obfuscation techniques, the attacks such as cycSAT and Satisfiability Module Theories (SMT) attacks were able to model the cyclic or behavioral locking into an SAT or SAT+theory solvable logic problems [24].

The solution to resist SAT-attack involves using reconfigurable blocks such as the LUT, where using large LUT sizes provides us with increased key size and number of functions that LUT can render. In LUT-based obfuscation, the combination of gates is replaced with LUT instead of adding key gates. The key in a LUT-based obfuscation is the memory element that configures/realizes the functionality of the LUT. The following section first illustrates the LUT-based obfuscation and its effect on security and design overheads.

Chapter 4

Lookup Table Obfuscation

This chapter expands upon the idea of LUT-based obfuscation and it is organized as follows: Section 4.1 describes the idea of leveraging LUT for securing IPs against SAT-attack. Section 4.2, Section 4.3 and Section 4.4 discusses the effect of various factors that can directly impact security and design overhead of an IP obfuscated using LUT-based obfuscation. As an outcome of the research covered in this chapter, this work has contributed to [4, 25, 26].

4.1. Lookup Table-based obfuscation

In obfuscation using LUT, the gates are selected from the design and are mapped to the LUTs. For example, to obfuscate a 2-input AND gate with LUT, one can replace the AND gate in the IP with the LUT whose configuration bits are set to {0,0,0,1} (Based on the truth table of AND gate). Obfuscation using LUT thus results in a netlist as a hybrid mixture of ASIC and programmable Field-Programmable Gate Array (FPGA) styles. In the LUT-based obfuscation, the keys that denote the logical function of the LUT can be stored in a tamper-proof (Secure) non-volatile memory. Without prior knowledge of the content of the non-volatile memory contents, the attacker doesn't have access to the IP's intended functionality and thus refrains the attacker from RE the IP. Figure 4.1 shows the LUT-based obfuscation, where part of the circuit is mapped to the LUT.

While storing the LUT configuration bits in tamperproof memory thwarts the attacker from understanding the content and functionality of the LUT, the attacker can still use SAT-attack to restore the content of the LUT. The attack on the LUT-based obfuscation using SAT-attack is

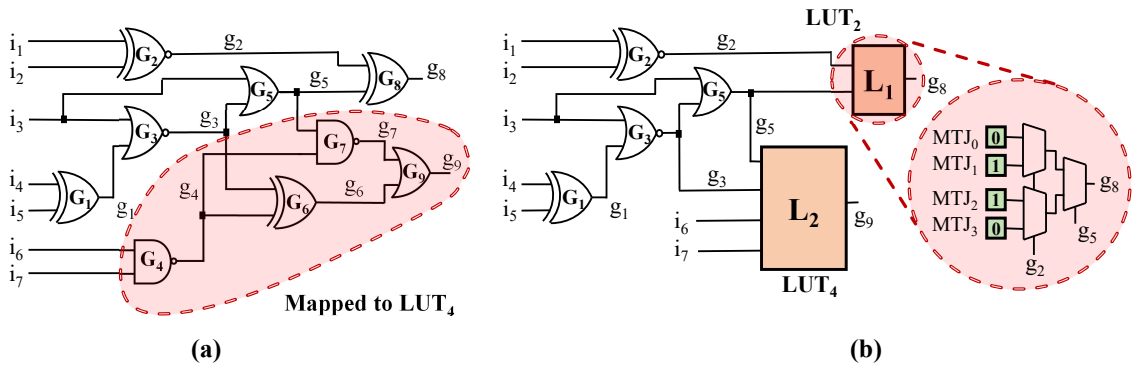


Figure 4.1. (a) Sample circuit used for obfuscation with gates G_4, G_6, G_7, G_9 selected for obfuscation (b) Obfuscation with LUT with representation of the LUT for SAT-attack simulation.

described here. The SAT-attack simulation does not offer the ability to model the LUT directly. Hence, LUT-based obfuscation is modeled using the MUXes, where each LUT is replaced with a (2+)-level MUX. Figure 4.1 (b) shows the representation of the LUT of size 2 for SAT simulation. The LUT of size 2 is built using 2:1 MUX. Since the attacker is interested in finding the LUT's configuration bits' content, which represents the functionality imparted by the LUT, the memory cells of the LUT are treated as the key inputs. The SAT-solver tries to find the value of these key inputs during the de-obfuscation process. After replacing all the LUT with their equivalent representation using MUXes, one can perform the SAT-attack.

Having shown the ways to attack LUT-based obfuscation using SAT-attack, one obvious way to hinder SAT-attack would be to use the larger sizes of the LUT in large quantities. However, it would also result in increased design overheads. Therefore, in LUT-based obfuscation, the question to address is (1) whether to use a few numbers of large LUTs *or* (2) use more numbers of small LUT sizes to defend against the SAT-attack. To answer this question, the work performs an extensive design-for-security space exploration for LUT-based obfuscation using four critical factors, namely (1) LUT technology, (2) LUT size, (3) number of LUTs, and (4) replacement strategy to find the impact of each on SAT-resiliency in the following section.

4.2. Impact of LUT Technology

This section discusses the design implementation of the MTJ-based LUT, followed by the impact of the SRAM and MTJ-based LUT on the design overheads.

4.2.1 Design and Integration of Spin Transfer Torque-based Look-up Table

As Spin Transfer Torque-based Look-up Table (STT-LUT) has shown higher efficiency in terms of design overheads [12], this work considers STT-LUT in this work. STT technology can provide incredible features like (1) higher integration density than Static Random Access Memory (SRAM), (2) high endurance and retention time, (3) near-zero leakage, and (4) soft error resilience [4]. Additionally, STT-LUT has shown to be highly integrative in the Complementary Metal-Oxide Semiconductor (CMOS) fabrication process [12]. Thus, STT-LUT, due to its virtue of on-die reconfigurability, enables us to achieve high performance and security against various hardware RE threats.

Additionally, for STT-LUT, reconfigurable bits can be stored in a Magnetic Tunnel Junction (MTJ) inserted between metal layers. MTJs are constructed of two ferromagnetic layers: a free layer and a fixed layer, and a thin oxide layer [6]. In the STT switching approach used in Spin Transfer Torque Magnetoresistive Random Access Memory (STT-MRAM), applying a bidirectional charge current through the terminals of the MTJ using a Metal-Oxide Semiconductor (MOS)-based circuit will result in the generation of a spin current. The spin current is used to change the magnetic polarity of the free layer to represent: 1) a high resistance or **Anti-Parallel** (AP) state and 2) a low resistance or **parallel** (P) state. The content stored in the MTJs is vulnerable to the de-layering process involved in the RE of the IP. The MTJ, in this manner, serves as the tamperproof memory to store the configuration of the LUT. The custom part of the design is implemented using the standard cell-based ASIC design flow. The resulting designs are static since the ASIC standard cells are implemented in the static logic style. This limits the LUT design to have a static type interface for connection with the static ASIC standard cells. Also, in the existing STT-LUT design styles, a dynamic circuit such as a dynamic sense amplifier resides in between the LUT's input and the output. However, this type of setting is not suitable for the obfuscation in static style [27]. In contrast, this work proposes STT-LUT with a design concept in which the path from the LUT inputs to the LUT output is a MUX, as shown in Figure 4.3 (a). The MUX of the LUT is a 2^n to 1 ($2^n : 1$) CMOS MUX implemented in static style, which can be written as a synthesizable Register-Transfer Level (RTL) code for automatic implementation and optimization by the logic synthesizer tool in the process of design compilation.

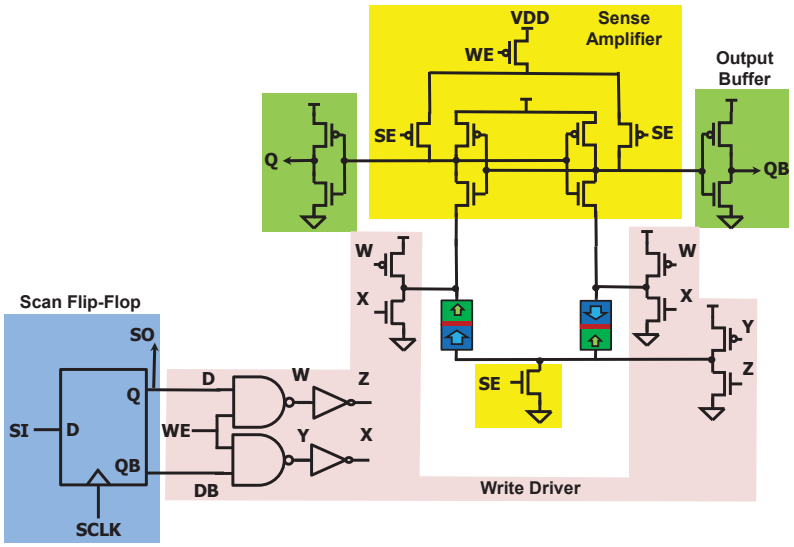
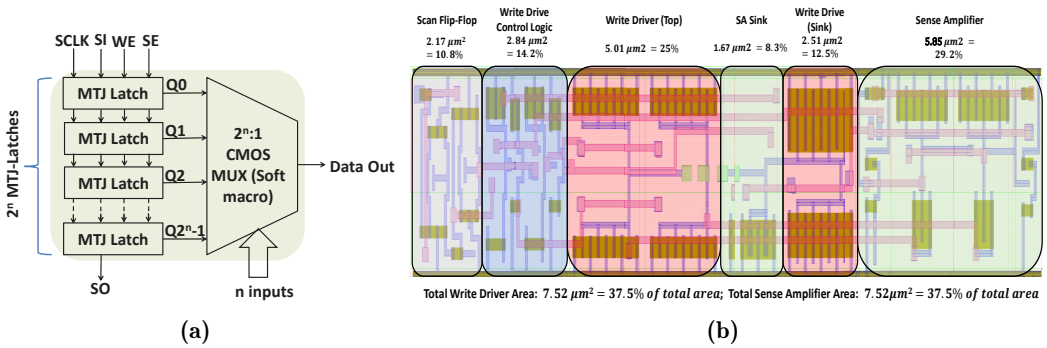


Figure 4.2. MTJ latch with Scan Chain Programming.

Each configuration bit is stored by a MTJ latch with scan chain programmability, as shown in Figure 4.2. The MTJ latch uses a pair of differentially programmed MTJs for non-volatile storage, a pre-charge sense amplifier for sensing the state of the MTJs, and three write driver schemes for parallel writing both MTJs simultaneously with each MTJ receiving full voltage swing, offering more write current. The Sense Enable (SE) signal must be low during the write operation, and the Write Enable (WE) signal must be low during the sensing operation. To avoid conflict of state between the pre-charge state of the sense amplifier (when SE=0) and the state of the write driver outputs in the write mode, the pre-charge path to V_{DD} is disconnected via the PMOS driven by the WE signal. The MTJ latch uses a dynamic latched sense amplifier that needs to be fired (SE low to high pulse) once on every power-up to convert the resistive state of the MTJs into the volatile voltage states at the outputs (Q and QB). In this configuration, the MTJs are read-only once, and for the remaining time in the active mode, the LUT read power and delay are determined by the static MUX. Moreover, by not reading from the MTJs repetitively in the active mode as in the dynamic STT-LUT styles, the stress is removed from the MTJs, enhancing their lifetime. Another critical thing to note here is that the scan chain used to program the LUT is separate from the one used in the design for testing and thus does not require critical MUX to switch from functional mode to test mode.

The MTJ latch is designed in a full-custom manner and needs to be optimized for sensing



(a) **(b)**

Figure 4.3. (a) Proposed STT-LUT with Scan Chain Programming (b) Full custom layout of MTJ latch in standard cell format and area breakdown between different blocks

reliability and area. The custom-designed MTJ latch is delivered as a standard cell for integration into the ASIC design flow. The full-custom design and optimization of the one-bit MTJ latch cell are performed in the Synopsys generic 28nm process [28]. The write drivers tend to require large transistors to produce sufficient current needed for MTJ write. The write transistors need to be optimized so that the write operation can succeed under Process Variation (PV). This experiment performed statistical transistor sizing optimization on the write driver for achieving a near-zero (less than 0.1%) write failure rate under process variation. After the write driver sizing optimization, the read path (i.e., the sense amplifier) transistor sizes are statistically optimized for achieving less than 0.1% sensing failure rate at the smallest possible area. Moreover, a minimum-sized scan flip-flop is inserted in front of the MTJ latch to store the data written to the MTJ latch. These scan flip-flops will form a scan chain for loading the configuration bits to the MTJ latches in a design. Figure 4.3 (b) shows the full-custom layout of the one-bit MTJ latch designed in a standard cell layout format (fixed height). The write drivers occupy most of the layout area (37.5%) since the MTJ write current is still significant. Notice that the MTJ devices are stacked on top of this layout between two metal layers (assuming M3 and M4) and do not occupy the 2D area. M3 pins are placed for connection to the MTJ layers.

4.2.2 STT-Based LUT versus CMOS-based LUT

Figure 4.4 shows the comparison of the area of the MTJ latch and STT-LUTs, along with the areas of other standard cells in 28nm. The MTJ latch area is 6× to 15× that of basic logic gates and 3× larger than rhe SRAM-based D flip-flop. The MTJ latch, however, shows much less leakage

power. It has 7× to 11× less leakage power than basic CMOS logic gates and is 20× smaller than SRAM-based D-FF.

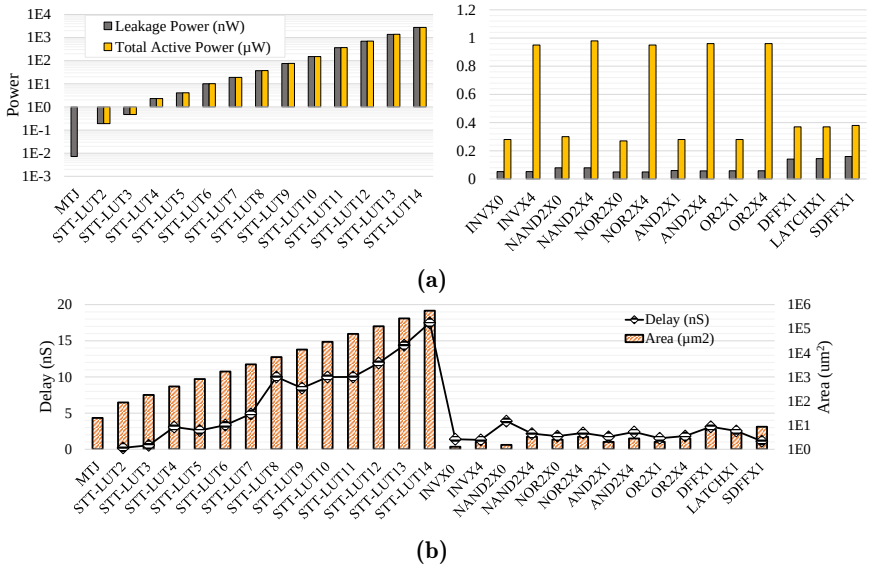


Figure 4.4. Comparison of (a) Power, (b) Delay, and Area of STT-LUT and Standard Cells in 28nm

The delay and active mode power of the STT-LUT are determined by the multiplexer part of the LUT, which is optimizable by the logic synthesizer. Figure 4.4 compares delay and active mode power for various fan-in STT-LUTs with standard cells. LUT_2 to LUT_7 have delays comparable to the standard cell delays. Due to the large MTJ latch area, LUTs are noticeably larger than the standard cells, and their area increases exponentially with fan-in. The power of STT-LUTs is significantly less than the standard cells due to low leakage MTJ latches. Therefore, relying on STT-LUT will render a lower overhead footprint.

4.3. Effect of Replacement Strategies

The location of the gate(s) selected for the obfuscation is one of the critical factors that define the strength of the obfuscation [11]. The replacement strategy finds the gate for obfuscation based on heuristics. LUT-based obfuscation needs to select the gates for replacement; however, an effective replacement strategy needs to meet several conditions to provide resiliency against the SAT-attacks. The two most essential requirements are (1) low corruptibility and (2) avoiding unintentionally correct key generation. By considering these conditions, this section introduces a replacement strategy and compares it with the random placement strategy [13]. To better evaluate the impact

of each condition, three different strategies are compared in this work as follows.

4.3.1 Random Selection (RND)

In the random selection algorithm, the gates are selected for obfuscation randomly. This method works as a baseline for comparison as opposed to the *independent selection* in [12].

4.3.2 Low Output Corruptibility (LC)

The state-of-the-art SAT-solvers use the Conflict Driven Clause Learning (CDCL) algorithm to find the solution. The CDCL works by searching for the conflicting clauses to learn clauses effectively and comparing the two outputs of the same netlist upon applying the input pattern with different keys helps find the conflicting clause. If the Hamming Distance (HD) between the two obtained outputs is high, finding the conflicting clause and distinguishing input is easy. An obfuscation strategy that influences more than one primary output (PO) of a circuit on the application of a wrong key input will result in a higher hamming distance, and the probability of *hamming distance* > 1 will be significant; (more than one primary output differs from the Oracle). This phenomenon is also known as the property of the obfuscation strategy to have high *output corruptibility*. High corruptibility leads to higher hamming distances, which provides the SAT-solver with an opportunity to find the conflict clauses much faster, resulting in lower de-obfuscation time. Due to this phenomenon, the maximum of one output must be different when a wrong key is applied to increase SAT execution time. This is called having low output corruptibility; if fewer (the best is 1) outputs are different after applying different keys and input. This phenomenon can also be thought of as reducing the observability in the presence of obfuscation, such that the effect of the wrong key can only be observed at fewer primary outputs. Due to this, sensitizing the key input to the primary output becomes intricate.

For the IP to exhibit the low output corruptibility, Breadth-first-search is employed on the circuit graph, where gates are treated as a node and edges are the connection between the logical gates. While traversing from Logical Cone Output (LCO) toward the inputs, a dictionary with all the gates and their corruptibility will be created, which allows us to pick the combination of the gates with the lowest output corruptibility. After traversing, based on the number of gates targeted for obfuscation, multi-objective optimization is performed to maximize the number of gates selected

for obfuscation while minimizing the output corruptibility.

4.3.3 Avoiding Unintentionally Correct Key Generation (LC_NoGen)

As the LUT is a reconfigurable unit, it can implement 2^{2^n} possibilities where n is the size of the LUT. Obfuscating two gates back to back with LUTs can generate additional correct keys. When the number of correct keys is increased, the SAT solver's search space to find the working key reduces the de-obfuscation time. Consider an example where two "NOT" gates are replaced with LUTs. In this scenario, if the LUTs are configured as buffers instead of "NOT," the circuit will still be equivalent to the oracle, and thus there exist 2 correct keys instead of 1. This means the probability of finding a key is doubled instantaneously. Therefore, obfuscating the gates with LUTs directly connected in a back-to-back fashion significantly decreases the SAT solver's search space. Due to the virtue of the reconfigurability in the LUTs, the number of correct keys must be reduced, and thus extra care should be taken to avoid this condition.

Considering the above case of avoiding the obfuscation of two directly connected gates, this work proposes a *LC_NoGen* replacement strategy. The pseudocode for the *LC_NoGen* replacement strategy is illustrated in Algorithm 2.

The algorithm involves traversing the graph followed by dictionary creation. The dictionary is filtered to eliminate the back-to-back gates and gates that contribute to the critical path. Further, an optimizer maximizes the gate coverage while reducing output corruption. The complexity of traversing the graph with V vertices and E edges is given as $\mathcal{O}(V + E)$. With fewer modifications, the dictionary creation and filtering can be done while traversing the graph, and thus, the overall complexity for traversing the graph and dictionary creation is given as $\mathcal{O}(V + E)$. The optimization problem, on the other hand, is a combinatorial optimization problem. While finding the optimal set of gates for obfuscation is an NP-hard problem. Thus off-the-shelf Integer Linear Programming (ILP) solvers are leveraged to get the sub-optimal solution using an approximation algorithm. The problem of gate selection with minimum output corruptibility is treated as a *variant* of a classical problem of minimum vertex cover. The task is to choose a minimum number of nodes that maximize the cover. The gates selected by the optimizer are replaced with the LUT, and the work illustrates that by considering the *LC_NoGen*, the Security per PPA overhead footprint is improved. The proposed algorithm can also be scaled to other obfuscation techniques for inserting the obfuscation

Algorithm 2: Avoiding Unintentionally Correct Key Generation (LC_NoGen)

```
LC_NoGen (Circuit C_org);
Input  : Original Circuit
Output: Obfuscated Circuit
foreach LCO in Logic_cones do                          // LCO: Logic Cone Output
    gate_list = BFS(LCO);            // Get all gates in the logic cone;
    foreach gate in gate_list do
        // Find all the Output gates of Logic Cone, which are affected by each
        gate.;
        gate.listLCOs = find_affected_LCOs(gate);
    end
end
foreach (gate in circuit) do
    foreach (LCO in gate.listLCOs) do
        tag_key(LCO);
        if isExist(tag_key(LCO)) then
            dictionary.add(gate);
        end
        else
            dictionary.add(gate);
            dictionary.addtag((tag_key(LCO));
        end
    end
end
CriticalPath = PrimeTime(Get Critical Path);
      // Get List of gates that are on Critical Path using Synopsys PrimeTime.
  Afterwards also remove gates that are adjacent to each other to avoid
  back-to-back LUT replacement.  Remove gates on critical path list;
foreach (tag in dictionary) do
    foreach (gate in tag) do
        if isExist(Parent(gate) in tag)
              or isExist(gate in CriticalPath) then
            dictionary[tag].delete(gate);
        end
    end
end
           // find tag_key which have maximum gate coverage with lowest Output
  Corruption;
tag_key = Optimize ();
foreach (gate in tag_key) do
    Replace_LUT(gates, target_no);
                              // Replace gates with LUT;
end
```

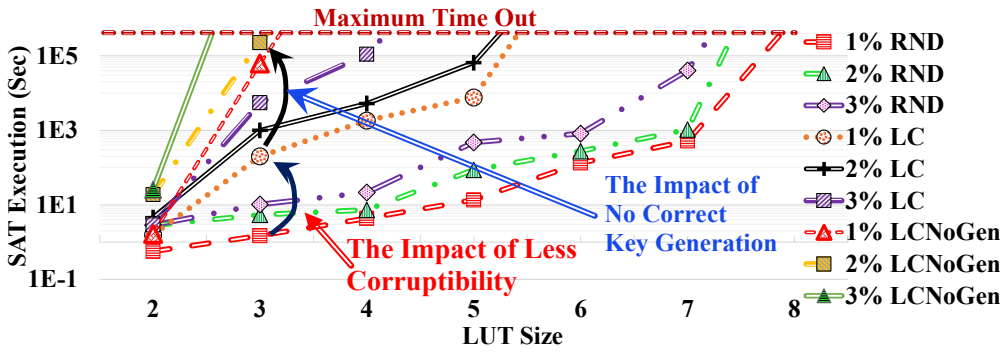


Figure 4.5. De-obfuscation time using SAT-attack for Different (1) Replacement Strategy, (2) LUT Size (scale up), and (3) Number of LUTs (scale out) in ISCAS-85 c7552 [4]

as this algorithm finds the optimal place for inserting a gate that minimizes the output corruptibility.

After developing replacement strategies tailored for LUT-based obfuscation, they are evaluated against the SAT-attack. Benchmark C7552 from ISCAS-85 is obfuscated with the different discussed obfuscation strategies for the experiment. Figure 4.5 illustrates the performance of the discussed obfuscation strategies against the SAT-attack. With the improved obfuscation strategies, the de-obfuscation time of the SAT solver is increased, and in most cases, obfuscation using the *LC_NoGen*, strategy outweighs the Random and *LC* obfuscation, which shows that high security can be obtained with a lower number of gates obfuscated. The lower number of gates used for obfuscation results in lower PPA overhead. However, while providing higher security at the cost of lower PPA overheads, these obfuscation schemes tend to have low output corruptibility.

While on the other hand, it can be observed that for the LUT with size 8 and above, obfuscating ~1% of overall gates with *any* obfuscation strategy also renders the timeout states for the SAT solver. Moreover, one can get the security against the SAT-attack and the increased output corruptibility for a random obfuscation scheme using a large LUT size. **Therefore, by scaling up the size of the LUT for obfuscation, the LUT-based obfuscation can break the trade-off of SAT-resiliency with output corruptibility.** Increasing the size of the LUTs increases the resiliency of the IP regardless of the replacement strategy.

The experiment also shows that using a larger LUT size overweighs the benefits of using a better gate replacement policy at the expense of increased design overheads. Another observation from the experiment is that a change in the obfuscation coverage from 1% to 3% (i.e., changing the number of LUTs inserted in the circuit) also increases the runtime of the SAT-solver. Thus, in the

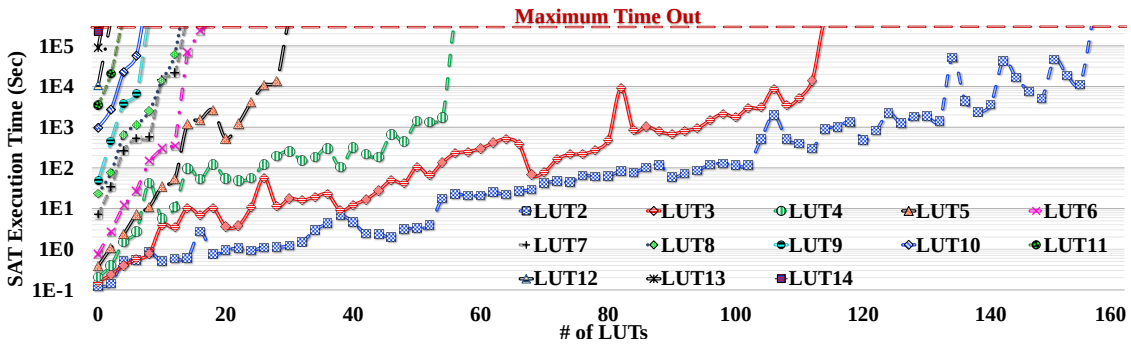


Figure 4.6. De-obfuscation time of C7552 benchmark with different number of LUTs and different sizes of LUT using SAT-attack [4].

next section, the effect of the size of LUT vs. the number of gates obfuscated (obfuscation coverage) on the circuit resiliency is discussed in more detail.

4.4. LUT Size versus Number of LUTs

To get the best security results using LUT-based obfuscation, it is wise to leverage the large size of the LUT because increasing the LUT's size can thwart the SAT-attack and increase the output corruptibility. This advantage of using larger LUT sizes is that LUTs are modeled using the $\log_2(n)$-level MUX-based structure, and with the increasing size of LUT, the SAT-attack will replace them with the deep MUX trees. The SAT-attack leverages the CDCL algorithm to find the distinguish input. However, when the symmetrical structure of the MUX tree is used for the obfuscation, there is no shortleaf in finding the conflict clause while running the SAT-attack. The increasing size of LUT grows the MUX tree deep, and consequently, the search space and the efforts for finding the conflicting clause are increased, making the resulting instance an SAT-hard. The following example shows how the de-obfuscation time rises exponentially, with larger LUTs to find the keys' value.

For larger LUT sizes, combinations of gates are replaced using large LUT. Figure 4.1 (a) shows an example of the large-sized LUTs used for the obfuscation. The 2-input gates (G_4, G_6, G_7, G_9) are replaced using LUT of size 4. The LUTs are represented using MUX-tree as shows in Figure 4.1 (b) for SAT-attack simulation. To evaluate the effect of LUT size vs. the number of LUTs, multiple gates with varying LUT sizes and various gates with a varying count of LUT are used.

Figure 4.6 shows SAT execution time with more details on the ISCAS-85 C7552 benchmark for different sizes of LUTs used for obfuscation. This experiment shows a similar trend of leveraging

larger LUT size provides higher resiliency than utilizing more LUTs for obfuscation. SAT execution time increases at the near exponential rate in both directions, i.e., scale-up (increasing the size of LUTs) and scale-out (increasing the number of LUTs). However, obfuscating only a single gate with a LUT of size 13 can render a timeout state for the SAT solver.

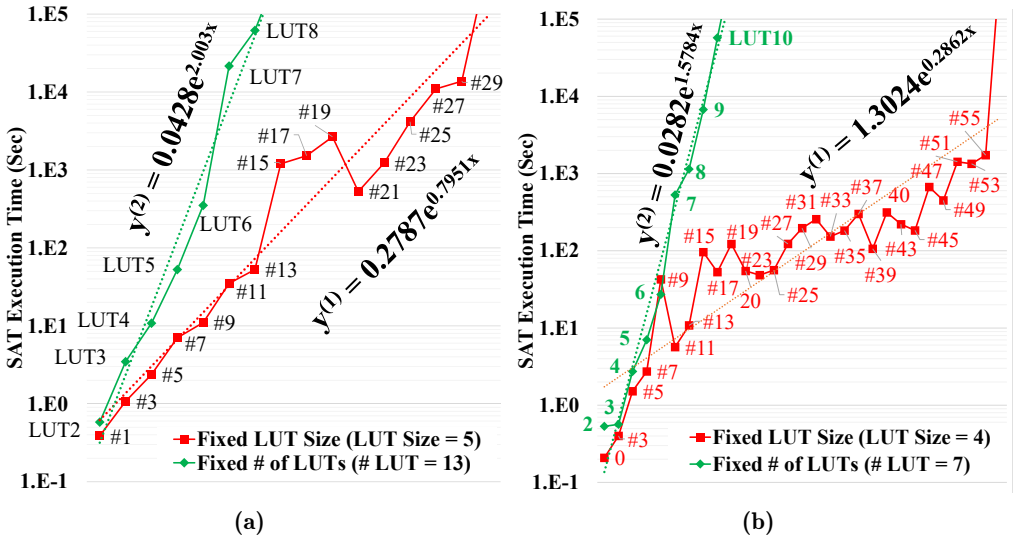


Figure 4.7. LUT scale up vs. scale out: Comparison between the Impact of LUT size and Number of LUTs on SAT Execution Time [4].

To further substantiate these results and understand and compare the impact of LUT scale up vs. scale out on SAT execution time, a regression model is utilized to demonstrate the relationship between SAT execution time and these two parameters. Figure 4.7 provides two different scenarios to accurately model the relationship between SAT de-obfuscation time and the size of LUTs, and the number of LUTs. In figure 4.7 (a), one factor (LUT size or number of LUTs) is fixed in each curve. LUT size is constant in one of them, while the number of LUTs is swept from 1 to 29. In another curve, the number of gates obfuscated is constant, and the size of LUTs has been swept from 2 to 8. Based on the independent (one-variable) exponential regression model illustrated on curves, it is clear that the LUT scale-up has significantly more influence on SAT execution time than the LUT scale-out. Figure 4.7 (b) shows another similar situation that proves that the LUT scale-up is more effective than the LUT scale-out. Also, according to a multi-dimensional linear regression, the impact coefficient of the number of gates obfuscated using LUT on SAT execution time is 72.347. However, the impact coefficient of using a large size of LUT is 1969.25. This regression coefficient demonstrates that LUT size is the most important factor for security purposes than the number of

LUTs used and replacement strategies. **Therefore, to render the best security, one should replace a few gates with large LUT sizes instead of opting for obfuscating more number of gates with small LUT sizes.**

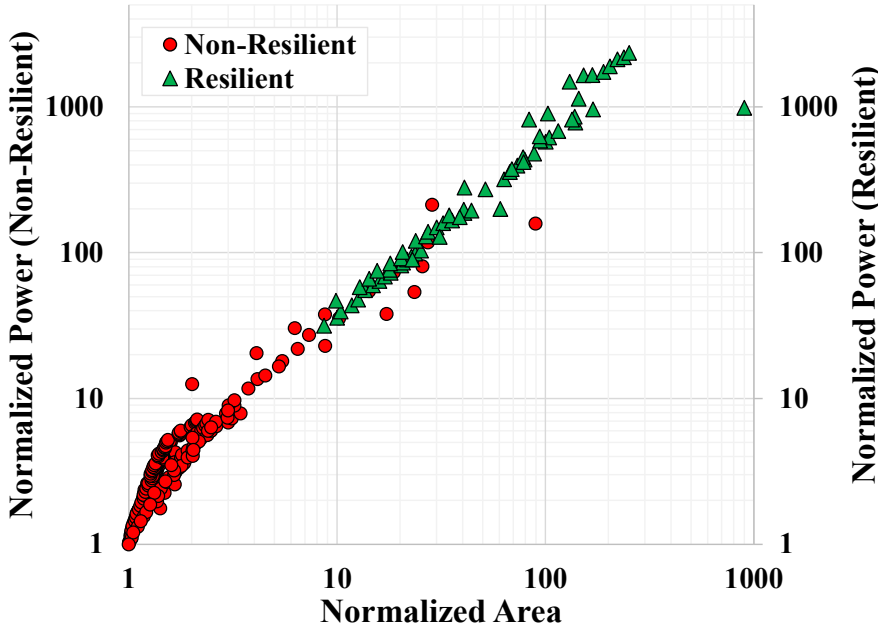


Figure 4.8. Normalized area and power overhead of LUT-based obfuscation. Points that result in SAT execution time out are marked as SAT-resilient configuration [4].

The thorough experiment on LUT-based obfuscation by considering (1) LUT technology, (2) LUT size, (3) number of LUTs, and (4) replacement strategy, it can be concluded that scaling up the size of the LUTs for obfuscation yields the excellent resiliency against state-of-the-art attacks. However, the investigation on PPA shows a sharp increase in the design overheads when using the large size of LUT. Figure 4.8 shows the Area and Power overhead for LUT-based obfuscation. Every point from Figure 4.6 was synthesized using Synopsys Design Compiler using TSMC 65nm library to obtain the power and area overheads. To render the timeout state for SAT solver using the LUT-based obfuscation, the PPA incurred is at least 10× the baseline version of C7552 with no obfuscation. **The enormous obfuscation overhead due to leveraging the traditional LUT-based obfuscation makes the LUT-obfuscation an idealistic method for hardware security.** This experiment also lets us find the smallest LUT size required for resulting in SAT-resilient obfuscation with lower PPA. For this example, the LUT of size 8 results in SAT-timeout with the lowest PPA among all samples, and thus in the following experiments, the LUT of size 8 is used as the baseline for

comparing obfuscations.

Given the benefits of the LUT-based obfuscation, and to realize it as a *realistic* solution, the work must 1) radically reduce the PPA overheads and 2) not compromise the security against the various attacks. However, both the goals are contradictory to each other with the discussed LUT-based obfuscation. One can reduce the size and number of LUTs, but that compromises the security against the SAT-attack. The following section proposes a novel LUT design that benefits from configurable barriers for obfuscation and mitigates the incurred area and power overheads.

Chapter 5

Breaking the design and security trade-off of Lookup Table-based Obfuscation

This chapter expands upon the idea of breaking the tread-off between security and design overheads in LUT-based obfuscation. The organization of the chapter is as follows: Section 5.1 identifies the areas of improvement for resisting the SAT-attack. Section 5.2, discusses the idea of using LUT for creating SAT-hard block that can resist the SAT-attack. Section 5.3.1 outlines the experimental setup to empirically prove the resiliency of proposed primitive against SAT-attack. The results are present in Section 5.3.2. Finally the obfuscation is compared against state-of-the-art obfuscation primitive in Section 5.3.3 and the scalability of the approach is demonstrated in Section 5.3.4. As an outcome of the research covered in this chapter, this work has contributed to [4, 25, 26].

5.1. SAT-Attack and Lookup Table-based Obfuscation

LUT-based obfuscation has the potential to resist SAT-attack but requires extra efforts to reduce overhead while maintaining resiliency. While the hardware security community has leveraged other obfuscation, this work tries to establish that LUT-based obfuscation is truly the best way to obfuscate a design. Recently many SAT-resilient obfuscation techniques use one-point functions to guarantee SAT-resiliency. One-point functions in obfuscation are logic locking circuitry added into the circuit, and the effect one-point function is concentrated at one single point. This makes the resulting instance SAT-hard which makes the process of SAT-solver inefficient, as it has to brute

force almost through all combinations. However, since the effect of logic locking is concentrated at one point, the rest of the circuit is functional even with the wrong key. Using removal techniques, which use structural examination, has demonstrated that obfuscation based on one-point function can easily be removed from the circuit. This means finding the root node of the AND-tree and removing that part from IP. Since logic locking adds extra gates, it is vulnerable to removal and SAT-attack. LUT-based obfuscation, in contrast, replaces the gates, and removing LUTs doesn't help the attacker in RE the circuit.

Moreover, using large LUT sizes for obfuscation increases SAT resiliency as it creates a SAT-hard instance. The resulting instance is SAT-hard as the CDCL algorithm, which is responsible for finding the DIP, needs to consider the increased search space. However, with the larger LUT sizes, the LUT obfuscation renders an idealistic method, and to make it an efficient obfuscation method, one must break the trade-off between the security offered by the LUT-based obfuscation and the imposed PPA overheads. In the proposed method, the work focuses on the Davis-Putnam-Logemann-Loveland (DPLL) algorithm (or one of its derivatives) used to perform CDCL. Utilizing the knowledge from the working of SAT-based attack, this work proposes SAT-hard problem using the small size of the LUT such that increased security can be obtained at the lower PPA footprint compared to the traditional LUT-based obfuscation.

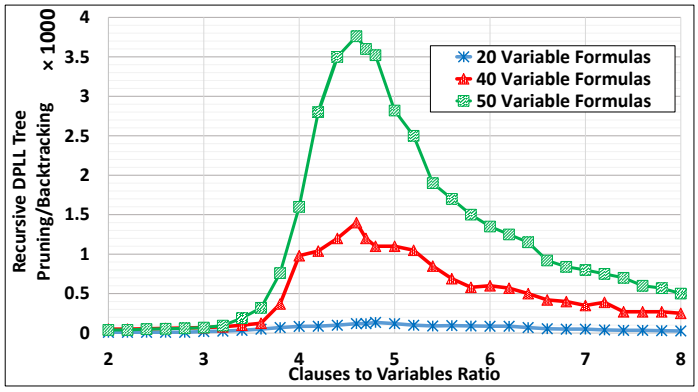


Figure 5.1. Median Number of Recursive DPLL Tree Pruning/Backtracking for Random 3-SAT Formulas, based-on the Ratio of Clauses to Variables [1, 5].

For the SAT-attack modeling, the obfuscated netlist is represented using the MUX tree, as discussed in Chapter 4. In the SAT solver, the ratio of the clause to variable can be used to evaluate the obfuscation strategy's security quantitatively. The resulting obfuscation instance can be called SAT-hard if the clause to variable ratio is around 4.3 [29]. Figure 5.1 shows that the

ratio from 3 to 6 provides much higher DPLL calls, and 4.3 [29] clauses per variable is the best ratio for generating the most computational SAT-hard instance, generating the highest number of DPLL calls. For example, a 100-variable and 300-clause instance (clause/variable = 3 is called "under-constrained" because there are many satisfying assignments), or a 100-variable and 5000-clause instance (clause/variable = 50 is called "over-constrained" because the contradictions can often be easily found) can be solved using the SAT solver very quickly. However, the SAT solver takes a long time to solve a 3-SAT instance constructed with 100 variables and 450 clauses.

Table 3.1 shows the Tseytin transformation [30] of various logic gates into their respective CNF expression. Only XOR/XNOR and MUX have 4 clauses per gate. This is when the clauses to variables ratio are 1 and 4/3 in MUX and XOR/XNOR, respectively. Despite the observation that the XOR/XNOR has a larger clause to variables ratio for a single gate, MUXes provides a better building block for constructing SAT-hard circuits. This is because: (1) with no unit propagation and purification, for having four variables, a MUX can make the recursive DPLL tree one level deeper, and (2) unit propagation and purification steps in the DPLL algorithm provide more simplified and smaller formula using enhanced Gaussian elimination while the contribution of XOR/XNOR gates is much higher [31]. Hence, MUXes need more DPLL recursive tree prunings/backtrackings than XORs/XNORs. Moreover, since unit propagation and purification satisfy fewer equations or clauses, the clause to variable ratio will increase.

5.2. Creating SAT-hard instances using Lookup Table

The next step for building an SAT-hard problem is to push the clause to the variable ratio in the desired range of 3 to 6 (4.3 being the best)[29]. This prevents the propagation and purification algorithm from simplifying the circuit before branching into a recursive DPLL tree. This agenda of pushing the clause to the variable ratio in the desired range can be achieved by building a MUX tree. This property can be utilized to reinforce the security offered by the conventional LUT-based obfuscation. This work combine the traditional LUTs with an extra layer of LUT whose input size is fixed to 2. The size of the small LUTs is restricted to 2 to impose lower PPA overheads. The resulting novel LUT combines a large LUT with the addition of the smaller LUTs at its input. The scenario can be visualized as adding MUX-tree by large LUT, supplemented by another 2-input

LUT, which resides at the select line of the previously inserted MUX-tree as shown in Figure 5.2 (b). By adding 2-input LUT, the MUX-tree can be imagined to grow in 2-D space (For example, a traditional LUT can be represented by the MUX tree growing in the horizontal direction, and the addition of MUXes on their select lines grows the MUX-tree in the vertical direction.) The proposed modification of adding another layer of LUTs to the LUT-based obfuscation benefits the IP from both reconfigurable and routing obfuscation. The additional layer of LUT increases the possible function of the large LUTs, thus increasing the search space for the SAT-solver. To restore the functionality of the IP, one has to find the correct functionality of both the small and large LUT simultaneously. For the n-LUT_2 followed by LUT_n, the number of possible functions implemented are

$$F_{possible} = n \times \underbrace{2^{2^2}}_{\text{Functions implemented by LUT}_2} \times \underbrace{2^{2^n}}_{\text{Functions implemented by LUT}_n} \tag{5.1}$$

When the output of one LUT feeds into the input of another LUT *indirectly*, the number of possible functions grows even more rapidly.

$$F_{total} = F_{LUT1} \times F_{LUT2} \tag{5.2}$$

Moreover, the attacker difficulty is increased as the observability of the effect of change in key bits of the previous LUT is masked by the current LUT. Indirectly cascading LUTs in LC_NoGen further decreases the observability.

Increased difficulty due to MUX-tree increases the search space, while reduced observability elevates the security offered by the proposed novel LUT. With the growing search space, the probability of finding a correct key is significantly reduced. The probability of finding a single key when 2 LUT_7 + 7:LUT_2 are added to the circuit is given by

$$P = \frac{1}{1.452496e + 81} \tag{5.3}$$

where

- $1.452496e{+}81$ is a number of functions implemented by 2 LUT_7 + 7:LUT_2 which are cascaded indirectly.

As the security obtained by the novel LUT is superior to the traditional LUT, one can reduce

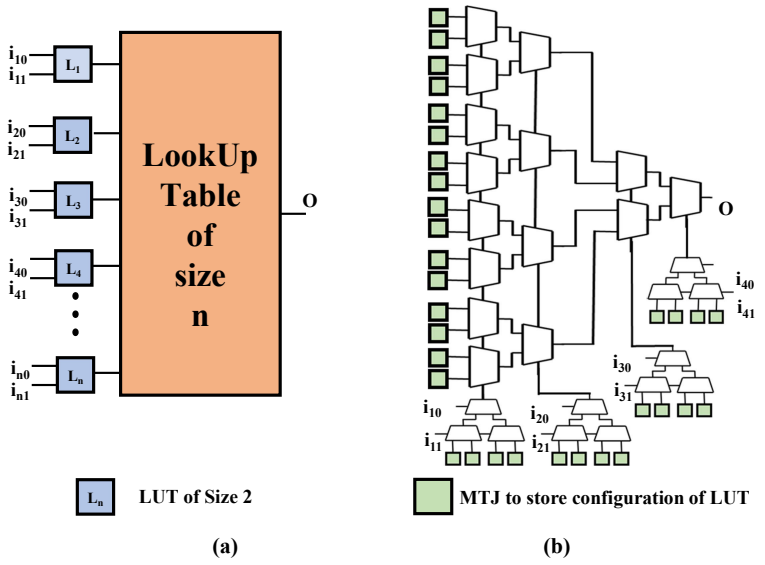


Figure 5.2. (a) Proposed novel (LUT_n+n:LUT_2) (b) SAT-representation of proposed obfuscation with size 4 (LUT_4+4:LUT_2)

the LUT size required for the obfuscation, resulting in lower PPA and thus breaking the trade-off between security and the PPA.

Figure 5.2 (a) shows the generic version of the proposed novel LUT. For obfuscation using traditional LUT, the gates are identified and replaced using LUT_8 as LUT_8 renders the timeout scenario as shown in Figure 4.8 with the lowest possible PPA. On the contrary, in the novel LUT, the gate is always replaced with the combination of 2-input LUT and LUT_n such that $n < 8$, where n is the size of LUT. Part (b) of the figure shows LUT_4, preceded by 4-LUT_2. For the de-obfuscation process using the SAT-attack, one can model the novel LUT block with the equivalent circuit shown in Figure 5.2 (b).

Once the gates have been replaced with the proposed novel LUT, the clause to the variable ratio of the obfuscated block is within 4-5 creating a SAT-hard instance. Figure 5.3 denotes the clause to the variable ratio of both pre-obfuscation (Original) and obfuscated circuit.

5.3. Experimental Evaluation

5.3.1 Experimental Setup

To explore the design space and determine the impact of LUT size, the number of LUTs, and the replacement strategy, a cluster computing environment with 53 Dell computing nodes were used,

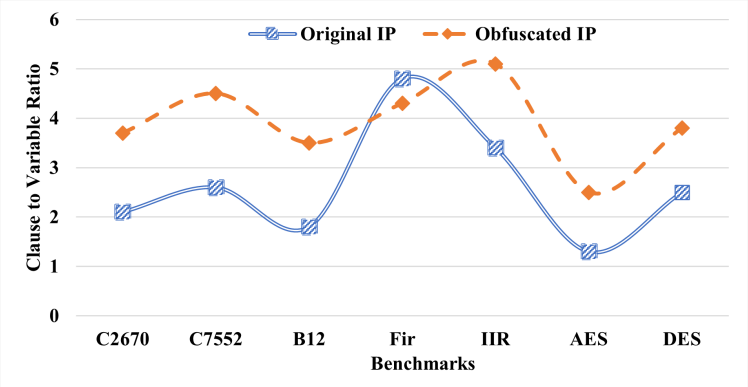


Figure 5.3. Effect of obfuscation on the Clause to Variable Ratio.

each with dual Intel Xeon CPUs. The total number of cores ranges from 16 to 24, with RAM varying from 64GB to 512GB.

Benchmarks from ISCAS-85[1], ISCAS-89[2] and Common Evaluation Platform (CEP)[3] are used for the experimental evaluation. The benchmarks are listed as part of Table 5.1. The CEP is a system on a chip design that represents typical microelectronics used by the Department of Defense (DoD) and includes instrumentation and government-specific benchmarks. These benchmarks are synthesized and flattened using the Synopsys Design Compiler.

Table 5.1. Benchmarks used for Experimentation with their gate counts.

Source	ISCAS			CEP				
Benchmark	C2670	C7552	B12	FIR	IIR	AES	DES	GPS
Gate Count	894	1,290	2,017	11,875	12,067	20,795	98,341	177,263

The adversary's aim is to retrieve the key to unobfuscated the IP using SAT-attack. For empirical evaluation, the primitive is subjected to SMT-attack [24], which is the super-set of SAT-attack and the newly developed state-of-the-art attack. The newly developed SMT-attack uses a primary SAT-solver with extra theory solvers. Combining two solvers allows the SMT-solver to model a more complex problem, thus resulting in a strong attack. The SMT-attack tries to find the correct key (LUT-configuration bits in the context of LUT-based obfuscation) that can restore the circuit's correct functionality. **For security evaluation, the runtime of the SAT-attacks is used as an empirical yet essential metric.** Each SAT-runtime presented in this work is the average run of 10 different

[1] http://www.pld.ttu.ee/~maksim/benchmarks/iscas85/verilog/
[2] http://www.pld.ttu.ee/~maksim/benchmarks/iscas89/verilog/
[3] https://www.ll.mit.edu/r-d/projects/common-evaluation-platform

SAT-solver execution. The obfuscation technique aims to render maximum security at the lower PPA overheads. The outcome of the experiments conducted in this study shows that the proposed LUT can deliver security at minimal PPA footprints, increasing the viability of the reconfigurable based obfuscation.

This work empirically track and explore the execution time of the SMT solver by sweeping the size of large LUT from 4 to 7. Also, a run-time limit of 5 days (432×10^3 seconds) for SMT attack is set to demonstrate *timeout* states. The timeout state of 5 days is chosen to demonstrate the toy example that breaks the trade-off between the security and the imposed PPA. However, a larger timeout state can be achieved by obfuscating using a larger LUT size or increasing the number of obfuscated gates.

For this experiment. The identified gates are replaced with STT-LUT technology, as described in 4.3.3, to produce an obfuscated netlist. Synopsys Design Compiler with the TSMC 65nm technology library is used for the overhead estimation. The reported overhead in this work is reported with respect to the unobfuscated design and it is inclusive of the overhead incurred due to the scan-chain mechanism used to load the values in MTJs. With the automation provided using the *ASIC Iterative Security-driven Design Flow* discussed in this work, the delay overhead is eliminated in almost all the cases for novel LUT-based obfuscation. Thus only area and power overhead are discussed in fine granularity. In the iterative flow, the *LC_NoGen* algorithm gets the timing report from the PrimeTime and removes the gates on the critical paths from the graph, which is used for finding the gates for obfuscation. The SAT-attack does not support the Verilog files for the SAT-simulation; the in-house developed python script also converts the obfuscated Verilog files to the SAT-supported bench files.

5.3.2 Security Analysis against SAT-based Attack

Figure 5.4(a) illustrates the Design Space Exploration (DSE) performed on the "C7552" benchmark by leveraging novel LUT-based obfuscation. By varying the size and the number of LUTs used for novel LUT-based obfuscation, the de-obfuscation time against the SMT-solver is plotted. The SMT-solver timeout states can be rendered using traditional LUT-based obfuscation by obfuscating 14 gates with LUT size 8. LUT size 8 is considered per the experimental results obtained in Figure 4.8; Recall that the combination of obfuscating 14 gates with LUT size 8 renders the lowest PPA with

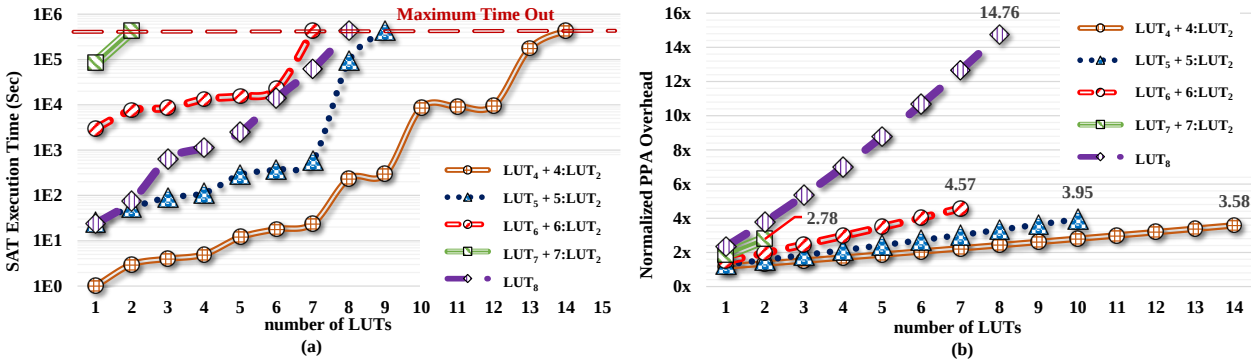


Figure 5.4. Comparison of the proposed primitive where the size of the LUT is varied from 4 to 7 with the traditional LUT-based obfuscation. Figure (a) shows the de-obfuscation time of various obfuscated samples using SMT-solver and Figure (b) shows the PPA incurred by the proposed primitive [4].

timeout state for SAT solver among all the configurations tested in Figure 4.8. However, with the proposed LUT. Replacing just over 2 gates with LUT_7 + 7:LUT_2[4] or replacing 6 gates with LUT_6 + 6:LUT_2 renders the obfuscation resiliency to create a timeout scenario. This increased resiliency with the smaller size and number of the LUT breaks the trade-off between PPA and security. This finding justifies the fact that the resiliency of the novel LUT provided by LUT size 6 and 7 is on par with that of the traditional LUT with size 8. This added security is due to the virtue of the SAT-hard instance created by the LUT preceded by the LUT configuration. The results further show that one can yield significant computational challenges for SAT-based attacks, which grow exponentially with the increasing size of LUT.

While leveraging the small size of the LUT, such as LUT sizes 4 and 5 for novel LUT, the number of gates required to achieve SAT-resiliency (timeout states) is more. Nevertheless, the overheads imposed by LUT of size 4 and 5 is far less than that of LUT size 8, and thus it can be concluded that the novel LUT with size 4 and 5 provides a high ratio of security per added design overhead than the traditional LUT of size 8. It is worth noting that all instances of obfuscation using novel LUT render SAT-attack timeout, resulting in the on par resiliency level using traditional obfuscation using LUT size 8. Figure 5.4(b) validates the previous statement by showing that obfuscating 14 gates with novel LUT using size 4 adds 3.58× overhead compared to 14.76× overhead added by traditional LUT of size 8.

[4] LUT_m + n:LUT_2 represents novel LUT where n LUTs of size 2 is preceded by LUT_m, where LUT_m represents LUT of size m.

The reduction in area and power is applicable for all of the experiments conducted as part of this work. The normalized area and power overhead from Figure 5.4(b) for novel LUT incurred lower overheads than the LUT obfuscation using LUT_8. It also warrants that using a large size of fewer LUT, i.e., using just 2 LUT_7 results in the lowest PPA and on par resiliency. As the resiliency provided by novel LUT increases as the function of the size of the LUT, one should replace a few gates with large LUT sizes, as it renders SAT-resiliency while incurring permissible overheads. These results are also consistent with the experimental results obtained from Figure 4.6.

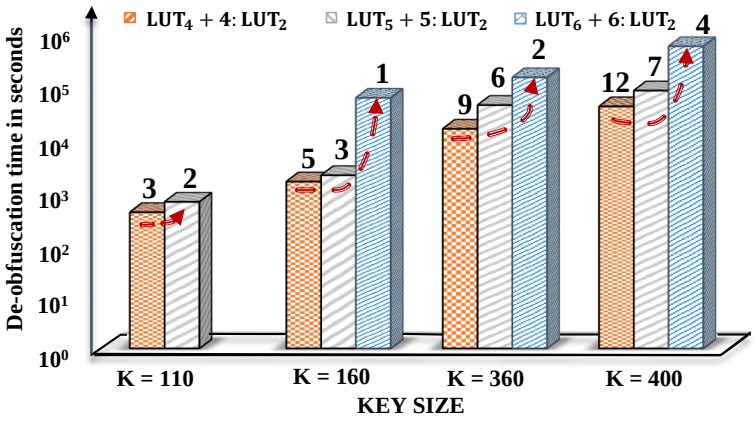


Figure 5.5. SMT-solver execution time to find the unlocking key when the benchmark is obfuscated with different size and number of LUTs. The number and size of the LUTs to be replaced are determined by the key size which is constrained in this experiment and denoted by the number above each individual bar [4].

To further reinforce the conclusion of using large LUT sizes in fewer quantities for better security against hardware security threats, this study performed another experiment where the key size used for obfuscation is constrained. The number of key bits is indicative of the overheads imposed. More key bits require more fuses to store configuration bits and a large selection tree. Increasing keys thus results in increased power and area overhead. By adding the constraint on the key bits, this work indirectly adds a constraint on the area, power, and the number and size of the LUT used for the obfuscation. The "AES" benchmark is utilized for this experiment, and the key lengths are constrained to sizes 110, 160, 360, and 400. With the constraint on the key lengths, one can use LUT of sizes 4 up to 6, and the number of gates replaced using the LUT is shown in Figure 5.5 over the bars. For example, when the key size is 360, one can have at max 2 novel LUT_6 + 6:LUT_2 or 6 LUT_5 + 5:LUT_2 or 9 LUT_4 + 4:LUT_2. For each configuration, SMT-solver's de-obfuscation time is plotted, and it can be visualized that the time required for de-obfuscation using a large size of LUT in fewer quantities yields in significant SAT runtime across all 4 key lengths. Using just 4 novel

LUTs of size 6; the SMT-attack encounter the timeout scenario, which is more than leveraging 12 novel LUTs of size 4 for obfuscation. This increased resiliency is created due to the virtue of using large LUT sizes and the large MUX trees that are added to the circuit. Leveraging the large size of the LUT obfuscates the actual function in the space, which grows exponentially as the function of LUT size. When the key lengths are equal, the overhead added is roughly equal. With the same overhead footprint or key size, using the large LUT size provides maximum resiliency. Thus, one should use large LUT sizes in fewer quantities for obfuscation using the novel LUT. Furthermore, the experiment concludes that security grows faster than the added overhead, or the additional security comes with lower PPA overheads when the large LUT size in fewer quantities is used. Using such configurations, the proposed novel LUT breaks the trade-off between security and design overhead.

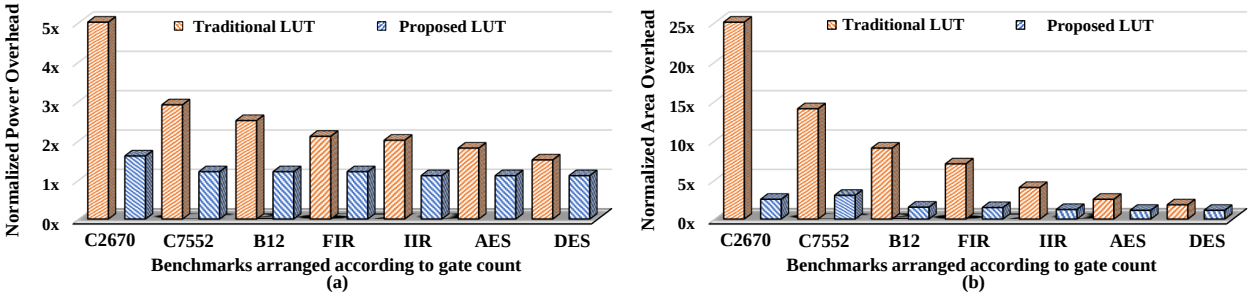


Figure 5.6. Comparison of Proposed LUT obfuscation with the traditional LUT-based obfuscation in terms of (a) Design Area overhead (b) Design power overheads. The number of LUTs and the size of the LUTs are chosen such that the obfuscated netlist results in the SMT-attack timeout with minimal PPA overheads. The iterative security-driven flow discusses in Section 8.1 can be leveraged to obtain these netlists. The added overhead for both traditional and obfuscated netlist is provided in reference to each benchmark's unobfuscated version. The figures are indicative of the fact that in proposed LUT-based obfuscation, the resiliency rendered by each LUT at comes at much lower costs of design overheads. [4]. Number on the top of the bars denote the overheads. where 1× denotes no overhead and 1.03× means 3% overhead.

Figure 5.6 shows the power and area overhead for the different benchmarks using the proposed LUT of size 7. Size 7 is used because the LUT of size 7 resulted in a lower PPA, as seen in Figure 5.4 (b). Timing results are omitted as all of the designs have maintained their initial timing specifications. As LUT_7 + 7:LUT_2 is the optimal PPA configuration, it incurred a small timing overhead while providing significant security performance. The overheads imposed by the novel LUT are compared with the traditional LUT-based obfuscation with LUT_8. LUT_8 is used for comparison, as it resulted in the SAT-resilient obfuscation while incurring the lowest PPA overhead, as seen in Figure 4.5. Compared to the LUT_8-based traditional obfuscation, the novel LUT with LUT_7 comes

with 8× and 2× average reductions in area and power overheads without sacrificing security. While one can argue that overheads for circuits such as "C7552" are very high, the circuit size of "C7552" is tiny (only 1290 gates). However, with the larger circuits like "AES," "DES," or "GPS" which is a representative of a real world IPs, the incurred overheads are justifiable rendering this technique a more efficient solution.

5.3.3 Comparison with other Obfuscation Methodologies

In this section, the proposed methodology is compared against the state-of-the-art works. SFLL-HD [32] and Anti-SAT attack [22] are SAT-resilient attacks, as the SAT-attack requires an exponential amount of queries to retrieve the keys.

For the comparison, various designs are obfuscated such that SAT-attack results in a timeout state while trying to retrieve the keys with minimum obfuscation overhead. Larger key sizes are required for the obfuscation primitives studied in this section for comparison against the proposed LUT-based obfuscation to result in timeout states. This is because the time out for SAT-attack simulation is set to 5 days, and most of the work uses few hours or couple of days as the runtime limit.

Table 5.2. Overhead comparison of various obfuscation techniques.

	Power overhead analysis				Area overhead analysis				SMT-Attack runtime			
Benchmarks	**Proposed**	SFLL-HD	AntiSAT	InterLock	**Proposed**	SFLL-HD	AntiSAT	InterLock	**Proposed**	SFLL-HD	AntiSAT	InterLock
C2670	53.27%	23.3%	33.74%	47.51%	91.53%	72.45%	73.42%	107.5%	∞	∞	∞	∞
C7552	20.4%	19.1%	18.5%	20.8%	91.53%	72.45%	78.42%	88.5%	∞	∞	∞	∞
B12	18.5%	18.8%	20.3%	17.04%	60.52%	38.64%	57.63%	55.14%	∞	∞	∞	∞
FIR	17.3%	14%	14.2%	13.8%	43.05%	30.86%	28.41%	39.54%	∞	∞	∞	∞
IIR	10.08%	3.8%	3.6%	8.46%	8.44%	7.91%	7.46%	8.6%	∞	∞	∞	∞
AES	2.75%	1.2%	1.8%	2.6%	4.94%	3.19%	3%	4%	∞	∞	∞	∞
DES	2.46%	1.1%	1.05%	1.9%	3.27%	2.59%	2.78%	3.18%	∞	∞	∞	∞

∞ denotes SMT-attack timeout.

From table 5.2, it is evident that the SFLL-HD and AntiSAT result in much lower Area and Power overhead than the proposed LUT-based obfuscation. This is true for the smaller benchmarks. However, with the increasing size of the benchmark, the LUT-based obfuscation overheads are comparable to both obfuscation primitives. Besides, it can be noted that Anti-SAT and SFLL-HD leverage a one-point function and thus has lower output corruptibility. This makes the **SFLL-HD and Anti-SAT vulnerable to approximation SAT-attacks [33] and removal attacks [34]. Though they have small design overheads for implementation, their applicability for obfuscation purposes is questioned.** Fall-attack demonstrated in [35] shows that the key for SFLL-HD can be retrieved

without having access to the oracle design.

On the other hand, LUT-based obfuscation allows the user to randomly replace the gates in the circuitry, thereby allowing the user to increase the output corruptibility while being resilient to SAT-attack, the approximation attack and removal attacks. Moreover, this study compares the state-of-the-art InterLock obfuscation, a unified routing and logic obfuscation technique [1]. The overhead for leveraging the Key programmable routing block size 64 is comparable to the proposed LUT-based obfuscation. It requires around 320 key programmable routing blocks where each block consists of 2 LUT_2 and 4 2:1 MUX. Moreover, using the size of 64 for yielding SAT-resiliency requires 10-stages of routing, making this obfuscation hard to deploy in a circuit. Proposed LUT-based obfuscation, on the other hand, can be deployed quickly and provides comparable overheads when compared to the state-of-the-art obfuscation primitives.

As discussed earlier, LUT-based obfuscation remains resilient to removal attacks, as removing the LUT from the circuit strips away circuit's functionality. The SAT-attack tries to find the configuration key for the LUT that can unlock the circuit, but the ample obfuscation space put forth by LUT-based obfuscation renders the SAT-attack futile. For the most significant design, such as DES obfuscated with proposed LUT-based obfuscation, when subjected to the SAT-attack results in error (internal error in "lglib.c": more than 894,489,346 variables). This shows that attacks cannot unlock the 100% correct functionality of the circuit using LUT-based obfuscation.

The proposed obfuscation was also compared against AppSAT [33]. For AppSAT, the termination criteria are determined by the error rate, which is one of the inputs to the attack. 50 random queries were performed (default setting of AppSAT) on the oracle attack after the key, which is given to us after the 20 iterations of the AppSAT. It is misleading to calculate the error rate using such a small amount of input patterns for the obfuscation with high output corruptibility [32]. When the number of queries are increased to 1000, AppSAT resulted in a timeout state. Nonetheless, the key given by the AppSAT when the number of queries was 50 or 1000 does not fully unlock the circuit's functionality.

5.3.4 Exploration of Large Benchmark

A large GPS benchmark from the CEP benchmark is chosen to show the efficiency of the proposed LUT insertion strategy. The benchmark is obfuscated with the optimal LUT_7 + 7:LUT_2 LUT

configuration discussed in previous sections to result in timeout scenario. For obfuscation, Civilian Acquisition (CA) code generator module inside of GPS was selected. The obfuscation resulted in 0.4% power and 0.3% area overhead. This design achieved an SAT-attack timeout with an area and power overhead of less than half a percent. This result affirms the trend established in the survey of small and medium-sized designs in the preceding sections: the power and area overheads required to implement the optimal SAT resilient LUT configuration amortizes as the design size

Through the various experiments and development, it is evident that LUT-based obfuscation can be used to resist many of the existing threats. However, having a single means of defense is not a great strategy to assure the best security. The constant red-teaming efforts in the research area have been able to find vulnerabilities in many of the obfuscation schemes, and thus for providing best zero-day attack security, this work further realizes the LUT-based obfuscation as a "defense-in-depth" mechanism. The next chapter illustrates the idea of a "defense-in-depth" tool.

Chapter 6

Robust "Defense-in-depth" solution using LUT-based Obfuscation

LUT-based obfuscation discussed in this work aims to thwart SAT-attack by exposing the SAT-solver to vast key search space. Similar approaches from the reconfigurable domain use a Magneto-Electric Spin-Orbit (MESO) device for obfuscation [36], which can offer both reconfigurability and dynamic morphing to thwart the SAT-attack. Since most of the works try to mitigate SAT-attacks by increasing the time required to find a solution, it is possible that with the increasing computing power and new research directions, it would be possible to break SAT-resilient techniques with new SAT-solvers.

The study in Section 6.1 shows the potential of using Power Side-Channel Attack (P-SCA) to understand the underlying behavior of the hardware by monitoring side-channel information, such as power consumption in particular. Using P-SCA with Machine Learning (ML) techniques can reveal the secret key required to unlock the obfuscated design. Therefore, there is a need for a technique to eliminate the SAT-attack and the ML -assisted P-SCAs. Thus, Deep-Learning Power Side-Channel Attack Mitigation using Emerging Reconfigurable Devices and Logic Locking (LOCK & ROLL) is proposed in Section 6.3.

Section 6.3.2 and Section 6.3.3 discusses how LOCK & ROLL resists SAT-attack and eliminates the threat of SAT-attack. Moreover, it provides resiliency to P-SCA and other attacks. This LUT-based solution provides "Defense-in-Depth" for current and any future zero-day attacks at minimal

design overheads. This chapter discusses the implementation and the empirical evaluation of the proposed primitive.

The contributions of LOCK & ROLL are as follows:

- The work proposes a Symmetrical Magnetoresistive Random-Access Memory based Look-up Table (MRAM-LUT) which successfully resists the ML-assisted P-SCA by attaining near-zero power variation in the output.
- The proposed obfuscation can still resist SAT-attack while maintaining a reliable and low-overhead solution with a wide read margin for LUT-based obfuscation.
- This work discusses the **Scan Enable Obfuscation Mechanism** (SOM), which is used to eliminate the SAT-attack.
- This work empirically evaluate the proposed primitive against ML-assisted P-SCA and demonstrate its comprehensive security coverage.
- Finally, the work empirically assesses the resiliency offered by the proposed "Defense-in-Depth" primitive against various attacks to demonstrate the broad applicability and resiliency of the LOCK & ROLL.

As an outcome of the research covered in this chapter, this work has contributed to [37, 38].

6.1. Power Side-Channel Attacks (P-SCA)

In the CMOS circuit, to change the state of the logic device, i.e., from zero to one or vice versa, the voltage is applied. This transition from one state to another reflects a side-channel signature, which can be captured through power consumption of the system or other electrical properties. Since the amount of power required in the system is proportional to the manipulated data, the power traces collected during the processing or transition of states contain vital information. Even a single transistor's effect can appear as a weak correlation in power measurements [39].

The proposed LUT-based obfuscation can be created using the architecture proposed in the work [6]. In this case, the P-SCA can be utilized to find the content of the LUT configuration. To demonstrate the process of RE using P-SCA, the current of 2-input LUT for different functions

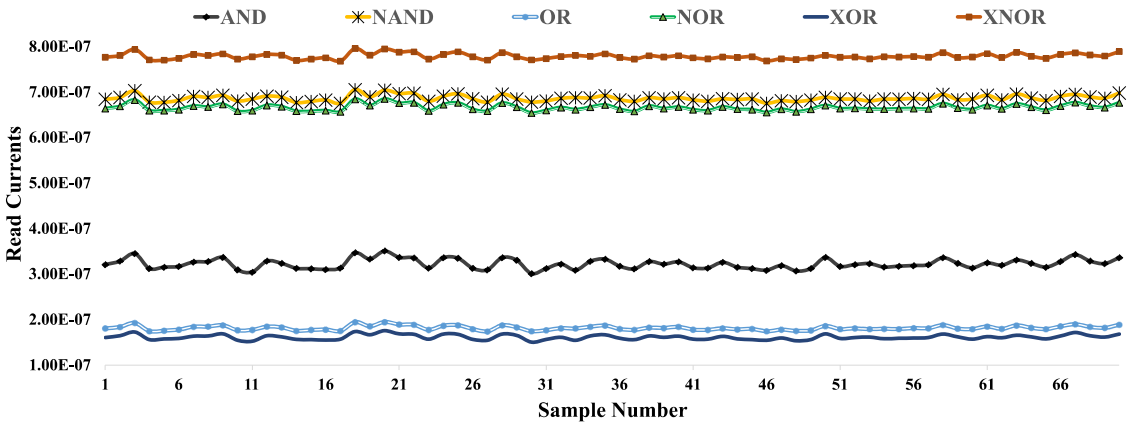


Figure 6.1. Read current traces of a 2-input MRAM-LUT in [6]. (Y-axis: Read Current in Amps; X-axis samples in collected data)

implemented by the LUT is measured using HSPICE simulation. Figure 6.1 shows how a LUT implemented based on the circuit proposed in [6] draws different currents when implementing different functions. Without a need for advanced algorithms, by comparing the current drawn from the LUT under test with standard measurements, the functionality of the LUT can be readily inferred. This experiment shows how the obfuscation key can be accessed without executing the SAT-attack.

6.2. P-SCA resilient LUT-based Obfuscation

The primitive proposed in this section builds on top of the LUT-based obfuscation discussed. It still uses the n-LUT$_2$ followed by LUT$_n$ configuration for providing resiliency against SAT-attack, but the architecture of the LUT is modified and it is discussed in the following section.

6.2.1 Symmetrical MRAM-LUT (SyM-LUT)

Generally, M-input LUTs have 2^M memory cells to implement M-input Boolean functions. A select tree MUX circuit is utilized to select the memory cell that holds the correct value of the function implemented by the LUT. This work proposes Symmetric Magnetoresistive Random-Access Memory-based Look-up Table (SyM-LUT) design, where the memory storage element is re-designed and optimized for increased security.

The memory storage in this design uses complementary MTJ memory cells, which helps in realizing different states of memory while maintaining equal overall resistance for determining states "0" and "1". Moreover, the proposed SyM-LUT provides a reliable and energy-efficient operation

due to a wide read margin.

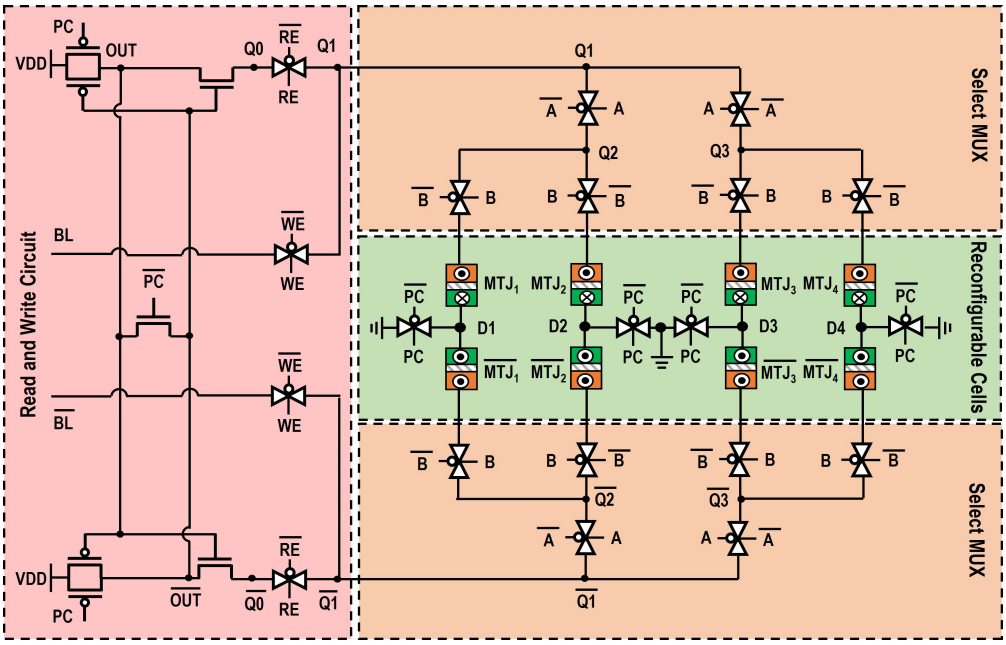


Figure 6.2. The circuit-level diagram of the proposed 2-input SyM-LUT using STT-MTJ devices.

Figure 6.2 depicts a 2-input example of SyM-LUT design. As shown in Figure 6.2, SyM-LUT contains two select tree MUXes, which use Pass Transistors (PG) and Transmission Gates (TG)s. The proposed SyM-LUT can be reconfigured using the WE, and $\overline{\mathbf{WE}}$ signals to perform a write operation. When **WE** and $\overline{\mathbf{WE}}$ are asserted, each memory cell can be connected separately to the Bit Line (BL). **BL** and $\overline{\mathbf{BL}}$ connect to the specific memory cell using inputs **A** and **B**. By setting **BL** and $\overline{\mathbf{BL}}$, one can change the content of the memory cells.

Additionally, in each write operation, the content of each memory cell is changed in a complementary fashion. As a result, $\mathbf{MTJ}_i$ and $\overline{\mathbf{MTJ}_i}$ always hold opposite values. In particular, assuming the data stored in the $\mathbf{MTJ}_1$ is in the P or low-resistance state, then $\overline{\mathbf{MTJ}_1}$ will be in the AP or high-resistance state and vice versa.

After termination of the write operation, the **PC** signal is asserted to pre-charge the intermediary output nodes **OUT** and $\overline{\mathbf{OUT}}$, and then by disabling the **PC** signal and asserting the Read Enable (RE) and $\overline{\mathbf{RE}}$ signals, the discharge path is enabled and read the data stored in the MTJs. Signals **RE** and $\overline{\mathbf{RE}}$ enable the read path from **OUT** and $\overline{\mathbf{OUT}}$ to **GND**. This will result in a race condition between the two branches of sense amplifier, which will be used the observe the resistance difference between the $\mathbf{MTJ}_i$ and $\overline{\mathbf{MTJ}_i}$. The select tree MUXes controls the output of LUT according to the

Table 6.1. Parameters of 2-terminal STT-MTJ device.

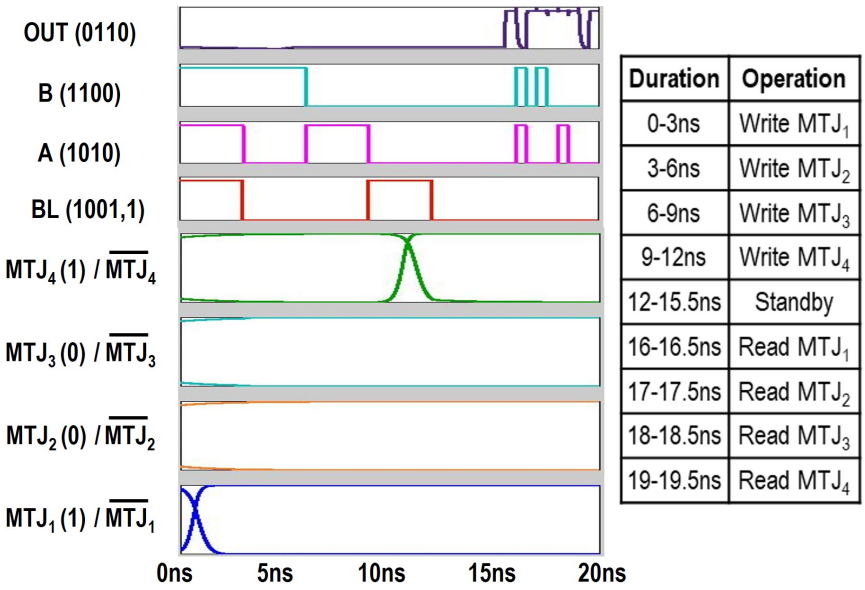
Parameters	Description	Value
MTJ_{Area}	$l_{MTJ} \times w_{MTJ} \times \pi/4$	$15nm \times 15nm \times \pi/4$
t_f	Free Layer thickness	1.3 nm
RA	MTJ resistance-area product	$9\ \Omega \cdot \mu m^2$
T	Temperature	358 K
α	Damping coefficient	0.007
P	Polarization	0.52
V_0	Fitting parameter	0.65
α_{sp}	Material-dependent constant	2e-5

input signals **A** and **B**. Thus, when storage elements are connected to the bit line, the voltage at the junction of two MTJ's provides enough margin to distinguish between two states i.e., "0" and "1" stored in each complementary MTJ storage allowing for a reliable read. This value of the LUT function is observed at the output nodes **OUT** and $\overline{\textbf{OUT}}$. In this manner, the LUT provides a wide margin while maintaining a similar power profile while reading "0" and "1".

To define the functionality of each 2-input SyM-LUT, a set of keys are shifted in via the Bit line **BL** signal, and by controlling **A** and **B** inputs, memory cells whose contents need to be updated can be selected. For example, for the "AND" function, **A** and **B** inputs are used to select each memory cell in the order of AB, such as 11, 10, 01, and 00, while the keys to configure the functionality of the LUT are shifted through the **BL** as 1, 0, 0, and 0, respectively.

HSPICE circuit simulator is used to validate the functionality of the proposed SyM-LUT using 45nm CMOS technology and the STT-MRAM model developed in [40]. The states of the MTJ are determined according to the angle, θ, between the magnetization orientation of the ferromagnetic layers. This work have adopted the MTJ device parameters from [6]. Table 6.1 lists the experimental parameters used herein to model the MTJ devices. Figure 6.3 shows the transient response of the proposed SyM-LUT while the Figure 6.3 depicts an implementation of a 2-input "XOR" gate utilizing SyM-LUT. As shown, the HSPICE simulations verify the correct functionality of the proposed SyM-LUT.

Furthermore, this work performs Monte Carlo (MC) simulation to analyze the reliability of reading and write operations of SyM-LUT in the presence of PV. The simulation helps cover a wide range of PV scenarios that may occur in the fabricated device. The MC simulation is performed with $10,000$ instances considering the effects of PV on the CMOS peripheral circuit and the MTJs. In particular, the variation of 1% for the MTJ's dimensions along with 10% variation on the threshold



Duration	Operation
0-3ns	Write MTJ_1
3-6ns	Write MTJ_2
6-9ns	Write MTJ_3
9-12ns	Write MTJ_4
12-15.5ns	Standby
16-16.5ns	Read MTJ_1
17-17.5ns	Read MTJ_2
18-18.5ns	Read MTJ_3
19-19.5ns	Read MTJ_4

Figure 6.3. Simulation waveform for implementation of a 2-input XOR gate using SyM-LUT.

voltage and 1% variation on transistors dimensions are assessed [6]. According to the MC simulation results, SyM-LUT provides reliable write performance resulting in less than 0.0001% write errors in 10,000 error-free MC instances. Additionally, since the states of the MTJs are complementary, they provide a wide read margin, and as a result, there are less than 0.0001% read errors caused by PV based on the 10,000 error-free MC simulation results for all the different gates that implemented using SyM-LUT.

6.2.2 SyM-LUT's Resiliency against P-SCA

The side-channel analysis focuses on the variation in the electrical characteristics of the circuit to gain access to the confidential information in the circuit. The power traces are targeted to measure the read current of the LUT under test. By varying the input of the LUT, the select tree selects various MTJs for sensing their content. The various state of the MTJs result in different currents. The MC simulation data from HSPICE provides the values of the MTJ read current in the presence of PV. The threat vector used in this work assumes that the attacker can use an invasive approach to probe the current.

If the SyM-LUT is representing "XOR" gate, as shown in Figure 6.3, 4 different read current measurements for reading four different states of the SyM-LUT (i.e., read current measurement

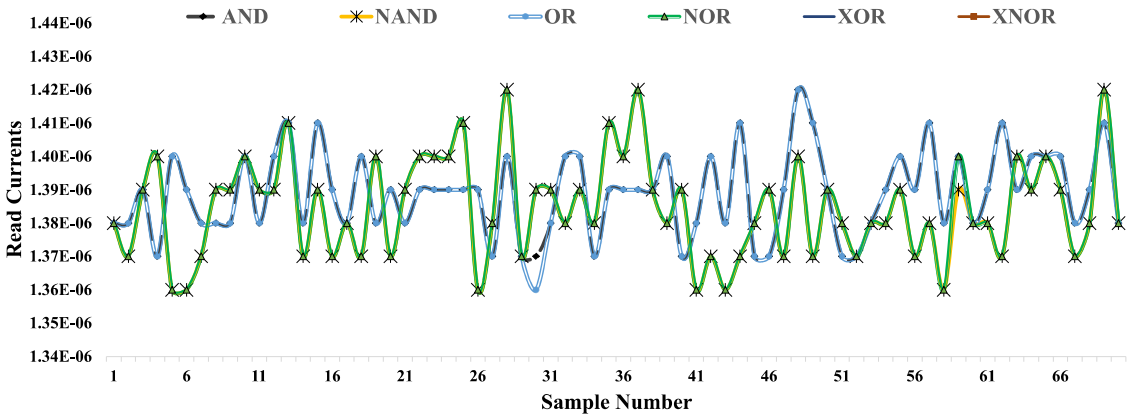


Figure 6.4. Read current trace samples of 2-input SyM-LUT implementing various functions using Monte Carlo instances.

when (1)A=0, B=0; (2)A=0, B=1; (3)A=1, B=0; (4)A=1, B=1) can be measured. Among 4 different states, the current measurement for MTJ with content "0" and the current measurement for MTJ with content "1" should show a difference for the P-SCA to succeed in learning the content of the MTJs. Using MC simulation with 10,000 instances for each logic gate implemented, power trace and current values for all MTJs are gathered. Figure 6.4 demonstrates the read currents of the MTJs from the MC simulation. The same trend is observed over the rest of the simulation generated to evaluate of the proposed LOCK & ROLL approach. As shown in Figure 6.4, the contents of the MTJs cannot be easily distinguished, which is not the case in the traditional implementation of the LUT. Therefore, to empirically assess the security of the proposed primitive, this work employs ML-assisted techniques to reverse-engineer the contents of the MTJs.

As a part of ML-assisted techniques for reverse engineering, a classification model using Random Forest, Multi-class Logistic Regression, and Support Vector Machine (SVM) is used. For this experiment, the 4 traces of power for reading different states for 16 logic functions implemented by LUT size 2 is used. The LUT of size 2 has two inputs and the features used for ML are Read Power when the two inputs are (1)A=0, B=0; (2)A=0, B=1; (3)A=1, B=0; (4)A=1, B=1.

For Multi-Class Logistic Regression, polynomial features of degree 4 are used for fitting. To avoid overfitting, the data lasso regularization is used. Moreover, in Logistic Regression, the Multi-Class Cross-Entropy is used as a Loss function. The number of classes is 16 for LUT of size 2. In Random Forest, **entropy** is used as the criterion to determine the quality of the split. In the case of the SVM Radial Basis Function (RBF) is used as a kernel function. For the empirical evaluation

Table 6.2. Performance of ML-assisted P-SCAs on SyM-LUT

Algorithm	Accuracy	F1-Score
Random Forest	31.55%	0.319
Logistic Regression	30.75%	0.304
SVM	28.09%	0.302
DNN	34.9%	0.343

of the SyM-LUT, a total of 640,000 different samples using MC simulation for 16 class labels are generated, and 10-fold cross-validation techniques along with accuracy and F-1 score as a metric are used to evaluate the performance of the ML algorithm. For data pre-processing, feature scaling and outlier filtering using z-scores are used. Table 6.2 shows the resiliency of the proposed primitive to thwart P-SCAs.

Proposed mechanism is evaluated against P-SCAs that are assisted using the Deep Learning (DL) techniques. A Deep Neural Network (DNN) is used to classify the functionality implemented by the LUT using the power traces obtained from the HSPICE simulation. The work in [41] has shown that DNN can bypass misalignment countermeasures in the ML-assisted P-SCAs. The input to the DNN is the scaled power trace vector with the value ranging from 0 to 1 for better convergence. The output layer of the architecture uses a softmax activation with a categorical cross-entropy as a loss function. The softmax activation provides a probability distribution over all possible functions implemented by the LUT. The DNN architecture uses the fully-connected layers with the Relu activation function and Adam optimizer for training the model. The 640,000 data traces are used for training, and the model is evaluated using 10-fold cross-validation. The accuracy of the DNN model was ∼35%.

All models have more than 90% classification accuracy on traditional LUT-based architectures. However, as soon as the same architecture of the ML model is used to reverse engineer SyM-LUT, it fails to classify or learn the functionality implemented by the LUT. Even upon re-training the model with read currents from SyM-LUT's MC simulation, the model shows no success in reverse engineering the content of the LUT.

Thus, the inability of the models to distinguish between the states of MTJs proves the ability of the SyM-LUT to mitigate the P-SCA. Moreover, the attacker will need to obtain a training set initially, which can be a challenge. A symmetrical circuit design implementing two identical select

tree MUXes that minimize power consumption through circuit optimization and complimentary MTJs results in robust security primitive. Thus, the outcome of the experiments proves that SyM-LUT maintains a near-zero power variation in the output and thus is resilient to P-SCAs.

6.3. Proposed LOCK & ROLL

6.3.1 Scan-Enable Obfuscation Mechanism

As demonstrated above, SyM-LUT can be a great candidate to resist P-SCA attacks. Pairing SyM-LUT with LUT-based obfuscation discussed in this work [4] makes it SAT-resilient. However, to eliminate the threat of SAT-attack, this work adds the Scan-Enable Obfuscation Mechanism (SOM), which aims to provide a multi-layer defense mechanism. It enables the circuit to eliminate the threat of SAT-based attacks and its derivative along with other powerful attacks, such as Scan attacks and removal attacks, to name a few. This work first discusses how SAT-attack can be eliminated, followed by how the proposed solution provides resiliency against other evolving attack vectors. Figure 6.5 depicts a 2-input example of SyM-LUT design with SOM.

As depicted in Figure 6.5, once the write operation is terminated, by asserting the **RE** and $\overline{\textbf{RE}}$ signals, one can read the data stored in the MTJs. The read operation of the SyM-LUT with SOM is similar to SyM-LUT, and the value of the LUT function is observed at the output nodes **OUT** and $\overline{\textbf{OUT}}$, as shown in Figure 6.5.

The scan chain locking in the proposed primitive is designed to offer resiliency to both P-SCA and SAT-attack. The SyM-LUT features the SOM circuitry, which is activated when the scan chain is enabled. The SAT-attack uses the scan chain to scan in the input to the oracle circuit and scan out the oracle response. Scan Enable (SE) is activated when the scan chain is enabled. The output of the LUT at this point will be affected by the data stored in $\textbf{MTJ}_{SE}$. The $\textbf{MTJ}_{SE}$ bits are configured to either "0" or "1" at random. These values are known to the trusted IP owner; however untrusted entities do not know these values. Since the value in $\textbf{MTJ}_{SE}$ is stored randomly, not all circuits will be provide the same output if **SE** is enabled. During the read operation using scan-chain, whenever the **SE** is asserted, the data stored in $\textbf{MTJ}_{SE}$ and $\overline{\textbf{MTJ}_{SE}}$ will determine whether the actual function makes it to the **OUT** or the random value in $\textbf{MTJ}_{SE}$ makes it to the **OUT**.

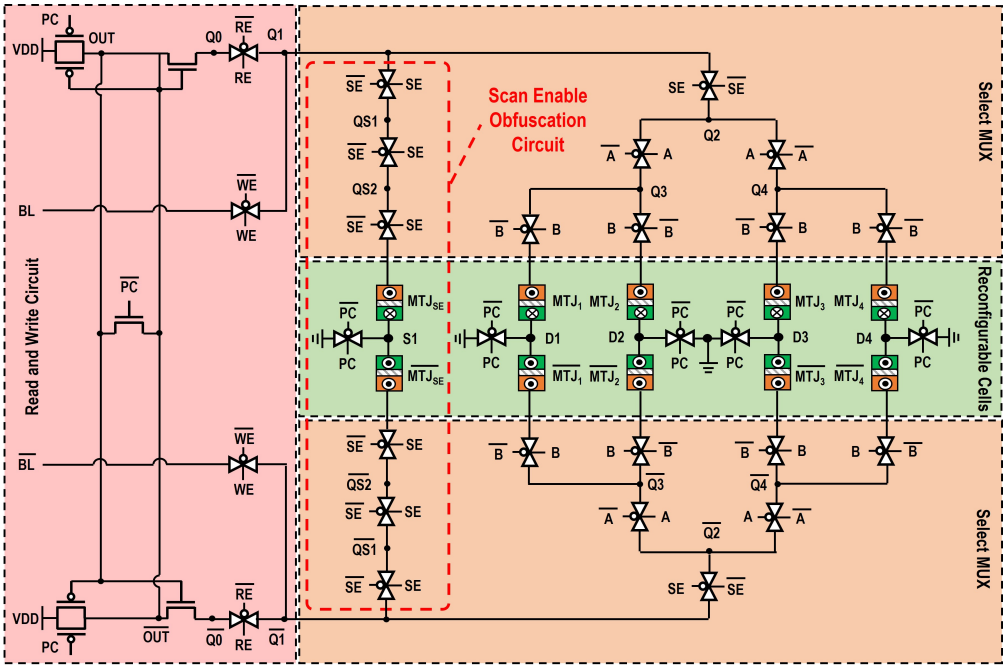
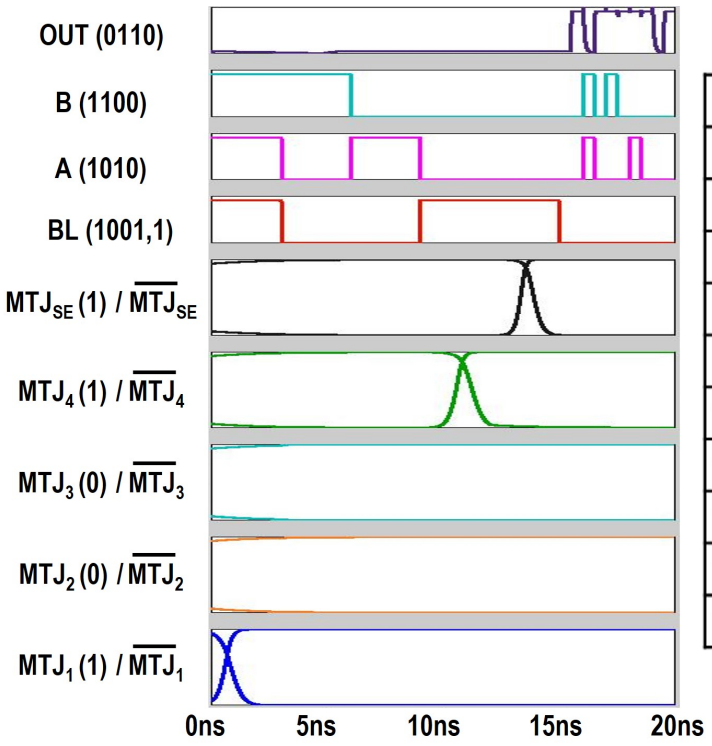


Figure 6.5. The circuit-level diagram of the proposed 2-input SyM-LUT with SOM using STT-MTJ devices.

Similar to SyM-LUT, the HSPICE circuit simulator is used for validating the functionality of the proposed SyM-LUT with SOM using 45nm CMOS technology and the STT-MRAM model used in [6]. Figure 6.6 illustrates an implementation of the XOR gate using SyM-LUT with SOM configured to the value of "0". It can be observed in Figure 6.6 that the content of the $\mathbf{MTJ}_{SE}$ is updated to provide the obfuscated output. Additionally, a MC simulation study is performed to examine the reliability of the proposed SyM-LUT with SOM in the presence of PV. Similar to the SyM-LUT, SyM-LUT with SOM also maintains error-free write and read operations. SyM-LUT demonstrates less than 0.0001% write errors and less than 0.0001% read errors caused by PV based on the 10,000 error-free MC simulation results for each gate implemented using SyM-LUT with SOM. The Sym-LUT with SOM also exhibits the same current trace, as shown in Figure 6.4. Analysis using ML techniques is performed to infer the functionality implemented by Sym-LUT with SOM, and the results depicted in Table 6.3 show that the functions implemented by the LUTs cannot be distinguished.

The experimental setup to verify the resiliency against P-SCA is the same as that of SyM-LUT. Table 6.2 shows the resiliency of the proposed primitive to thwart the P-SCA. For the evaluation, decision tree and the random forest model was used. 10-fold cross validation was used in the



Duration	Operation
0-3ns	Write MTJ_1
3-6ns	Write MTJ_2
6-9ns	Write MTJ_3
9-12ns	Write MTJ_4
12-15ns	Write MTJ_{SE}
15-15.5ns	Standby
16-16.5ns	Read MTJ_1
17-17.5ns	Read MTJ_2
18-18.5ns	Read MTJ_3
19-19.5ns	Read MTJ_4

Figure 6.6. Simulation waveform for implementation of a 2-input XOR gate using SyM-LUT with $\mathbf{MTJ}_{SE}$ being set to value of "0".

experiment and the performance was evaluated using accuracy and F-1 scores as a metric. For both tree classifier, various hyperparameter tuning was done, and the best results are presented in the table. For the random forest classifier, the number of estimators was as large as 512 with no bounds on the length of the tree. Considering a similar P-SCA evaluation performed on SyM-LUT, the results prove that SyM-LUT with SOM is also resilient to P-SCAs.

Table 6.3. Performance of ML-assisted P-SCAs on SyM-LUT with SOM

Algorithm	Accuracy	F1-Score
Random Forest	31.6%	0.322
Logistic Regression	30.93%	0.310
SVM	26.36%	0.284
DNN	35.01%	0.357

6.3.2 Elimination of SAT Attack and It's Derivatives!

The SAT-attack is thwarted because the oracle responses are incorrect when the DIPs from the SAT-attack are applied to the oracle circuit. The keys deduced using these responses will be wrong,

and the attacker won't be able to reverse engineer the functionality of the obfuscated IP. Compared to [36, 42], which also thwarts the SAT-attack using their dynamic nature of obfuscation, the SyM-LUT provides better applicability. The primitive in [36, 42] can only be used in the IP that can tolerate a few error margins; however, the LUTs are static in the proposed work. Thus the circuit works error-free during the normal functioning and allows the primitive to be used widely for obfuscation. Moreover, in the proposed primitive, the LUT itself creates the SAT-hard instance despite its location in the circuit [4], thus not requiring the IP designer to rely on the heuristics approaches. The IP owner can identify the circuitry they want to secure and insert the SyM-LUT with SOM.

6.3.3 Security Coverage of LOCK & ROLL

Since the LOCK & ROLL builds on top of the proposed LUT-based obfuscation, it provides security against many SAT-attacks and its derivatives, as discussed in 5.3.3. In this section, the work focuses on how LOCK & ROLL can be used to increase the security of the design against various other infamous attack vectors.

The HackTest [43] attack can reverse engineer the obfuscated circuit by utilizing the ATPG test vectors which the IP manufacturer provides to the testing facility. The ATPG test vectors are usually generated to offer the highest level of fault coverage. Using the testing data provided, the actual response of the oracle IP upon application of the test vectors can be recorded, and the functionality of the obfuscated gates can be deduced/resolved. The SyM-LUT-based obfuscation and its dynamic nature to morph the functionality based on the contents of the MTJ can circumvent the HackTest attack. To protect against HackTest, the fabricated IP is programmed with the keys, K_d, which are different from the original intended keys, K_o. The ATPG test patterns are generated using K_d and given to the testing facility. Moreover, testing the IP does not require the IP to be functional [43, 44]. The test patterns are generated for key K_d such that it provides maximum fault coverage and the ability to test the IP for any faults. Once the IP is tested and returned to the trusted regime, the LUTs can be configured using the correct keys K_o.

Another attack, i.e., ScanSAT, tries to model the obfuscated scan chain as a logic locking problem for leveraging the SAT-attack. In the proposed SyM-LUT primitive, when the Scan chain is activated, the Scan Lock circuitry is activated and now becomes part of the circuit. When the

resulting circuit is modeled using the SAT-attack [11], the circuit is originally SAT-hard due to LUT-based obfuscation. Moreover, this work has discussed how the SOM thwarts the SAT-based attack in Section 6.3. The secure cell saves the key value for activating the IP and the values from the secure cell can be propagated using the methodology discussed in the Scan and Shift attack. For programming, in the proposed the SyM-LUT, a separate scan chain is used. The keys are shifted in for the configuration of various SyM-LUT blocks, and the scan-out port of this entire scan chain is blocked. This prevents the attacker from reading the values of MTJ during the writing stage. In the proposed primitive, the MTJs are non-volatile, and thus the programming of the MTJs using this entire chain will only be performed in the trusted regime, thus mitigating the threat of Scan and Shift attack. The structural analysis of the LUTs yields no concrete information that can help in eliminating the LUTs from the circuit. Thus unlike other obfuscation, the proposed primitive is resilient to the removal-based attack. In this manner, SyM-LUT with SOM offers a multi-layer defense mechanism to provide resiliency against various attacks.

The LUT-based obfuscation in this way can thwart many attacks. In the following sections, this work discusses the methodology to validate the security and functionality of the obfuscation, as validation is one of the essential steps of Physical Design while fabricating an IP.

Chapter 7

Security Validation of an Obfuscated IP

The current practice of evaluating the obfuscation resiliency against the SAT-attack consists of simulating the attack for an indefinite time until it de-obfuscate the design. This simulation time can be several days, weeks, or even months until the SAT-attack can de-obfuscate the design. Given the uncertainty of de-obfuscation time using the SAT-attack, the simulation method for assessing and quantifying the security is not practical. It can introduce an indefinite delay in the IP design flow.

This chapter discusses the methodology that can be leveraged instantaneously to validate the security added by the LUT-based obfuscation. This will ease the DSE process performed as part of optimization during physical design flow, thereby making LUT-based obfuscation a convenient method for restoring the trust in silicon.

7.1. Quantifying Security

This work has conducted several experiments to understand the effect of SAT-attack on various gate replacement policies and obfuscation. The observations are summarized as follows: a) the de-obfuscation time of the SAT-attacks highly depends on different parameters such as the location of the obfuscated gates, type of obfuscated gates, obfuscation coverage or key size, gate mutability, and the topology of the underlying circuit [11]; b) verifying whether obfuscation is SAT resilient or not might take an undetermined amount of time, especially when employing SAT-hard obfuscation techniques. In most cases, experiments for validating SAT-attacks are run for a couple of days, but

in this work, the runtime of simulations is extended for as much as 5 days for better evaluation of the obfuscation.

Numerous defense mechanisms exists for an IP owner who wants to protect his IP. As the defense mechanisms' success depends on the underlying characteristics of the IP being secured, evaluating and validating the security of the IP after the integration of obfuscation primitive is a highly crucial step. However, security evaluation against a powerful SAT-attack can take a few days to months using a contemporary method of simulating the attack. Therefore, a mechanism that can instantaneously assess the strength of different obfuscation schemes against various attacks is the need of the hour.

This work focuses on the SAT-attack because the SAT-attack can be used to break many of the existing obfuscation techniques [14, 45, 46]. Other methods such as removal attacks and structural attacks cannot be quantified. They either succeed or fail, and the success of such primitives mostly depends on the expertise of the attacker. Whereas, in the case of the SAT-attack, it is widely available to deploy and relatively easy to use. Therefore, an IP owner can not overlook the possibility of their designs being subjected to the SAT-attack.

Even in the case of Scan-based circuitry, i.e., SyM-LUT with SOM failing, there must be a clear metric on how much security the LUT-based obfuscation can provide before the content of the LUT is stolen.

Many of the proposed schemes demonstrate the efficacy of their obfuscation scheme by simulating the attacks for a few hours to a few days [32, 45, 47, 48]. Obfuscation schemes such as SFLL-HD^0 [32] are more sophisticated to study as they have functions like hamming distance, obfuscation coverage, key length, and gate selection algorithm that can influence the SAT-runtime and the designer has a large security design space which they must explore before finding and deploying optimal and secure obfuscation strategy. However, it is not optimal nor feasible for the IP designer to brute-force through all the possible combinations of the parameters that influence the security of the design and evaluate them by simulating attacks for an indefinite amount of time. Therefore, this challenge in the Hardware Security domain calls for a framework that can assess the obfuscation's security accurately and quickly.

Given the dire need for quantifying and evaluating the security, the problem is yet highly under-explored in the community because of the significant challenges listed here:

i *Difficulty characterizing the hidden and sophisticated algorithmic mechanism of attackers.* Over recent years, many de-obfuscation methods have been proposed with various techniques [49] to thwart hardware security threats. These include methods with sophisticated theories, rules, and heuristics, to name a few. The behavior of such highly nonlinear and strongly-coupled systems is prohibitive for conventional simple models (e.g., linear regression and support vector machine [50]) to characterize.

ii *Difficulty extracting determinant features from discrete and dynamic graph-structured ICs.* The inputs for the de-obfuscation time estimation problem are the circuit or IP topology (netlist) with gates selected for obfuscation. Conventional feature extraction methods are not intuitive to apply to such types of varying-structured data without significant information loss. Hence, it is highly challenging to intactly formulate and seamlessly integrate features as mathematical forms that can act as an input to conventional machine learning models.

iii *The Requirement of high efficiency (speed) and scalability.* The ability to perform obfuscation space exploration and deploy a defense depends on the speed and scalability of the framework. The faster the defender can estimate the de-obfuscation runtime, the more candidate set or obfuscation samples can be evaluated and verified against the attack. The ability to weigh in various choices can lead to a better obfuscation scheme being deployed. Moreover, the estimation speed (latency) of the framework must be insensitive to obfuscation strategy and the circuit topology.

To model the effect of obfuscation and address all of the open challenges described above, the work proposes a Security Evaluation Platform for Hardware Logic Obfuscation using Intelligent Artificial Neural Net (SEPIANN) framework. This fundamental framework for de-obfuscation time prediction is based on the CNF representation of the obfuscated circuit. The graph neural network model is trained on instances of the obfuscated circuit to capture obfuscation's trend and effect on the design to be secured. Later, the resulting model is leveraged to extrapolate de-obfuscation time instantly, which, on the contrary, would have taken weeks or months to de-obfuscate. This resulting model allows the IP owner to sweep various obfuscation metrics (such as obfuscation coverage, different obfuscation methods, gate selection) and helps in evaluating the security instantly, thereby eliminating the need to wait for an indefinite amount of time. The SEPIANN, in this manner, can

also be employed as an Electronic Design Automation (EDA) tool for hardware security assessment during IC design and development process.

The contributions of this work are outlined as follows.

- Formulate a graph learning SEPIANN framework to predict the de-obfuscation time through SEPIANN's **CNF-NET**[1], which is a graph-based neural network.
- Automatically extract the properties of the circuit/IP represented in the form of CNF utilizing a multi-order graph learning technique, which is utilized to estimate the de-obfuscation time.
- Design an energy-based neural layer to process varying sizes of graph data. For unifying the dynamic topology of the CNF graph, this work innovatively leverages the energy of a restricted Boltzmann machine to quantify the complexity of the CNF. The bipartivity of the CNF graph is utilized to optimize the computational cost.
- Deploy, assess, and validate the SEPIANN framework on various benchmarks.

As an outcome of the research covered in this chapter, this work has contributed to [51–58].

7.1.1 SAT Resiliency Pit-Fallacies

As aforementioned, most research work claims the resiliency of their proposed obfuscation against the SAT-attacks and the variants of SAT-attack by the method of simulation. Using a subset of small-scale benchmarks, the time-constrained simulation demonstrates the obfuscation's effectiveness. Albeit the trend might look promising in terms of resiliency for the selected benchmarks, the SAT-resiliency does not scale with the number of obfuscated gates or the percentage of obfuscated gates. The SAT-attack runtime is not a simple function of obfuscation percentage. As discussed earlier, the circuit's topology also plays a pivotal role in determining the SAT resiliency for a given obfuscation scheme.

One way to validate SAT resiliency could be to find the number of correct keys for the circuit. For this purpose, ApproxMC3 [59] is utilized, which is the SAT-based tool used to count the number of valid solutions for a given problem in each SAT-attack iteration. It has been demonstrated that more than one set of a key can be valid for a given obfuscated circuit, and the work in [47] has

[1]CNF-NET framework is the principal component of the SEPIANN

demonstrated this effect. Suppose the number of correct keys is less while the total key search space is vast. In that case, the probability of finding the correct key decreases, and one can build the probabilistic model to determine the likelihood of guessing the correct key. The SAT-solver requires pruning additional DIPs to find the keys when only a small subset of keys are valid for a given circuit. However, it is found that using ApproxMC3 [59] to count possible correct solutions is not scalable when the circuit size increases.

To estimate SAT-solver runtime, the work further investigated the SATZilla [60] framework; a portfolio-based SAT selection algorithm that chooses the best SAT-solver given a problem. One of the essential criteria for selecting the SAT-solver is faster execution time. However, the execution time of different SAT-solvers that can be used to solve a problem is unknown to the algorithm. Thus SATZilla initially tries to predict the SAT-solver's performance and runtime. This cost-sensitive classification is used for every pair of solvers in the portfolio, and the solver that receives the highest score is chosen for solving the problem. The framework relies heavily on domain knowledge for this classification task to identify features that characterize the problem instances. The chosen feature must relate to instance hardness and should be relatively cheap to compute [60]. In hardware security, obfuscation strategies have been tailored based on the experiences and rules hand-crafted by domain experts. Therefore, the features extracted are also based on heuristics and a case-by-case obfuscation strategy that may induce bias. Inspired by the SATZilla's cost-sensitive classification as the indicative metric of SAT-solver's performance, this work utilizes the features used in SATZilla. However, instead of using heuristic features for runtime estimation, other features are automatically extracted and learned from the raw input by the Graph Convolutional Network (GCN). The following section introduces the Graph Convolutional Network (GCN) and their deployment for SAT runtime prediction.

7.1.2 Graph Neural Networks

Recently, there has been an increasing interest in applying deep learning for various graph data [61], such as social networks, molecular structure, road networks, and brain connectivity. Many graphs and geometric convolution methods have been proposed recently for modeling graph data. The spectral convolution methods [62, 63] are the mainstream algorithms developed as the graph convolution methods. The graph convolution methods are based on the graph Fourier analysis [64].

Inspired by the polynomial approximation proposed in [65], graph convolutional neural networks (GCNNs) [62] were able to leverage the idea of Convolutional Neural Network (CNN) in dealing with the Euclidean data for modeling graph-structured data. Kipf and Welling proposed GCNs [63], which naturally integrate the connectivity patterns and feature attributes of graph-structured data and outperform many state-of-the-art methods such as GCNN. Another category of graph CNN is a spatial domain method such as Diffusion Convolutional Neural Network (DCNN) [66], which is considered as a diffusion process (random walk) on the graph. DCNN generate different features by applying a diffusion kernel of different size. MoNet[67] also have generalized spectral and spatial methods in the non-Euclidean domains (graphs and manifolds) to learn local, stationary, and compositional task-specific features.

Preliminaries

First, let us introduce the utilized notations for GCN [62, 63]. The graph signals are defined on undirected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where $\mathcal{V}$ is a set of n vertexes, $\mathcal{E}$ represents edges, and $\mathcal{A} = [a_{ij}] \in \{0,1\}^{n \times n}$ is an unweighted adjacency matrix. A signal $x : \mathcal{V} \to \mathbb{R}$ defined on the nodes may be regarded as a vector $x \in \mathbb{R}^n$. Combinatorial graph Laplacian [68] is defined as $\mathbf{L} = D - \mathcal{A} \in \mathbb{R}^{n \times n}$ where D is degree matrix.

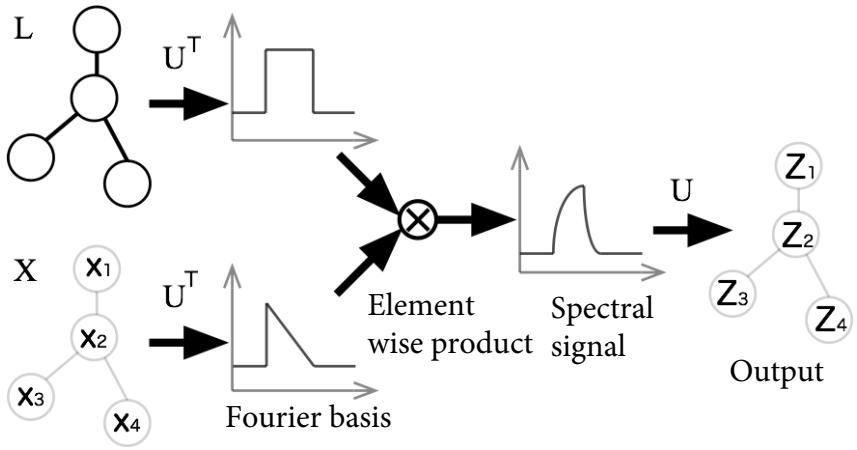


Figure 7.1. Graph Convolutional Network workflow.

The processing of signals in a GCN is performed as follows. The Laplacian is diagonalized by the Fourier basis $\mathbf{U}^\intercal$, so eigendecomposition can be applied as $\mathbf{L} = \mathbf{U}\,\Lambda\,\mathbf{U}^\intercal$ where Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, i.e., $\Lambda_{ii} = \lambda_i$. The

graph Fourier transform of a signal $x \in \mathbb{R}^n$ is defined as $\hat{x} = \mathbf{U}^\intercal x \in \mathbb{R}^n$ and its inverse as $x = \mathbf{U}\,\hat{x}$ [64, 69]. To enable the formulation of fundamental operations such as filtering in the vertex domain, the convolution operator on the graph is defined in the Fourier domain such that $f_1 * f_2 = \mathbf{U}\,[(\mathbf{U}^\intercal f_1) \otimes (\mathbf{U}^\intercal f_2)]$, where $\otimes$ is the element-wise product, and f_1/f_2 are two signals defined on vertex domain. The intuitive workflow of GCN is shown in Figure 7.1. It follows that a vertex signal $f_2 = x$ is filtered by another spectral filter which is defined as $\hat{f}_1 = \mathbf{U}^\intercal f_1 = \mathbf{g}(\mathbf{\Lambda})$ as:

$$\mathbf{g} * x = \mathbf{U}\,[\mathbf{g}(\mathbf{\Lambda}) \odot (\mathbf{U}^\intercal f_2)] = \mathbf{U}\,\mathbf{g}(\mathbf{\Lambda})\,\mathbf{U}^\intercal x.$$

Note that a real symmetric matrix $\mathbf{L}$ can be decomposed as $\mathbf{L} = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^{-1} = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^\intercal$ since $\mathbf{U}^{-1} = \mathbf{U}^\intercal$. Similar works in [62, 65] apply polynomial approximation on spectral filter $\mathbf{g}$ so that graph topology and node attributes such as gate type can be combined in spectral-domain by convolutional operation.

$$\begin{aligned}
&\mathbf{g} * x = \mathbf{U}\,\mathbf{g}(\mathbf{\Lambda})\,\mathbf{U}^\intercal x \\
&\approx \mathbf{U} \sum_k \theta_k T_k(\tilde{\mathbf{\Lambda}})\,\mathbf{U}^\intercal x && (\tilde{\mathbf{\Lambda}} = \frac{2}{\lambda_{max}}\mathbf{\Lambda} - \mathbf{I_N}) \\
&= \sum_k \theta_k T_k(\tilde{\mathbf{L}}) x && (\mathbf{U}\,\mathbf{\Lambda}^k\,\mathbf{U}^\intercal = (\mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^\intercal)^k)
\end{aligned}$$

where T_k is the coefficients of Chebyshev approximation. Further, Kipf and Welling [63] provide simplification tricks based on previous works as shown below.

$$
\begin{aligned}
&\mathbf{g} * x \\
\approx &\theta_0 \mathbf{I_N}\, x + \theta_1 \tilde{\mathbf{L}} x && \text{(expand to } 1^{st} \text{ order)} \\
= &\theta_0 \mathbf{I_N}\, x + \theta_1 (\frac{2}{\lambda_{max}} \mathbf{L} - \mathbf{I_N}))x && (\tilde{\mathbf{L}}{=}\tfrac{2}{\lambda_{max}} \mathbf{L} - \mathbf{I_N})) \\
= &\theta_0 \mathbf{I_N}\, x + \theta_1 (\mathbf{L} - \mathbf{I_N}))x && (\lambda_{max}{=}2) \\
= &\theta_0 \mathbf{I_N}\, x - \theta_1\, \mathbf{D}^{-\frac{1}{2}}\, \mathcal{A}\, \mathbf{D}^{-\frac{1}{2}}\, x && (\mathbf{L}{=}\mathbf{I_N} - \mathbf{D}^{-\frac{1}{2}}\, \mathcal{A}\, \mathbf{D}^{-\frac{1}{2}}) \\
= &\theta_0 (\mathbf{I_N} + \mathbf{D}^{-\frac{1}{2}}\, \mathcal{A}\, \mathbf{D}^{-\frac{1}{2}})x && (\theta_0{=}-\theta_1) \\
= &\theta_0 (\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathcal{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}})x && (\text{renormalization:} \tilde{\mathcal{A}}{=}\mathcal{A} + \mathbf{I_N}, \\
& && \tilde{\mathbf{D}}_{ii}{=}\textstyle\sum_j \mathcal{A}_{ij}),
\end{aligned}
$$

the above GCN in matrix form can be rewritten as:

$$
\mathrm{g}_\theta * X \approx (\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathcal{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}) X \Theta
$$

which leads to *symmetric normalized Laplacian* with raw feature. GCN has been analyzed using smoothing Laplacian [70]:

$$
y = (1-\gamma)x_i + \gamma \sum_j \frac{\tilde{a}_{ij}}{d_i} x_j = x_i - \gamma (x_i - \sum_j \frac{\tilde{a}_{ij}}{d_i} x_j)
$$

where γ is a weight parameter between the current vertex x_i and the features of its neighbors x_j, d_i is the degree of x_i, and y is the smoothed Laplacian. This smoothing Laplacian has a matrix form:

$$
\begin{aligned}
Y &= x - \gamma \tilde{\mathbf{D}}^{-1} \tilde{\mathbf{L}} x \\
&= (\mathbf{I_N} - \tilde{\mathbf{D}}^{-1} \tilde{\mathbf{L}}) x && (\gamma = 1) \\
&= (\mathbf{I_N} - \tilde{\mathbf{D}}^{-1} (\tilde{\mathbf{D}} - \tilde{\mathcal{A}})) x && (\tilde{\mathbf{L}} = \tilde{\mathbf{D}} - \tilde{\mathcal{A}}) \\
&= \tilde{\mathbf{D}}^{-1} \tilde{\mathcal{A}} x.
\end{aligned}
$$

The above formula is *random walk normalized Laplacian* as a counterpart of *symmetric nor-*

malized Laplacian. Therefore, GCN is nothing but a first-order Laplacian smoothing that averages neighbors of each vertex. By utilizing graph topology, attributes of the nodes, and graph convolutional networks, the resulting framework can be utilized for security evaluation in the hardware security domain.

Restricted Boltzmann Machine (RBM)

Another problem that needs to be addressed to evaluate the security is that the GCNs require the graph data to be of fixed size or the same structure; however, it is often not feasible in the real world. Two different versions of obfuscated netlist may differ in length, and thus to address this challenge, the work studied the Boltzmann Machines (BM) representation. BMs [71–76] were initially introduced as a general approach for learning an arbitrary probability distribution over binary vectors. Variants of BM include other kinds of variables, such as the restricted Boltzmann machine [77] that have long ago surpassed the popularity of the original BMs. Invented under the name harmonium, Restricted Boltzmann Machines (RBM) are some of the most common building blocks of deep probabilistic models. In particular, the graph structure of the RBM is a bipartite graph with no connection permitted between any variable in the observed layer between any units in the underlying layer. Like the general Boltzmann machine, the RBM is an energy-based model with the joint probability distribution specified by its energy function. The energy-based model provides an aggregation methodology for representing a graph of dynamic size. This energy aggregation using the RBM is used with the GCN in SEPIANN to deal with the varying size of the circuit.

7.2. SEPIANN: CNF-NET and De-obfuscation Time Prediction

To address the challenges of the SEPIANN, the work proposes a CNF-NET based on clause-literal bipartite graph operators. Figure 7.2 shows an overview of the proposed CNF-NET. The architecture of CNF-NET is described here.

7.2.1 CNF Bipartite Graph Representation

First, the converted CNF of an obfuscated circuit is modeled as an undirected and signed bipartite graph, which uses one node type for clauses and the other for literals. This *CNF bipartite graph* is

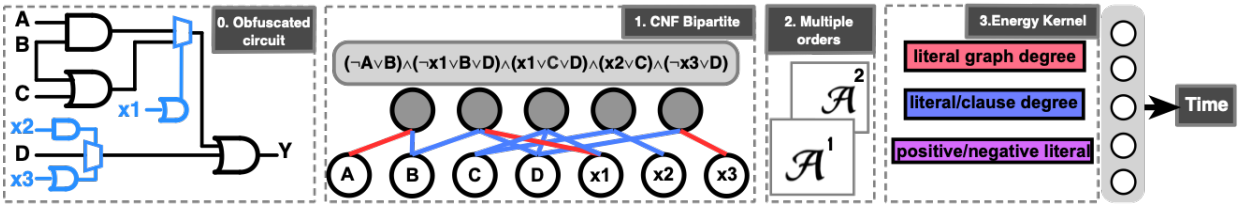


Figure 7.2. Architecture of CNF-NET: 1) extract CNF graph from obfuscated circuit; 2) derive multiple order information from graph representation (section 7.2.1); 3) apply energy kernel to aggregate intermediate features (section 7.2.2); 4) utilize distribution layer on graph properties to sample runtime.

exemplified in Figure 7.3 and defined as $\mathcal{G}(E, V^{literal}, V^{clause})$, where $V^{literal}, V^{clause}$ indicate the set of literal and clause nodes, respectively. The sign of an edge between a literal l and a clause c denotes whether l is negated or not in c. That means the edge value is: (1) (positively connected) if l is in c, and l is positive; (2) -1 (negatively connected) if l is in c, and l is negative, i.e., $\bar{l}$; (3) 0 (disconnected) if l is not in c.

Based on the formulation discussed above, one can denote the i^{th} CNF bipartite graph as $\mathcal{G}_i(E_i, V_i^{literal}, V_i^{clause})$, with adjacency matrix $\mathcal{A}_i \in \mathbb{R}^{N_i \times N_i}$, where $N_i = |V_i^l| + |V_i^c|$ is the total number of literal and clause nodes. Typically, the CNFs for different obfuscations i, j of a given circuit (or across different circuits) are different, leading to $\mathcal{A}_i \neq \mathcal{A}_j$ given $i \neq j$.

The CNF bipartite graph provides a comprehensive representation of the CNF without information loss, thus enabling the end-to-end framework to automatically learn the critical underlying features. Moreover, based on the experiments, the CNF bipartite graph and its multi-order version have been powerful representations in capturing additional meanings/information from the CNF representation of an obfuscated IP. Representation in the CNF bipartite graph allows us to study the effect of previous stages'/logical gate(s) on a given gate. Multi-order expands the number of stages to be considered to represent the circuit effectively. The following section illustrates how to extract the 1^{st} and 2^{nd}-order information, which can be further extended to extract higher-order information. This multi-order information can be used as an immediate representation of an obfuscated IC to explore the patterns associated with the de-obfuscation runtime estimation.

1^{st} and 2^{nd}-order CNF bipartite graph information: The 1^{st} order information is the connectivity between literals and clause, i.e., adjacency matrix $\mathcal{A}$. This helps in capturing the effect of a gate on its children node. Considering the CNF presented in Figure 7.3, 1^{st} order information, i.e., $\mathcal{A}$, is shown on the left side of Figure 7.4, which only has zero values for the diagonal blocks and non-zero

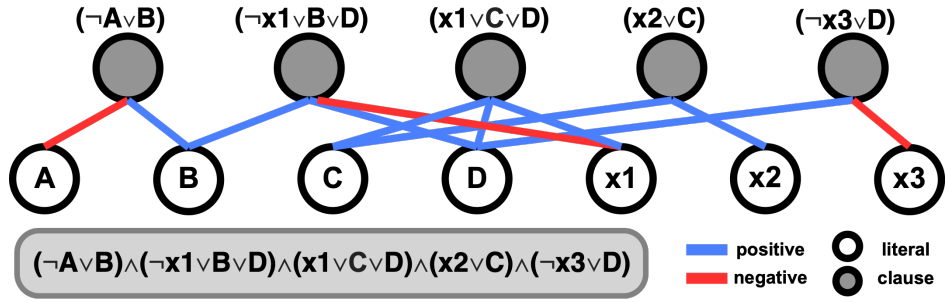


Figure 7.3. An example showing the conversion from CNF to 1^{st} order graph.

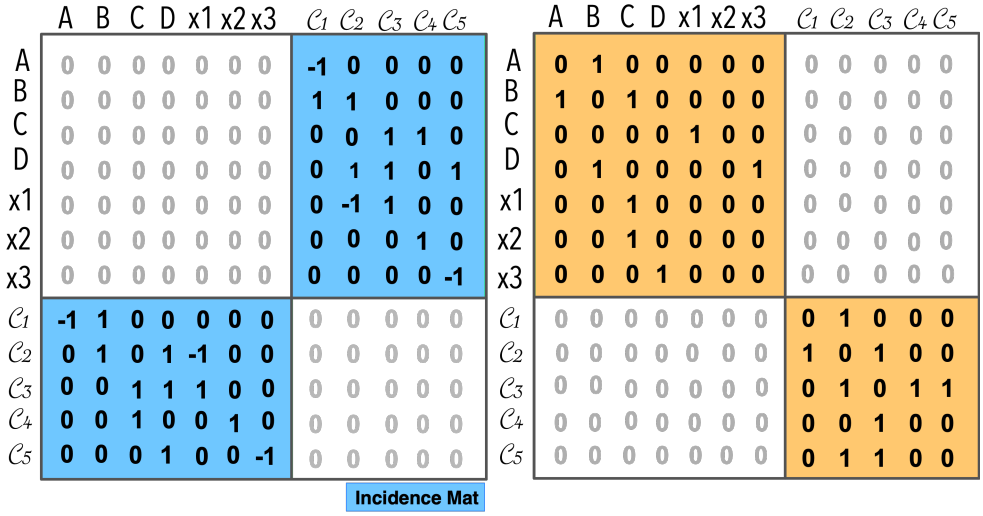


Figure 7.4. A simple example showing (left): adjacency matrix of 1^{st} order graph: $\mathcal{A}$, which models the example in Figure 7.3; and (right): adjacency matrix of 2^{nd} order graph: $\mathcal{A}^2$. There are 5 clauses: clause $\mathcal{C}_1 = \{\neg A, B\}$, clause $\mathcal{C}_2 = \{\neg x1, B, D\}$, clause $\mathcal{C}_3 = \{x1, C, D\}$, clause $\mathcal{C}_4 = \{x2, C\}$, clause $\mathcal{C}_5 = \{\neg x3, D\}$.

values for other parts, each of which is an incident matrix. Since the incidence matrix is symmetric in $\mathcal{A}$, only one blue block is sufficient to represent $\mathcal{A}$ without any information loss, leading to significant computational savings. The 2^{nd} order graph holds 2^{nd} order connectivity, which can be obtained by taking a square of the adjacency matrix, i.e., $\mathcal{A}^2$. The resulting matrix $\mathcal{A}^2$ indicates the connections only between the same type of nodes. As shown on the right side of Figure 7.4, the values in the diagonal blocks are all zero, which is precisely the reverse case of the 1^{st} order adjacency matrix. Therefore, one can refrain from adding the $\mathcal{A}^2$ into the model as the resulting value is redundant. Section 7.2.2 discusses a method that fully utilizes this property to reduce the computation cost dramatically. Note that $\mathcal{A}^N$ is implemented by matrix power: $\mathcal{A}^N = \prod^N \mathcal{A}$.

Odd and even order information: The 3^{rd} order adjacency matrix, i.e., $\mathcal{A}^3$, indicates the connectivity with 3^{rd} order neighbors. The representation also has the non-zero values in the diagonal

blocks while zeros elsewhere (similar to the 1^{st} order adjacency matrix, shown on the left of Figure 7.4). Similarly, the 4^{th} order adjacency matrix is similar to the 2^{nd} order adjacency matrix. This means that a significant amount of computations can be saved when considering higher-order graph information in the same way as discussed in the case of 1^{st} and 2^{nd}-order graphs. There is a trade-off between efficiency and order number, as more orders indicate more information at the cost of additional computational resources. Based on the experimental evaluations and the obtained performances, it is evident that 1^{st} and 2^{nd} order information is sufficient to achieve a reasonably good performance in this work. However, multiple order information could be extracted and embedded for further complex problems. The way to utilize this multi-order information is described below.

7.2.2 Energy-based Operators for CNF Graph

The energy model is used to encode the CNF bipartite graph such that the representation of varying-size CNF bipartite graphs can have a unified dimension for the machine learning technique. Once the multi-order information from the CNF is extracted, the multi-order information is used effectively to learn the mapping from the CNF bipartite graph to the non-deterministic runtime using the proposed framework. However, this task cannot be effectively handled by existing graph classification or regression models because of the input and output's unique properties. Unlike the conventional graphs, the correlation among the neighboring logical nodes in the CNF bipartite graph does not indicate "proximity" or "similarity." Instead, it indicates the logical relation with signed edges in a variable-size bipartite graph. A novel graph encoder layer has been proposed by leveraging and extending the energy of the Restricted Boltzmann Machines (RBM) to address this unique issue. The following subsections expand on the CNF graph'a building process and propose a representation method for obfuscated IP using CNF-based graph.

RBMs for CNF Bipartite Graphs

By innovatively treating the literals and clauses as visible and hidden units, the CNF bipartite graph can be modeled by RBM. The energy of the original RBM is defined as:

$$E(\mathbf{v}, h) = -\underbrace{\mathbf{a}^\top \mathbf{v}}_{\text{visible}} - \underbrace{\mathbf{b}^\top \mathbf{h}}_{\text{hidden}} - \underbrace{\mathbf{v}^\top \mathbf{W}\mathbf{h}}_{\text{interaction}}, \tag{7.1}$$

Where **v** and **h** are the values of visible and hidden nodes, respectively, and **a, b, W** are weights to learn. The first term in equation (7.1) is the energy of visible nodes, the second term is the energy of hidden nodes, and the last term is the interaction energy between visible and hidden nodes. Inspired by the two group modeling, **v** and **h** represent a literal and a clause in the CNF bipartite graph, respectively. Similarly, an energy form is defined for characterizing a CNF: $E = -\alpha \cdot E_{literal} - \beta \cdot E_{clause} - \gamma \cdot E_{interaction}$, where $E_{literal}$, E_{clause}, and $E_{interaction}$ are the energies of literals, clauses, and their connections, while α, β, γ are the weights of them, respectively. Since SAT runtime estimation over CNF is a highly nonlinear process, the traditional linear function has been generalized into a new nonlinear version:

$$E = f_{\Phi}(E_{literal}, E_{clause}, E_{interaction}), \tag{7.2}$$

where f_{Φ} is a neural network function controlled by parameter Φ. The following section provides the study of bipartite connection energy $E_{interaction}$ followed by $E_{literal}, E_{clause}$. Based on RBM, $E_{interaction}$ is defined as a linear function of literals:

$$E_{interaction} = \sum_{m} \sum_{n} v_m w_{m,n} h_n, \tag{7.3}$$

where v_m is a literal and h_n is a clause in one single CNF bipartite graph $\mathcal{G}_i$. However, $E_{interaction}$ is not necessarily a sum function. Therefore, $E_{interaction}$ is obtained by generalizing convolutional graph layers into bipartite graphs.

While most of the existing work in graph deep learning operators focuses on graphs with fixed topology, in this work, the size and topologies of the CNF bipartite graph can vary across different instances dramatically. To solve this problem, a kernel for aggregating interaction information in one graph is designed. Specifically, a d-dimensional vector of pseudo coordinates is associated with $[\mathbf{v}, \mathbf{h}]$. Moreover, this work defines a weighting kernel $Z_{\Theta}(\cdot, \cdot)$, so that for one CNF bipartite graph $\mathcal{G}_i$, it can obtain:

$$E_{interaction} = \sum_{m} \sum_{n} Z_{\Theta}(\mathcal{E}(\mathbf{v}_m, \mathbf{h}_n)) \cdot \mathcal{E}(\mathbf{v}_m, \mathbf{h}_n), \tag{7.4}$$

where $Z_{\Theta}(\cdot)$ projects the $[\mathbf{v}, \mathbf{h}]$ into a new value as the weight of $[\mathbf{v}, \mathbf{h}]$, and $\mathcal{E}(\mathbf{v}_m, \mathbf{h}_n)$ represents the interaction or edge value between $\mathbf{v}_m$ and $\mathbf{h}_n$, which can be 1, -1 or 0. Similarly, $E_{literal}$, E_{clause} are

generalized as:

$$E_{literal} = \sum_m Ent(\mathbf{v}_m) \cdot \mathbf{v}_m, \text{and } E_{clause} = \sum_n Ent(\mathbf{h}_n) \cdot \mathbf{h}_n, \tag{7.5}$$

where $\mathbf{v}$ and $\mathbf{h}$ indicate attributes of literal and clause, respectively, while Ent denotes entropy function. Therefore, the final model for CNF is:

$$\begin{aligned} E = f_\Phi(&\sum_m Ent(\mathbf{v}_m) \cdot \mathbf{v}_m, \sum_n Ent(\mathbf{h}_n) \cdot \mathbf{h}_n, \\ &\sum_m \sum_n Ent(\mathcal{E}(\mathbf{v}_m, \mathbf{h}_n)) \cdot \mathcal{E}(\mathbf{v}_m, \mathbf{h}_n)), \end{aligned} \tag{7.6}$$

Energy Model Multi-Order Graph Operators

Equation (7.6) above does not consider the sign of the edges between literals and clauses. Hence, positive and negative information is encoded separately: $E = \{E^+, E^-\}$. The corresponding incidence matrix $INC \in \mathbb{R}^{|V^{literal}| \times |V^{clause}|}$ is utilized to capture the sign information. In the following section, the normalization is performed to ensure that each instance is comparable and that the learning of the model is not affected by the varying scale of the designs.

Normalized positive and negative edge distribution in clause scope (NPNC): Count positive and negative edges for each clause, and consider the normalization of both positive and negative counts:

$$\left\{ \frac{c^+_{literal}(i)}{c^+_{literal}(i) + c^-_{literal}(i)} \right\}_{i=0}^{|V^{literal}|-1}, \left\{ \frac{c^-_{literal}(i)}{c^+_{literal}(i) + c^-_{literal}(i)} \right\}_{i=0}^{|V^{literal}|-1}$$

which can be obtained by column-wise summation on positive values only and negative only of incidence matrix.

Normalized positive and negative edge distribution in literal scope (NPNL): Similar to the NPNC, positive and negative degrees are counted for each literal, and the normalization per literal is taken. There will be $2|V^{literal}|$ number of features:

$$\left\{ \frac{c^+_{clause}(i)}{c^+_{clause}(i) + c^-_{clause}(i)} \right\}_{i=0}^{|V^{clause}|-1}, \left\{ \frac{c^-_{clause}(i)}{c^+_{clause}(i) + c^-_{clause}(i)} \right\}_{i=0}^{|V^{clause}|-1} \tag{7.7}$$

which can be obtained by row-wise summation on positive value only and negative value only of the incidence matrix.

The 2^{nd} order of the graph with adjacency matrix $\mathcal{A}^2$ denotes the literal-wise and clause-wise mutual information, which corresponds to the top left and bottom right block, respectively, in the right subfigure of Figure 7.4. Their physical meaning is:

- **Literal-wise**: frequency of two literals appearing in the same clause.
- **Clause-wise**: frequency of two clauses sharing the same literal.

Intuitively, whether two literals share the same clause or not is more pivotal than how many times they share. The model leverages the second-order graph information to emphasize this vital trait further and reduce the computational complexity. For example, literal A and B are not connected in the first-order graph (Figure 7.3), but they are linked in the second-order of the graph since A and B share the same clause $\mathcal{C}_1 = \{\neg A, B\}$. This can be obtained by setting all non-zero values of $\mathcal{A}^2$ to 1. The "0/1" value means if two literals appear in the same clause at least once. Therefore, Equation (7.5) is applied to calculate the energy of this graph, and two feature maps are built for the model. Once the circuit transformation is performed, using energy calculation, the CNF-NET predicts the runtime for SAT-attack.

7.3. Summarizing the SEPIANN

To address the challenges discussed earlier, this work proposed the SEPIANN framework. This section summarizes the working of the entire framework. The input to the framework is the obfuscated netlist and label (de-obfuscation time using SAT-attack). This work considered the SAT-attack proposed in [11]. However, any other SAT-attack(s) for de-obfuscation runtime evaluation could be used. For the security evaluation using SEPIANN, SAT runtime is used as the metric for representing SAT resiliency. This process of generating and collecting data, i.e., the netlist and corresponding de-obfuscation time for a given attack, is termed data acquisition. This generated data is further used for training the ML model. Before this data is fed to the model, it requires pre-processing, which involves transformation to CNF and creating the adjacency matrix shown in Figure 7.2. **Thus, the challenge of netlist modeling, which consists of representing the CNF in an intact and structured way for further processing, is addressed in the proposed SEPIANN**. The conversion to CNF captures the vital information of the circuit, i.e., the type of gates and the circuit's topology.

Algorithm 3: CNF-NET

1 CNF-NET $(\mathcal{G}, T_i)$;

Input : $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_N\}$, the real runtime T_i for instance $\mathcal{G}_i$

Output: a neural network function with parameters Φ, Θ, parameters of fully connected layers(τ), and distribution parameter σ

2 Transform from obfuscated circuit $\mathcal{G}_i$ into CNF representation CNF_i and derive bipartite graph; `// Data Preprocessing`

3 Extract adjacency matrix $\mathcal{A}$ from this bipartite graph and calculate power series (multi-scale) : $\{\mathcal{A}^1, \mathcal{A}^2, ..., \mathcal{A}^N\}$;

4 $\theta = \{\Theta, \Phi, \tau, \mu, \sigma\}$;

5 Initialize θ with standard Gaussian;

6 **while** δ *convergences* **do** `// update CNF-NET`

7 Apply the energy kernel on literals and clauses;

8 Apply the energy kernel on the connection between literals and clauses (Eq. 7.4) ;

9 Calculate the overall energy E and then feed E into fully connected layer (Eq. 7.2);

10 Get a intermediate predicted value t, and apply distribution e^t as predicted time;

11 Calculate residues $\delta = T - e^t$;

12 Compute derivatives to update parameters: $\theta \leftarrow \theta + \alpha \nabla_\theta \delta$, where α is learning rate;

13 **end**

The data is sampled and divided into training and testing sets. The ML uses the training set for training the model, while the testing set is used for model validation. After data is generated, **the proposed SEPIANN leverages a CNF-NETwork (CNF-NET) to overcome the challenge of learning the mapping of CNF along with distribution kernel to model the non-deterministic behavior of SAT-attack**. CNF-NET is the proposed cardinal part of the SEPIANN framework, which builds the model for predicting the de-obfuscation runtime. Algorithm 3 presents the pseudo-code of the CNF-NET in the proposed SEPIANN framework. Initially, the obfuscated circuit is represented as an adjacency matrix (line 2). The intermediate output of this operation is the CNF bipartite graph, as introduced in Section 7.2.1.

Following this, multiple order information is extracted from the CNF bipartite graph. By lever-

aging the bipartite graph's property, the computational cost is significantly reduced without loss of generality. The first two orders of the CNF bipartite graph are considered in this model, and it is easy to extend to any order. Using the odd order information from the CNF bipartite graph results in an incidence matrix, which is less than 25% of the adjacency matrix of the CNF bipartite graph. However, one can also use even order information and save space and computation cost as discussed in Section 7.2.1. This step significantly reduces the computational cost compared with typical graph neural networks. **This strategy improves the efficiency of the SEPIANN such that it can be scaled for larger circuits and for large de-obfuscation time estimation.**

After extracting the raw features of the *CNF bipartite graph*, an energy-based [78] kernel, an innate part of CNF-NET, is proposed in Section 7.2.2, used to model the dynamic-size data. This new kernel calculates the energy, which identifies the complexity of the corresponding CNF bipartite graph. **This assists the SEPIANN in modeling the non-deterministic property of SAT runtime to produce the final output.** To handle the dynamic size of CNF, the energy concept from RBMs is employed to aggregate literal and clause distribution into the fixed dimension. The energy aggregation is treated as a features of the targeted obfuscated IC. A distribution kernel is applied in the last layer to model the runtime variance for different instances. This distribution kernel can be replaced with other suitable distributions, such as the logarithmic or exponential kernel.

The final model generated using this process can be deployed to predict the runtime for the newly generated netlist. Algorithm 1 presents a case study using an exponential distribution. Then specific parameter optimization (i.e., Adam) of DNN is employed. Based on the built model, during the inference, the de-obfuscation time for a given obfuscation scheme and a given SAT-attack (or its variant) can be predicted instantaneously.

Using SEPIANN, the prediction takes less than a few seconds, enabling us to perform DSE more efficiently and enabling the defender to deploy better obfuscation. By utilizing graph neural networks, the proposed model derives determinant features from raw input in a supervised fashion. To perform the runtime prediction for different attacks, one must repeat the data acquisition step and include that information in the training data for inference. The proposed SEPIANN framework is obfuscation and attack agnostic.

7.4. Evaluation of SEPIANN

The input to the framework is the obfuscated netlist and label (de-obfuscation time using SAT-attack). This work considered the SAT-attack proposed in [11] for security validation. However, any other derivatives of SAT-attack(s) for de-obfuscation runtime evaluation could be used. The data for initial training of the model is obfuscated netlist and time taken by the SAT-attack to reverse engineer them. Various benchmarks shown in Table 8.1 are used for generating obfuscated netlist, and they are further obfuscated with LUT-based obfuscation discussed in this work [4]. The size of LUT and obfuscation coverage is varied for generating the initial dataset for training. The total dataset consists of roughly 21000 obfuscated instances. Before this data is fed to the model, it requires pre-processing, which involves transformation to CNF and creating the adjacency matrix. The data is sampled and divided into 80% training and 20% testing sets.

To demonstrate the effectiveness of the SEPIANN for security validation, the Pearson and Spearman coefficients are used. Positive scores of 1 indicate the network's capacity to predict the trend of runtime. While another metric, i.e., Mean Squared Error (SSE), shows the prediction error. For calculating the MSE scores, the log of the runtime is used. From the results, it is evident that the MSE doesn't go beyond ∼3, meaning that the reported runtime has at max the delta of +/- 1000 seconds (15 minutes) while predicting for larger benchmarks. From table 7.1, it is evident that for all the benchmarks, MSE is low, and Pearson and Spearman have a positive correlation.

Table 7.1. Performance of Design Security Analyzer

Benchmarks	MSE	Pearson	Spearman	Benchmarks	MSE	Pearson	Spearman
B01	1.241	0.95	0.97	DES Area	2.81	0.93	0.91
B02	1.26	0.91	0.89	DES Perf	2.38	0.92	0.92
B04	1.30	0.93	0.90	SHA-3	2.54	0.94	0.91
B12	1.8	0.95	0.93	8-bit CPU	3.1	0.91	0.90
AES	2.76	0.94	0.88	32-bit RISC-V	3.30	0.90	0.89

The obtained results also show the effectiveness of the proposed framework in considering and utilizing the graph features. This advantage confirms that the proposed model can learn the features that impact the security metric and mimic the runtime trend. Overall, the performance of CNF-NET is stable across different benchmarks while exhibiting excellent performance in predicting and

capturing the runtime trend. The higher correlation and smaller MSE for the benchmarks like AES, DES, and RISC-V show the scalability of the CNF-NET for larger IP designs.

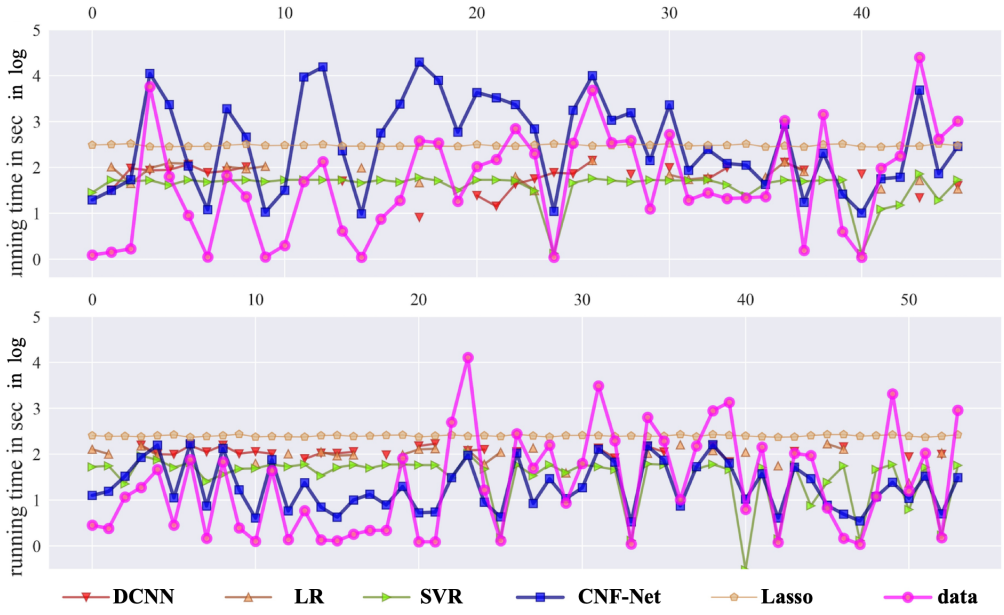


Figure 7.5. Prediction performance on B04 and B12 benchmarks: x-axis is various obfuscated instances and y-axis denotes the predicted runtime compared with real runtime (label of *data*).

The performance of the SEPIANN is evaluated against several state-of-the-art regression models such as Linear Regression (LR), LASSO [79], Epsilon-Support Vector Regression (SVR). Besides, a neural network for predicting runtime work, DistNet [80], is also employed for comparison. Finally, from the state-of-the-art graph deep learning models, DCNN [66] is also chosen for comparison since it can learn graph embedding directly on a set of dynamic-size graphs.

General regression models cannot directly learn patterns on graph data; hence, several predefined features such as the size of the clause, size of literal, the ratio of the clause, and literal are utilized for predicting the runtime. Figure 7.5 shows the performance of various methods on two benchmark, i.e., B04 and B12 for illustration purposes. It is evident that the SEPIANN is able to capture the trend of runtime across various obfuscated IPs efficiently.

As per the discussion earlier, there are many moving targets such as obfuscation coverage, netlist topology, and obfuscation scheme, to name a few, that contribute to the de-obfuscation time of the SAT attack. However, using SEPIANN, one can model the effect of various netlist topology and obfuscation scheme. This work further conducts an experiment to demonstrate the effectiveness of SEPIANN for runtime prediction against different gate placement policies, i.e., (LC_NoGen and

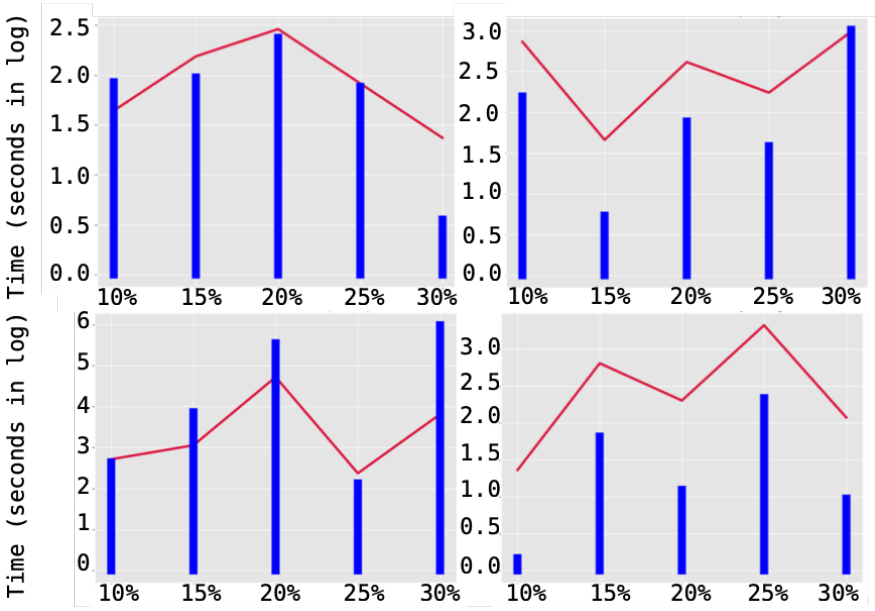


Figure 7.6. Prediction performance on samples from benchmark under different obfuscation policies (*LC_NoGen* and Random), here X-axis denotes obfuscation percentages while Y-axis demonstrate the de-obfuscation time in seconds.

Random). As shown in Figure 7.6, the predicted values (solid red lines) well characterize the trend of test instances (blue bars) throughout different obfuscation policies. Thus, one can predict the outcome of the obfuscation scheme for various benchmarks without the need to simulate the attack.

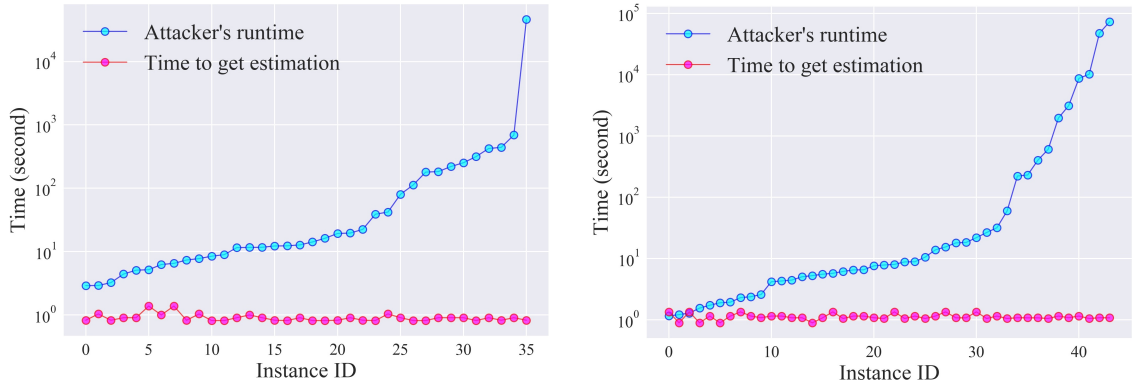


Figure 7.7. Real runtime compared with prediction time on y-axis and different obfuscated instances on x-axis for benchmarks B04 and B12 respectively.

As discussed, another goal of the SEPIANN is to evaluate security instantaneously, accurately, and efficiently. As shown in Figure 7.7, the prediction time of CNF-NET is compared with the real attacker's runtime (i.e., running the SAT-based attack). CNF-NET took a small amount of time (1.0187 seconds on average) after training for predicting the de-obfuscation runtime. This

indicates that with SEPIANN, one can predict security, which in this case is SAT runtime in almost one second as compared to an attack simulation, which can take an uncertain amount of time (to de-obfuscate the design).

Having established the methodology to instantaneously validate the security, in the following section, the work establishes a integration tool flow for adding LUT-based obfuscation in the design.

Chapter 8

Integration of Tool Flow

Following the discussion on validating the security, this chapter proposes the methodology for integrating obfuscation and validating the functionality. This step ensures the smooth integration of the obfuscation in the traditional Physical Design flow. In the proposed technique, LUTs for obfuscation are inserted [4] after the completion of the initial synthesis stage. The proposed strategy only requires predictable and straightforward modifications to the design test methodology. Therefore, the obfuscated design can be moved through the remaining stages of the physical design flow without any additional accommodations while allowing the ability to validate the functionality and security of the design.

Most obfuscation works have presented their results from experimental results. However, this is the first work that validates the claims of LUT-based obfuscation by fabricating the test IC. Additionally, this work also presents a case study for LUT-based obfuscation using both volatile and non-volatile versions. The case study also involves the effect of the obfuscation key sizes on security and design overheads.

In the following sections, the Physical Design flow for LUT-based obfuscation is introduced, followed by the discussion on the functional validation of the IP, and concludes with the results obtained from the fabricated obfuscated IP. As an outcome of the research covered in this chapter, this work has contributed to [4, 25, 26, 51].

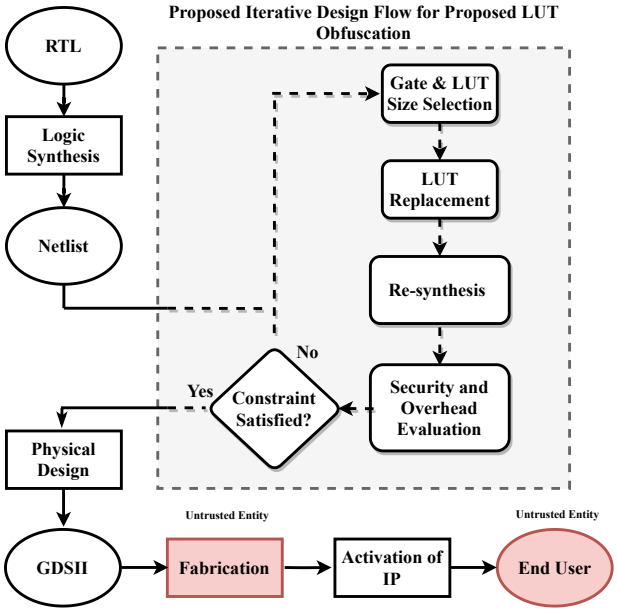


Figure 8.1. Iterative-based Security-driven Design Flow for PPA optimization with optimal security solution.

8.1. ASIC Iterative Security-driven Design Flow

This section provides an overview of the proposed methodology used to obfuscate the design using the proposed LUT-based obfuscation.

For the integration of the STT-LUT, the LUT is defined as the Verilog module, which contains the instances of MTJ or NV latch cell and the RTL-level code of an input multiplexer. When the gates are identified for the obfuscation, they are replaced with the proposed novel LUT block, which is nothing but the new netlist with NV-LUTs containing the RTL code of the multiplexer. Upon insertion of these blocks into the netlist, the firm macros containing the RTL of multiplexers are re-synthesized and optimized for optimizing the overheads. The iterative-based design flow is introduced to optimize the design overheads further while inserting the novel LUT in the security design.

Figure 8.1 illustrates the proposed concept of the iterative security-driven ASIC design flow for overhead optimization[1]. The main aim of this iterative flow is to find optimal gates for obfuscation such that overheads are mitigated. As per the conclusions drawn in the previous chapter, the LUT's size is the most influential factor for SAT-resiliency, and even for a random gate selection, LUT-

[1]gray part in Figure 8.1 is iterative to obtain the best result.

obfuscation results in SAT timeout, as seen in Figure 4.6. This gives the user flexibility to choose gates, as they do not have to abide by a particular obfuscation rule or policy. Even if the gate-selection policy (LC_NoGen) discussed in this work is used for finding the gates for the obfuscation, the proposed flow is non-disruptive to the industrial design flow. In the proposed flow, the netlist is passed as the input along.

Any High-level synthesis tools or RTL-syntheses tools can be used to generate the netlist after the insertion of the novel LUT. In house developed Python scripts are used as a wrapper around industrial tools such as Synopsys DC, PrimeTime, and VCS to automate the flow. The given flow supports the netlist generated with any library and does not restrict the designer in library support. The script first creates a Verilog Module Object (VMO) data structure of the standard cell library, which is then used to define the synthesized netlist abstractly. The result is a complete map of the flattened netlist, defined as interconnected VMO instances.

The script reads a configuration file, which contains the list of the gates to be replaced and the configuration of the LUT to generate the Configurable Logic Elements (CLE). The gates to be replaced are given by the LC_NoGen, or by the user. The configuration of a given LUT is defined by the number of inputs to both the primary and preceding LUT. The generated CLE is added to the standard cell VMO collection and replaces the VMO instances of the gates in the top layer selected for the replacement. The program creates and runs an exhaustive test bench to extract the individual CLE configuration keys in parallel using a logic simulator (i.e., VCS). The output configuration keys are then chained together to form the top-level configuration key. The script finally generates RTL blocks that are inserted within the synthesized netlist, and after a final synthesis stage, the synthesized obfuscated design is provided along with its configuration key.

There are two modes of LUT insertion supported by the script. The first insertion mode assumes that a separate IP block will be included in the design to hold all configuration keys (i.e., an e-fuse, MTJ, or ReRAM macro). In this mode, configuration bits will not be scanned into the LUTs directly instead into the macro, holding the configuration key in a non-volatile state. In this manner, the configuration key being driven by the non-volatile IP are simply top-level inputs to the LUT module. While in the second mode, the LUTs contain the non-volatile bit-cells (including read/write circuitry). In this mode, LUTs have a dedicated scan chain to shift in configuration bits to write to the NV bit-cells.

If the addition of obfuscation modules violates the constraint of the design (i.e., slack violation), then the iterative flow can be leveraged. The iterative flow reports the type of violation. For the timing violation, the LC_NoGen finds new sets of gates such that they are not on the critical path of the design, and timing violations can be removed. For area and power, the iterative design flow requires altering the number and the size of the LUT before creating a new design revision. The gate selection, replacement, and re-synthesis processes are iteratively performed until the PPA constraints, and the security constraints are satisfied. For validating the security instantaneously, SEPIANN can be leveraged. This flow primarily benefits from design-space exploration (DSE) and adds additional overhead in performing the DSE but guarantees optimal design configuration. For the circuit benchmarks obfuscated in this work, at most 4 levels of iteration were sufficient to meet security and design constraints. By excluding the gates on the critical paths, the LC_NoGen finds the gates for obfuscation while eliminating the timing overheads.

To verify logical equivalency between the original target benchmark and the obfuscated version, the Synopsys *Formality* tool is used, which is used for formal verification. Moreover, to avoid missing functional bugs introduced by altering the gate-level netlist, the test benches are also designed such that the logical function provided by the target gates is exercised. In the case of the LUT-mapped design, an initialization task is required to load the configuration key. Once loaded, the original testbench can be run. This initialization-dependent testbench is run after re-synthesis as part of the constraint checking phase depicted in Figure 8.1.

Once the timing, area, and power constraints are met, the traditional ASIC design flow can be utilized before sending the design to the foundry. A dummy LUT configuration key with the test data vector could be provided for the third-party vendor to test the IP. Once the fabricated IP is returned in the trusted regime, one can load the correct configuration key using the scan chain.

8.2. Configuring the LUT Unit Obfuscation Cell

The LUT unit obfuscation cell configuration depends on the logical function the cell is replacing and the input selection. To program the cell, two methods may be used. A dedicated scan chain is used if Configuration Bit (CB)s are to be loaded in externally. CBs may be driven internally from a non-volatile macro cell to bolster security. In this configuration, CBs may be directly driven to their

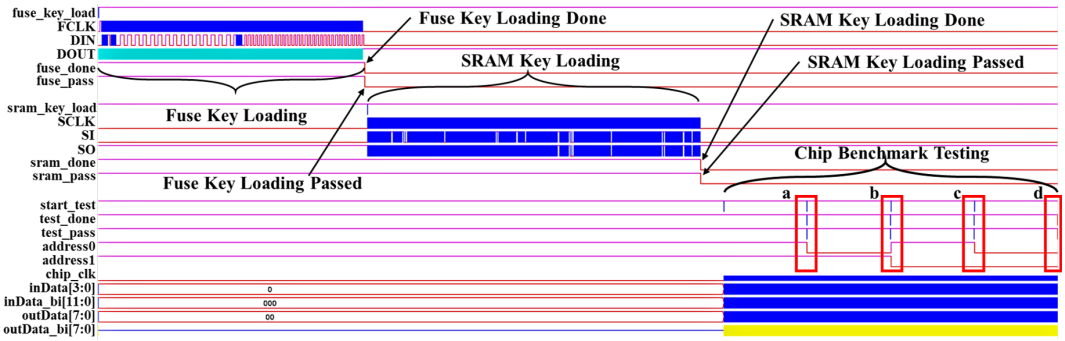


Figure 8.2. Synopsys VCS pre-silicon simulation for verification of obfuscated IP. The simulations include validation of a) original design, b) low, c) medium, and d) high obfuscated versions.

respective LUTs, and programming is performed in the manner required by the non-volatile macro cell. A Python script converts the netlist to a graph of module objects, determines uncorrelated dummy input options, then randomly selects a number of them depending on the obfuscation needs. Then the script replaces the target cells with LUT unit obfuscation cells and connects them to the input list. Synopsys VCS logic simulator is used to generate logical CBs, and the Python script combines them with Python-generated input obfuscation CBs to create the configuration bit-stream.

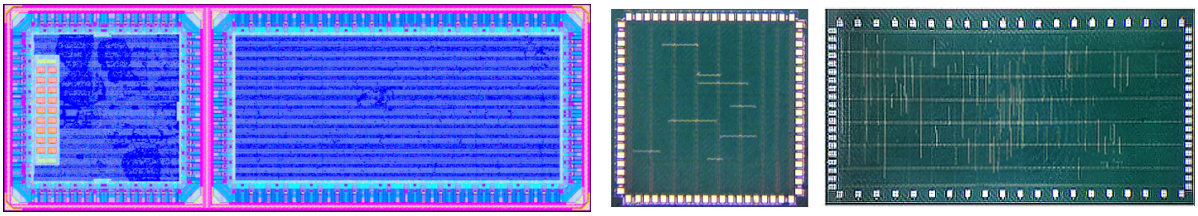
8.3. Validating Obfuscated Designs

A target design is considered in a "locked" state if the proper configuration is not applied. After configuration, the target design will return to its specified functionality. A short programming task is completed, followed by the original design testbench to validate that an obfuscated design is functional. This task may be appended to the initial block of a **Hardware Verification Language** (HVL) testbench for pre-silicon validation. In the case of a scan chain, this task drives the input signals to the CB scan chain, as shown in Figure 8.2. If a non-volatile cell macro is used, the task drives the configuration logic of the macro cell, and upon configuration, the CB contents are driven directly to the CB input of the respective LUTs.

In this work, four different versions of the IP are fabricated, with varying obfuscation coverage, to study the effect of obfuscation on security and overhead. To select the appropriate benchmark, *address*_0 and *address*_1 are set to 00, 01, 10, or 11 for selecting the test chip's original, low, medium, and high obfuscated versions.

8.4. Design and Implementation of Test Chip

After validating the security and the functionality of the obfuscated design, one must perform the post-fabrication functional verification. This experiment demonstrate the process of validating the functionality of the obfuscated design while verifying the claims of LUT-based obfuscation [4] using the data obtained post-fabrication rather than relying on the results obtained from simulation. The functionality test conducted here ensures that the design performs functional when the correct key is applied and enters obfuscated state when the wrong key is used.



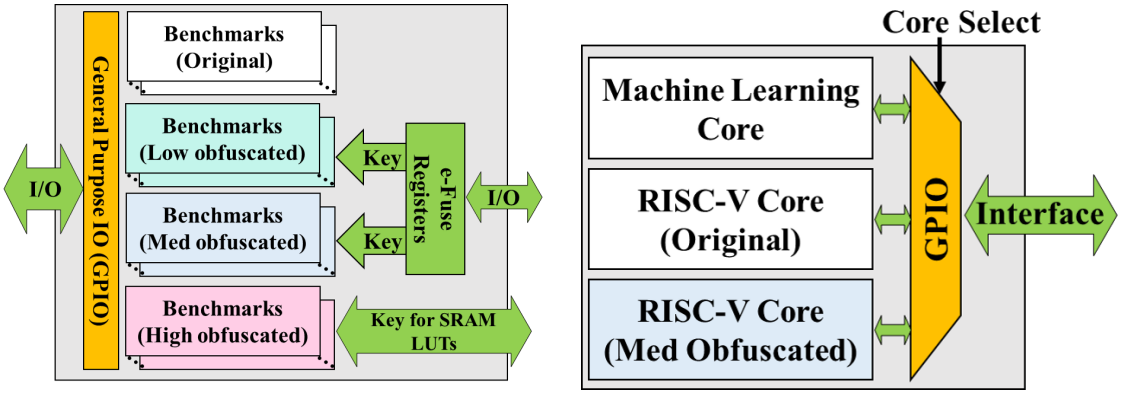
(a) GDSII of chip1 (left) and chip2 (right)

(b) Die photo of chip1 (left) and chip2 (right)

Figure 8.3. (a) GDSII and (b) Die of chip1 and chip2

Two test chips, as shown in Figure 8.3, were implemented and fabricated in TSMC 65nm technology to validate the LUT-based logic locking method. Chip1 contains three popular encryption engines (AES, DES, and SHA-3), a custom ALU, and other benchmarks. Chip2 contains three designs: a logic-locked 32-bit RISC-V microprocessor core and its original counterpart. All benchmarks incorporated in both chips are shown in Table 8.1. LUT-based obfuscation is evaluated by implementing the LUTs in volatile and non-volatile forms while sweeping the length of obfuscation key to provide readers with qualitative and quantitative results. Figure 8.4a shows the architecture of chip1, which contains 10 different benchmarks, each of which has four versions - original, low obfuscated, medium obfuscated, and high obfuscated. The key length in the low, medium, and high obfuscated designs is 288, 576, and 3168 bits, respectively. A single $LUT_8 + 8\times LUT_2$ is used in low obfuscation, while medium obfuscation contains 2, and high obfuscation contains 11 $LUT_8 + 8\times LUT_2$. $LUT_8 + 8\times LUT_2$ refers to the Large LUT of size 8, whose 8 inputs are driven by the small LUT of size 2, as described in [4].

A **General-Purpose IO** (GPIO) block selects one of the designs connected to the top-level IO pins for testing. To demonstrate different key storage methods, the key bits for low and medium obfuscated designs are stored in non-volatile e-fuse registers, whereas those of the high obfuscated



(a) Chip1 architecture

(b) Chip2 architecture

Figure 8.4. Overall Architecture for chip1 (a) and chip2 in (b)

Table 8.1. Benchmarks included in test chip

Source	Benchmark	Description	Total # of cells
OpenCore	DES_area	DES optimized for area	2,085
	DES_perf	DES optimized for performance	15,851
	AES	AES cipher	10,787
	SHA-3	SHA-3 Encryption core (Keccak 512)	13,702
ITC'99	B01	FSM that compares serial flows	34
	B02	FSM that recognizes BCD numbers	26
	B04	Compute min and max	310
	B12	1-player game (guess a sequence)	2656
Custom	ALU	Multiplier/Adder/AND	136
OpenCores	CPU	8-bit microprocessor	1620
PicoRV32	CPU	32-bit RISC-V microprocessor	6892

designs are stored in volatile SRAM registers. E-fuse was chosen as it was the only embedded non-volatile memory available in the target fabrication technology; however, non-volatile LUTs can be made of any embedded non-volatile technology and are preferable with enhanced security against reverse engineering. This work also studied SRAM storage as an alternative low-cost option for fabricating LUT-based obfuscation as SRAM storage is readily available in standard CMOS. However, it requires external and secure non-volatile key storage. Through this experiment, the intention is to show that proposed LUT-based obfuscation can be utilized with different memory storage technologies.

8.5. Post-Silicon Validation of LUT-based Logic-Locked Cores

8.5.1 Post-silicon Test Fixture

After fabrication, the post-silicon functional validation is performed on each benchmark using an Intel Aria 10 development kit as a test instrument. A **Printed Circuit Board** (PCB) with a socket for the chip to allow simple interfacing between the **Field Programmable Gate Array** (FPGA) and the test IC was developed as part of this work. The test structure is configured in the manner shown in Figure 8.5. This configuration is deployed to hardware, as shown in Figure 8.6.

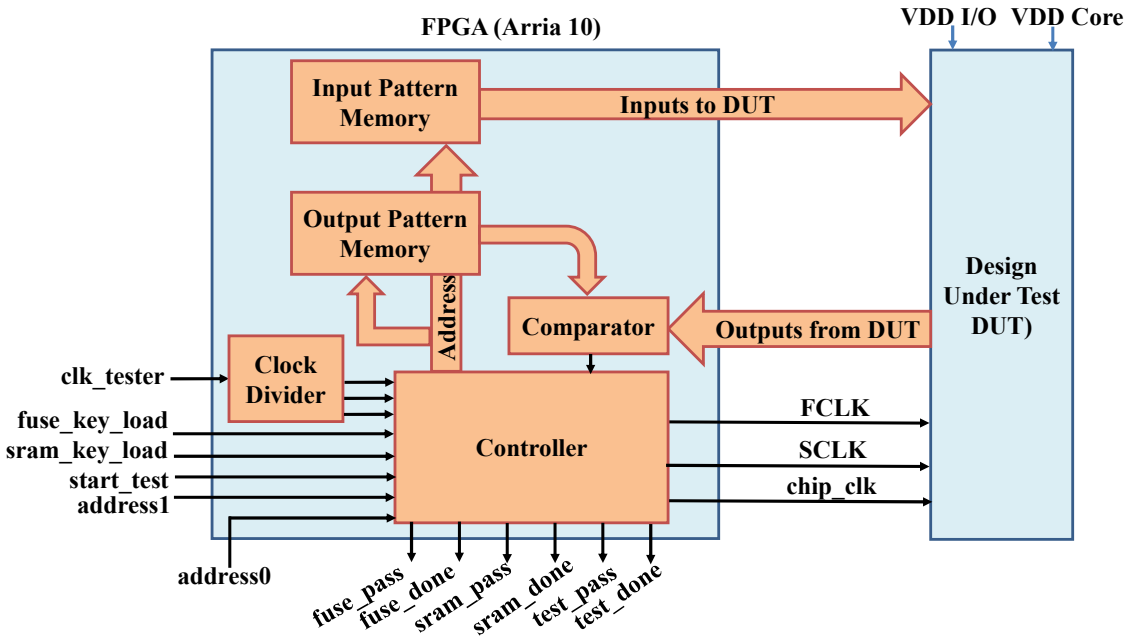


Figure 8.5. Diagram of FPGA-based Silicon validation setup for validating the functionality of obfuscated IP post fabrication.

For testing, the test vector input/response patterns are cached inside the FPGA and are selected via an address line from the controller. After a test vector is applied, the response is checked with the expected result. Test vectors are derived from the original test benches to validate the individual benchmarks. Both SRAM and e-fuse-based LUTs are validated, and the controller module on the FPGA performs all configuration programming before applying test patterns in the same manner as pre-silicon validation. Through this small-scale experiment, this work shows how post-silicon validation can be achieved.

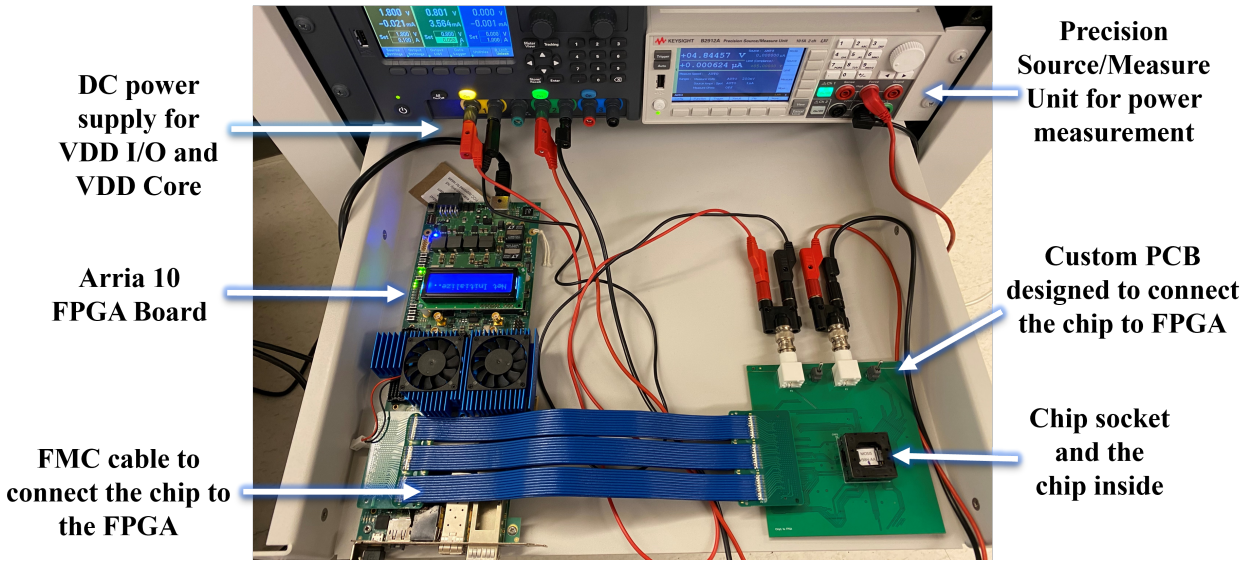


Figure 8.6. Test setup using the Intel Aria 10 development kit.

The video for testing and validating the obfuscated IP can be found here https://youtu.be/syhJz5Xk9_c and https://youtu.be/2cTZ36tpFK4.

8.5.2 Post-silicon validation results

All benchmarks are tested using test vectors generated from the original test bench of each design. To select the benchmark variant, control signals are added. Keys are loaded upon pressing a push-button switch. The LEDs have been assigned on the FPGA test setup to identify failure or the success of the testing process. To show that the test bench properly stimulates the obfuscated cells, incorrect keys are also programmed for each benchmark, and failure is observed for improperly programmed design.

Following the functionality validation results, this work have also validated the claim of LUT-

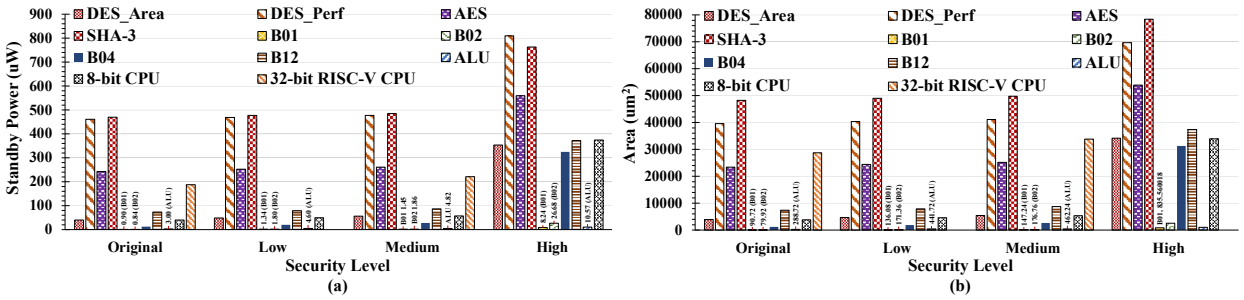


Figure 8.7. Study of (a) Standby power and (b) area for various obfuscation levels and benchmark for LUT-based obfuscation.

based obfuscation [4]. Figure 8.7 shows the standby power and area footprint of the fabricated designs. All 3 variants, i.e. (Low, Medium, and high), result in SAT timeout. These SAT-resilient benchmarks have an average of 7% area overhead for low obfuscated configuration while 14% and 262% for medium and high obfuscation. Standby power scales dramatically as the security level is improved with 33.93%, 45.93%, and 903.92% in the low, medium, and high cases, respectively. It must be noted that all versions of the design can resist the SAT-attack.

On the other hand, the LUT-based obfuscation incurs only 0.03%, 3.53%, and 17.82% average active power overheads for low, medium, and high obfuscations, respectively. These results validate that standby power and area dominate PPA cost as LUT unit cell size is increased. The timing information is not included as the part of the results as all designs maintained their original target frequency of 200 MHz for the RISC-V core and 100MHz for all other benchmarks.

Chapter 9

Conclusion & Future Work

9.1. Conclusion

This work studied logic locking using reconfigurable logic, i.e., Lookup Table, and performed a comprehensive analysis of LUT-based obfuscation. The initial experimental results showed that the size of LUT is the most influential and straightforward factor in meeting SAT resiliency. However, it introduces significant and unacceptable PPA overheads. To mitigate the overhead due to the large LUT sizes, the work introduces a customized LUT that breaks the trade-off between security and design overheads. The introduced obfuscation using the proposed custom LUT reduces the design overhead to an acceptable range without compromising the SAT resiliency. After obfuscation using the proposed primitive, the obtained overhead results indicate that the proposed obfuscation is an effective and scalable solution against today's state-of-the-art attacks.

For providing maximum resiliency against zero-day attacks, the work proposes the idea of 'defense-in-depth' using a Power Side-Channel Attack (P-SCA) resilient block, which uses emerging reconfigurable devices and logic locking. The resulting primitive blocks traditional and ML-assisted P-SCA while eliminating the threat of SAT-attack. The obfuscation proposed in this work is also shown to be resilient against various other attacks such as removal/structural attack, HackTest, Scan, and Shift attack, to name a few.

The work expands on the methodology of integrating and validating the LUT-based obfuscation. Security Evaluation Platform for Hardware Logic obfuscation using Intelligent Artificial Neural Net (SEPIANN) is proposed for security validation. This system-level framework instantaneously

estimates the obfuscation strength in attack resiliency time, eliminating the need to simulate de-obfuscation using the SAT attack. The cardinal part of the SEPIANN framework is a conjunctive normal form (CNF)-based bipartite graph network (CNF-NET), a novel convolutional graph-based neural network. CNF-NET leverages the conjunctive normal form (CNF)-based bipartite graph to characterize the SAT problem's complexity and incorporates the energy-based layers to encode an obfuscated IC to predict the runtime distribution. This end-to-end framework thus automatically extracts the determinant features to predict de-obfuscation runtime for state-of-the-art SAT attacks.

The ability to instantaneously assess the security of design allows the user to perform design-space-exploration for the optimal design of the IP. The proposed flow is non-disruptive to the current physical design flow for obfuscation using LUT. The easier integration of the proposed design flow uplifts the chances of realizing the LUT-based obfuscation as a promising candidate for restoring trust in silicon.

Finally, nonvolatile internal (e-fuse) and volatile external (SRAM) LUT key configuration-based designs were fabricated to demonstrate the practical implementation of LUT-based obfuscation.

9.2. Contribution

Following is the list of peer-reviewed conference and journal paper that were published.

- **G. Kolhe**, T. D. Sheaves, K. I. Gubbi, T. Kadale, S. Rafatirad, S. M. P. Dinakarrao, A. Sasan, H. Mahmoodi, and H. Homayoun, "Silicon validation of LUT-based logic-locked IP cores," in 2022 59th ACM/IEEE Design Automation Conference (DAC), 2022.

- **G. Kolhe**, S. Salehi, T. D. Sheaves, H. Homayoun, S. Rafatirad, S. M. P. Dinakarrao, and A. Sasan, "LOCK & ROLL: Deep-Learning Power Side-Channel Attack Mitigation using Emerging Reconfigurable Devices and Logic Locking," in 2022 59th ACM/IEEE Design Automation Conference (DAC), 2022.

- **G. Kolhe**, T. D. Sheaves, S. M. P. Dinakarrao, H. Mahmoodi, S. Rafatirad, A. Sasan, and H. Homayoun, "Breaking the design and security trade-off of look-up table-based obfuscation," ACM Trans. Des. Autom. Electron. Syst., jan 2022.

- **G. Kolhe**, S. Salehi, T. D. Sheaves, H. Homayoun, S. Rafatirad, S. M. P. Dinakarrao, and A. Sasan, "Securing hardware via dynamic obfuscation utilizing reconfigurable interconnect and logic blocks," in 2021 58th ACM/IEEE Design Automation Conference (DAC), pp. 229–234, 2021.

- Z. Chen, L. Zhang, **G. Kolhe**, H. M. Kamali, S. Rafatirad, S. M. P. Dinakarrao, H. Homayoun, C.-T. Lu, and L. Zhao, "Deep graph learning for circuit deobfuscation," May 2021.

- Z. Chen, **G. Kolhe**, S. Rafatirad, C.-T. Lu, S. M. P.Dinakarrao, H. Homayoun, and L. Zhao, "Estimating the circuit de-obfuscation runtime based on graph deep learning," in 2020 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 358–363, 2020.

- R. Hassan, **G. Kolhe**, S. Rafatirad, H. Homayoun, and S. M. P. Dinakarrao, "A neural network-based cognitive obfuscation towards enhanced logic locking," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 1–1, 2021.

- R. Hassan, **G. Kolhe**, S. Rafatirad, H. Homayoun, and S. M. P.Dinakarrao, "SATconda: SAT to SAT-hard clause translator," in 2020 21st International Symposium on Quality Electronic

Design (ISQED), pp. 155–160, 2020.

- V. V. Menon, **G. Kolhe**, J. Fifty, A. Schmidt, J. Monson, M. French, Y. Hu, P. A. Beerel, and P. Nuzzo, "Logic obfuscation: Modeling attack resiliency," in GOMACTech Conference, 2020.

- **G. Kolhe**, H. M. Kamali, M. Naicker, T. D. Sheaves, H. Mahmoodi, S. M. P.Dinakarrao, H. Homayoun, S. Rafatirad, and A. Sasan, "Security and complexity analysis of LUT-based obfuscation: From blueprint to reality," in 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1–8, 2019. **[Best Paper Nominee]**

- V. V. Menon, **G. Kolhe**, A. Schmidt, J. Monson, M. French, Y. Hu, P. A. Beerel, and P. Nuzzo,"Quantifying security and overheads for obfuscation of integrated circuits," in GOMACTech Conference, 2019.

- V. V. Menon, **G. Kolhe**, A. Schmidt, J. Monson, M. French, Y. Hu, P. A. Beerel, and P. Nuzzo, "System-level framework for logic obfuscation with quantified metrics for evaluation," in 2019 IEEE Cybersecurity Development (SecDev), pp. 89–100, 2019.

- **G. Kolhe**, S. M. P.Dinakarrao, S. Rafatirad, H. Mahmoodi, A. Sasan, and H. Homayoun, "On custom LUT-based obfuscation," in Proceedings of the 2019 on Great Lakes Symposium on VLSI, GLSVLSI'19, (New York, NY, USA), p. 477–482, Association for Computing Machinery, 2019.

9.3. Future Work

The LUT-based obfuscation can offer a 'defense-in-depth' solution. However, the integration of the LUTs in the design is performed at the very end of the design process. A new methodology can be devised where the process of securing the IP can be modified such that IPs are designed since the beginning of their life cycle to provide resiliency against state-of-the-art attacks. This ideology follows the quote 'security should be baked in, not bolted on.' Such an approach can allow the designer to optimize obfuscation's security and overhead impact.

To allow for the integration of security at the earlier stages of IP development, one should be able to identify and map parts of the design to LUT and quantify its effect on security and design. Moreover, the ideology of software and hardware co-design can be utilized to provide multi-layer security.

While multi-layered security is the only way to fly, the primitive proposed in this work heavily relies on the MRAM. The MRAM technology is still in its development phases, and future work can focus on finding alternative solutions that can make the proposed primitive industry-ready.

Chapter 10

Acknowledgments

1. The work of Lookup table-based obfuscation was supported by Defense Advanced Research Projects Agency (DARPA-AFRL, #FA8650-18-1-7819)

2. The fabrication of IP was funded by Defense Advanced Research Projects Agency (DARPA-AFRL, #FA8650-18-1-7819). The design and fabrication of the IP was done in collaboration with team at San Francisco State University led by Hamid Mahmoodi and Logimem Corporation.

3. The work LOCK & ROLL was supported in part by the National Science Foundation through Computing Research Association for CIFellows #2030859 and NSF CHEST IUCRC Industrial Support.

4. The SEPIANN work was supported in part by NSF CHEST IUCRC Industrial Support.

Chapter 11

References

[1] H. M. Kamali, K. Z. Azar, H. Homayoun, and A. Sasan, "Interlock: An intercorrelated logic and routing locking," in *Proceedings of the 39th International Conference on Computer-Aided Design*, ICCAD '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[2] M. Yasin, J. J. Rajendran, and O. Sinanoglu, *Pre-SAT Logic Locking*, pp. 33–46. Cham: Springer International Publishing, 2020.

[3] S. Dupuis and M.-L. Flottes, "Logic locking: A survey of proposed methods and evaluation metrics," *Journal of Electronic Testing*, vol. 35, pp. 273–291, Jun 2019.

[4] G. Kolhe, H. M. Kamali, M. Naicker, T. D. Sheaves, H. Mahmoodi, P. D. Sai Manoj, H. Homayoun, S. Rafatirad, and A. Sasan, "Security and complexity analysis of lut-based obfuscation: From blueprint to reality," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2019.

[5] D. Mitchell, B. Selman, and H. Levesque, "Hard and easy distributions of sat problems," in *AAAI*, vol. 92, pp. 459–465, 1992.

[6] S. Salehi and *et al.*, "Clockless spin-based look-up tables with wide read margin," in *GLSVLSI*, GLSVLSI '19, (New York, NY, USA), 2019.

[7] Nathan Associates Inc., "Beyond borders - the global semiconductor value chain," May 2016.

[8] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proceedings of the IEEE*, vol. 102, pp. 1283–1295, Aug 2014.

[9] J. P. Skudlarek, T. Katsioulas, and M. Chen, "A platform solution for secure supply-chain and chip life-cycle management," *Computer*, vol. 49, pp. 28–34, Aug 2016.

[10] M. M, S. Garg, and M. V. Tripunitara, "Integrated circuit decamouflaging: Reverse engineering camouflaged ics within minutes," in *NDSS*, (USA), NDSS, 2015.

[11] 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*, (USA), pp. 137–143, IEEE, May 2015.

[12] T. Winograd, H. Salmani, H. Mahmoodi, and et.al., "Hybrid STT-CMOS designs for reverse-engineering prevention," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference*, (USA), pp. 1–6, IEEE, June 2016.

[13] J. A. Roy, F. Koushanfar, and I. L. Markov, "Ending piracy of integrated circuits," *Computer*, vol. 43, pp. 30–38, Oct 2010.

[14] J. Rajendran, Y. Pino, O. Sinanoglu, and et.al., "Security analysis of logic obfuscation," in *Design Automation Conference 2012*, (USA), pp. 83–89, IEEE, June 2012.

[15] J. Rajendran, H. Zhang, C. Zhang, and et.al., "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 64, pp. 410–424, Feb 2015.

[16] J. Rajendran, M. Sam, O. Sinanoglu, and et.al., "Security analysis of integrated circuit camouflaging," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security*, CCS '13, (Germany), pp. 709–720, ACM, 2013.

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

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

[19] F. Yang and *et al.*, "Stripped functionality logic locking with hamming distance-based restore unit (SFLL-HD)–Unlocked," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, 2019.

[20] M. Yasin and *et al.*, "SFLL-HLS: Stripped-functionality logic locking meets high-level synthesis," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–4, 2019.

[21] M. Yasin, B. Mazumdar, and J. J. V. R. et.al, "SARLock: SAT attack resistant logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2016.

[22] Y. Xie and A. Srivastava, "Anti-sat: Mitigating sat attack on logic locking," in *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, (Germany), pp. 199–207, IEEE, Feb 2019.

[23] B. Shakya and *et al.*, "CAS-Lock: A security-corruptibility trade-off resilient logic locking scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 175–202, 2020.

[24] 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, pp. 97–122, Nov. 2018.

[25] G. Kolhe, S. M. PD, S. Rafatirad, H. Mahmoodi, A. Sasan, and H. Homayoun, "On custom LUT-based obfuscation," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, GLSVLSI '19, (New York, NY, USA), p. 477–482, Association for Computing Machinery, 2019.

[26] G. Kolhe, T. D. Sheaves, S. M. P. D, H. Mahmoodi, S. Rafatirad, A. Sasan, and H. Homayoun, "Breaking the design and security trade-off of look-up table-based obfuscation," *ACM Trans. Des. Autom. Electron. Syst.*, jan 2022.

[27] A. Attaran, T. D. Sheaves, P. K. Mugula, and H. Mahmoodi, "Static design of spin transfer torques magnetic look up tables for ASIC designs," in *Proceedings of the 2018 on Great*

Lakes Symposium on VLSI, GLSVLSI '18, (New York, NY, USA), p. 507–510, Association for Computing Machinery, 2018.

[28] "Synopsys 32/28nm gneric process design kit," *online: https://www.synopsys.com/community/university-program/teaching-resources.html*, 2005.

[29] E. Nudelman, K. Leyton-Brown, H. H. Hoos, and et.al., "Understanding Random SAT: Beyond the Clauses-to-variables Ratio," in *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming*, CP'04, (Heidelberg), pp. 438–452, Springer-Verlag, 2004.

[30] G. Tseitin, "On the complexity of derivation in propositional calculus," *Studies in Constructive Mathematics and Mathematical Logic*, pp. 115–125, 1968.

[31] M. S. *et al.*, "Extending sat solvers to cryptographic problems," in *Int'l Conf. on Theory and Applications of Satisfiability Testing (SAT)*, pp. 244–257, 2009.

[32] M. Yasin, A. Sengupta, M. T. Nabeel, and et.al., "Provably-secure logic locking: From theory to practice," in *2017 ACM SIGSAC Conference on Computer and Communications Security*, (USA), pp. 1601–1618, ACM, ACM, 2017.

[33] K. Shamsi, M. Li, T. Meade, and et.al., "AppSAT: Approximately deobfuscating integrated circuits," in *Hardware Oriented Security and Trust*, (USA), pp. 95–100, IEEE, May 2017.

[34] M. Yasin, B. Mazumdar, O. Sinanoglu, and et.al., "Security analysis of Anti-SAT," in *Asia and South Pacific Design Automation Conference*, (Japan), pp. 342–347, IEEE, Jan 2017.

[35] D. S. *et al.*, "Functional analysis attacks on logic locking," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019.

[36] N. Rangarajan and *et al.*, "Opening the doors to dynamic camouflaging: Harnessing the power of polymorphic devices," *IEEE Transactions on Emerging Topics in Computing*, p. 1–1, 2020.

[37] G. Kolhe, S. Salehi, T. D. Sheaves, H. Homayoun, S. Rafatirad, M. P. D. Sai, and A. Sasan, "Securing hardware via dynamic obfuscation utilizing reconfigurable interconnect and logic blocks," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 229–234, 2021.

[38] G. Kolhe, S. Salehi, T. D. Sheaves, H. Homayoun, S. Rafatirad, S. M. P. Dinakarrao, and A. Sasan, "LOCK & ROLL: Deep-Learning Power Side-Channel Attack Mitigation using Emerging Reconfigurable Devices and Logic Locking," in *2022 59th ACM/IEEE Design Automation Conference (DAC)*, 2022.

[39] P. e. Kocher, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, (Berlin, Heidelberg), pp. 388–397, 1999.

[40] J. Kim and *et al.*, "A technology-agnostic MTJ SPICE model with user-defined dimensions for STT-MRAM scalability studies," in *2015 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, 9 2015.

[41] S. e. Picek, "On the performance of convolutional neural networks for side-channel analysis," in *Security, Privacy, and Applied Cryptography Engineering*, 2018.

[42] S. P. *et al.*, "Advancing hardware security using polymorphic and stochastic spin-hall effect devices," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 97–

102, 2018.

[43] M. e. Yasin, "Testing the Trustworthiness of IC Testing: An Oracle-Less Attack on IC Camouflaging," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2668–2682, 2017.

[44] M. e. Yasin, "Activation of logic encrypted chips: Pre-test or post-test?," in *DATE*, 2016.

[45] M. Yasin, J. J. Rajendran, O. Sinanoglu, and et.al., "On improving the security of logic locking," vol. 35, pp. 1411–1424, Sep. 2016.

[46] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware IP protection," in *2014 51st ACM/EDAC/IEEE Design Automation Conference*, (USA), pp. 1–5, IEEE, June 2014.

[47] H. M. Kamali, K. Z Azar, K. Gaj, and et.al., "LUT-Lock: A Novel LUT-Based Logic Obfuscation for FPGA-Bitstream and ASIC-Hardware Protection," (China), pp. 405–410, IEEE, July 2018.

[48] M. Yasin, B. Mazumdar, J. Rajendran, and et.al., "Sarlock: Sat attack resistant logic locking," in *Hardware Oriented Security and Trust*, (USA), pp. 236–241, IEEE, May 2016.

[49] S. Khaleghi and W. Rao, "Hardware obfuscation using strong pufs," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 321–326, IEEE, 2018.

[50] C. M. Bishop and T. M. Mitchell, "Pattern recognition and machine learning," 2014.

[51] G. Kolhe, T. D. Sheaves, K. I. Gubbi, T. Kadale, S. Rafatirad, S. M. P. Dinakarrao, A. Sasan, H. Mahmoodi, and H. Homayoun, "Silicon validation of LUT-based logic-locked IP cores," in *2022 59th ACM/IEEE Design Automation Conference (DAC)*, 2022.

[52] Z. Chen, L. Zhang, G. Kolhe, H. M. Kamali, S. Rafatirad, S. M. Pudukotai Dinakarrao, H. Homayoun, C.-T. Lu, and L. Zhao, "Deep graph learning for circuit deobfuscation," May 2021.

[53] R. Hassan, G. Kolhe, S. Rafatirad, H. Homayoun, and S. M. P. Dinakarrao, "A neural network-based cognitive obfuscation towards enhanced logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2021.

[54] Z. Chen, G. Kolhe, S. Rafatirad, C.-T. Lu, S. Manoj P.D., H. Homayoun, and L. Zhao, "Estimating the circuit de-obfuscation runtime based on graph deep learning," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 358–363, 2020.

[55] R. Hassan, G. Kolhe, S. Rafatirad, H. Homayoun, and S. M. Dinakarrao, "SATConda: SAT to SAT-Hard Clause Translator," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*, pp. 155–160, 2020.

[56] V. V. Menon, G. Kolhe, J. Fifty, A. Schmidt, J. Monson, M. French, Y. Hu, P. A. Beerel, and P. Nuzzo, "Logic obfuscation: Modeling attack resiliency," in *GOMACTech Conference*, 2020.

[57] V. V. Menon, G. Kolhe, A. Schmidt, J. Monson, M. French, Y. Hu, P. A. Beerel, and P. Nuzzo, "Quantifying security and overheads for obfuscation of integrated circuits," in *GOMACTech Conference*, 2019.

[58] V. V. Menon, G. Kolhe, A. Schmidt, J. Monson, M. French, Y. Hu, P. A. Beerel, and P. Nuzzo, "System-level framework for logic obfuscation with quantified metrics for evaluation," in *2019 IEEE Cybersecurity Development (SecDev)*, pp. 89–100, 2019.

[59] M. Soos and K. S. Meel, "Bird: Engineering an efficient cnf-xor sat solver and its applications to approximate model counting," in *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 1 2019.

[60] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Satzilla: portfolio-based algorithm selection for sat," *Journal of artificial intelligence research*, vol. 32, pp. 565–606, 2008.

[61] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[62] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 3844–3852, Curran Associates, Inc., 2016.

[63] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[64] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[65] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[66] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1993–2001, 2016.

[67] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proc. CVPR*, vol. 1, p. 3, 2017.

[68] F. R. Chung, *Spectral graph theory*. No. 92, American Mathematical Soc., 1997.

[69] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *Applied and Computational Harmonic Analysis*, vol. 40, no. 2, pp. 260–291, 2016.

[70] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[71] G. E. Hinton and R. R. Salakhutdinov, "A better way to pretrain deep boltzmann machines," in *Advances in Neural Information Processing Systems*, pp. 2447–2455, 2012.

[72] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[73] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[74] H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio, "Learning algorithms for the classification restricted boltzmann machine," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 643–669, 2012.

[75] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural networks: Tricks of the trade*, pp. 599–619, Springer, 2012.

[76] M. A. Cueto, J. Morton, and B. Sturmfels, "Geometry of the restricted boltzmann machine," *Algebraic Methods in Statistics and Probability,(eds. M. Viana and H. Wynn), AMS, Contemporary Mathematics*, vol. 516, pp. 135–153, 2010.

[77] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Artificial intelligence and statistics*, pp. 448–455, 2009.

[78] C. Poultney, S. Chopra, Y. L. Cun, *et al.*, "Efficient learning of sparse representations with an energy-based model," in *Advances in neural information processing systems*, pp. 1137–1144, 2007.

[79] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[80] K. Eggensperger, M. Lindauer, and F. Hutter, "Neural networks for predicting algorithm runtime distributions.," in *IJCAI*, pp. 1442–1448, 2018.

Appendix

Automated Tool for LUT-based Obfuscation

To thwart the prevailing RE attempts on the IP, LUT-based obfuscation is discussed. LUT-based renders superior attack resiliency against the state-of-the-art SAT-attack and various other attacks as discussed in Section 5.3.3 and Section 6.3.3. In this section, fully automated Graphical User Interface (GUI) that automates the process of obfuscating a given netlist and gives the user the ability to simulate the SAT-attack on the obfuscated netlist is introduced.

The GUI flow is effectively captured in Figure 11.1. The input to the GUI (Module 1) is the RTL or synthesized Netlist, which needs to be obfuscated. In the case of the synthesized netlist given as an input, the user needs to provide the path to the Standard Cell Library, which was used for synthesizing the netlist. This tool converts the given netlist into the flattened netlist using the Synopsys GTECH library for further processing. The tool uses this flattened netlist throughout the entire flow of the obfuscation.

The tool offers the user a choice to obfuscate a netlist using Low-output Corruptibility with no back-to-back LUT placement, also known as LC_NoGen as discussed in Section 4.3.3 (Module 2), or the user can modify the config file and provide a custom list of selected gates. When the user wants to leverage the tool to find the set of gates for obfuscation, the user must specify the number of gates that needs to be obfuscated along with the size of the LUT that should be used for obfuscation. After providing all user inputs (i.e., Netlist from Module 1, obfuscation coverage, size of LUT), the tool finds the gates for obfuscation. The module also generates the config file and the bench files needed for the SAT simulation.

For the gate selection process, the in-house developed python script (Module 2) generates the graph of the flattened netlist before it is passed as an input to the function that selects the gates

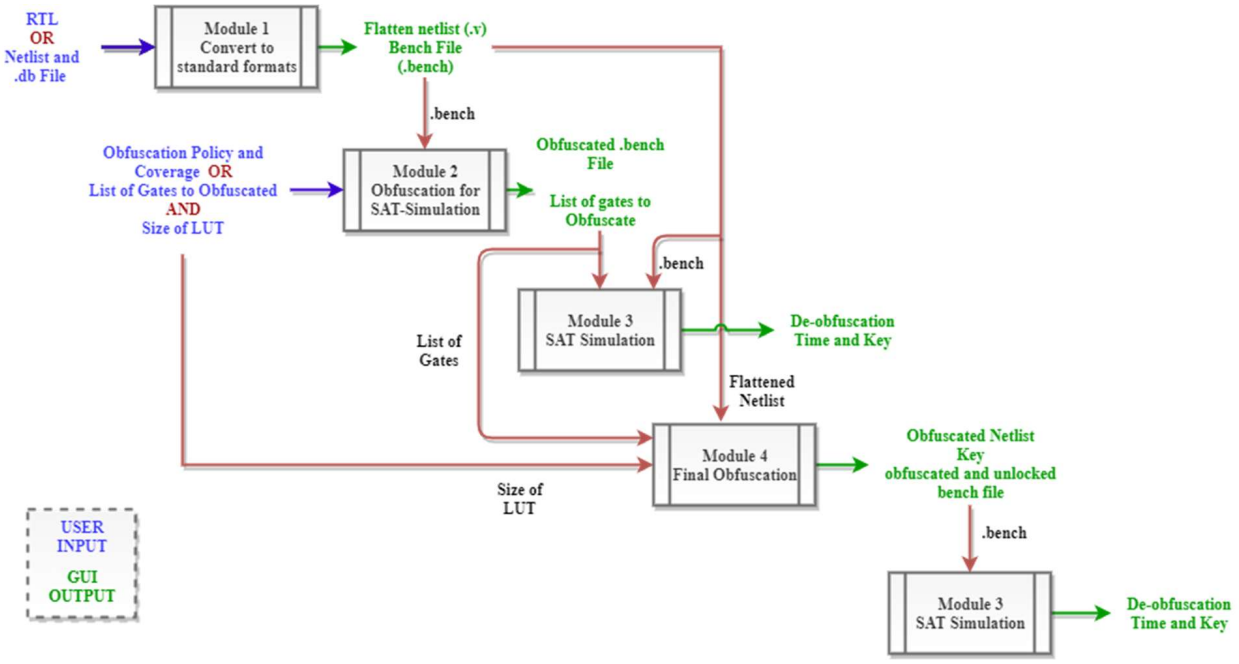


Figure 11.1. Automated LUT-based Obfuscation Flow

according to the replacement policies. The Low output corruptibility with no back-to-back gate replacement, i.e., *LC_NoGen* policy bundled in the tool increases efforts and the number of iterations needed to find the correct key when the SAT-attack is performed. The ILP model is used to minimize the output corruptibility, which optimally selects the gate. The python script generates the two bench files, i.e., oracle and the obfuscated bench files, along with the config file, which will be utilized later for the backend process. The obfuscated bench file has the gates replaced using the customized LUT, where the user specifies the size of the large LUT. Additional 2-input LUTs are added to the input of the large LUT to render the customized LUT block. The customized LUT further aggravates the level of security offered by the LUT while having minimal design overheads compared to the traditional LUT-based obfuscation. The bench files can simulate the SAT-attack in Module 3.

Before synthesizing the final netlist, the user can leverage the SAT-attack simulation (Module 3). SAT-attack requires the oracle netlist and obfuscated netlist in bench format without the presence of the sequential elements. The SAT-attack also has a specific limitation in terms of gates that can be simulated; however, the bench format generated by the in-built script (Module 2) generates the bench format that can be simulated directly using the SAT-attack. The SAT-attack has been

bundled as part of the GUI and is compatible with the pre-existing SAT-attack frameworks. Instead of simulation SAT-attack, SEPIANN model can also be leveraged. The input to the SEPIANN can be the obfuscated bench file generated by the tool.

After the SAT simulation, if the obfuscated bench file satisfies the security constraints, the GUI can be leveraged to render the final obfuscated netlist. The flattened netlist that was earlier generated in Module 1 and the list of the gates to be obfuscated is passed for generating the final obfuscated netlist.

The backend script (Module 4) requires the flattened netlist and the config file for obfuscating the netlist. The script first creates the Verilog Module Object (VMO) data structure of the standard cell library and further maps it to the flattened netlist, which results in the VMO instance. The script reads the configuration file, which contains the list of the gates to be replaced and the configuration of the LUT to generate the Configurable Logic Elements (CLE). The generated CLE is added to the standard cell VMO collection and replaces the VMO instances of the gates in the top layer selected for the replacement.

The program creates and runs an exhaustive test bench to extract the individual CLE configuration keys in parallel using a logic simulator (i.e., VCS). The output configuration keys are then chained together to form the top-level configuration key. The in-house script generates the mix-style RTL that outputs the VMO instances as a gate-level netlist, while the CLE elements are written out as an RTL file. The script for the last time calls the synthesizer to synthesize the mixed RTL file into a netlist. The output of this Module 4 is a Design Key and an Obfuscated Netlist.

To allow for the SAT simulation post synthesizing, we have added another set of a script (Module 5) that generates the bench file (oracle and obfuscated) from the synthesized verilog file. The script reads the key-value generated from the VCS and assigns the key values in the bench format such that it remains in a valid format to simulate the SAT-attack. SAT-attack post-synthesis might be required, as the synthesis tools change the netlist to a certain degree for optimizing the PPA overheads. The output of this module is the configuration key and the netlist (.v format), along with the bench files for SAT simulations. The flexibility in the input format and the functionality offered by the automated tool flow is shown in Figure 11.2. The GUI has four sections/modules that carry out the various tasks and give the user various options.

The key specification of the automated tool flow is summarized below.

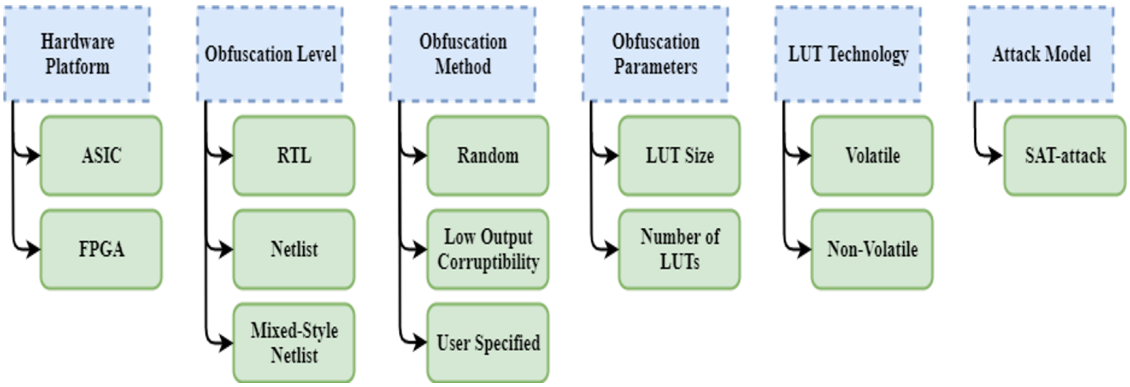


Figure 11.2. Flexibility and Functionality offered by the Customized LUT-obfuscation GUI Tool

1. Synthesizing the RTL, mix-style RTL, or synthesized netlist to standard flatten netlist using the SAED 32nm library and create the bench file for SAT-simulation.

2. Finding the gates by leveraging various replacement policy. The tool also provides the over-riding methods where the user can provide gates for generating the obfuscated netlist.

3. The tool offers full control over obfuscation coverage and LUT size used for obfuscation.

4. SAT simulation can be performed, and the tool handles all necessary bench file conversion. The tool enables the user to leverage SEPIANN framework for finding the SAT-resiliency instantly.

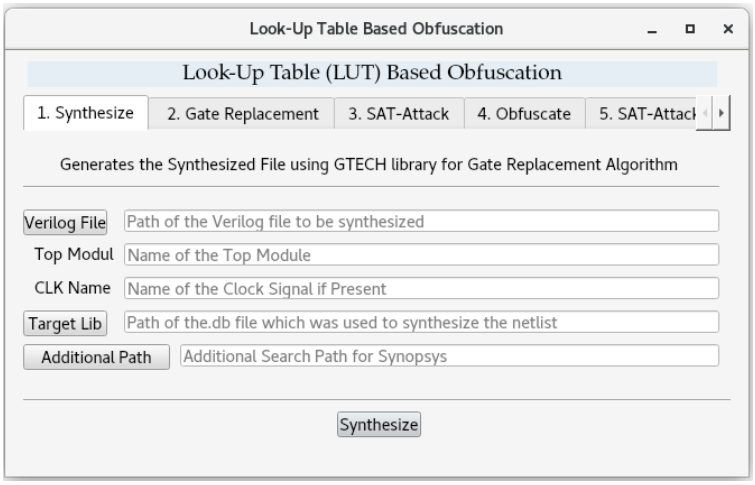


Figure 11.3. Synthesize Module (Module 1)

Figure 11.3 shows the GUI interface with Four different modules and an about page. It provides the user with a search box to find the relative paths and run that individual module. Each module

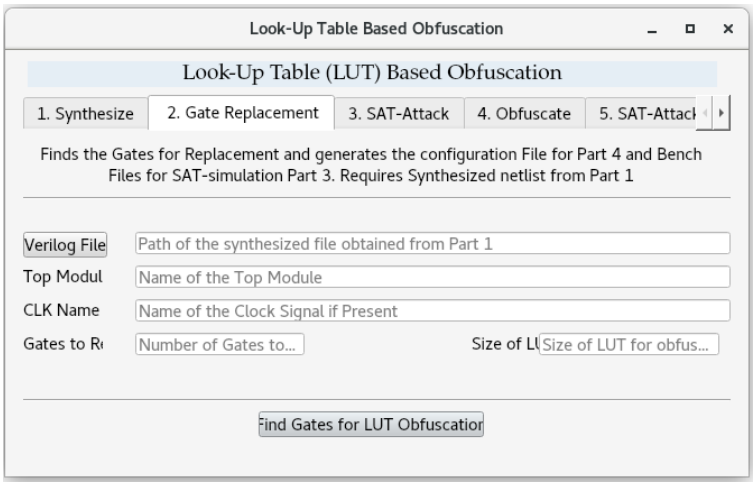


Figure 11.4. Gate Replacement Module (Module 2)

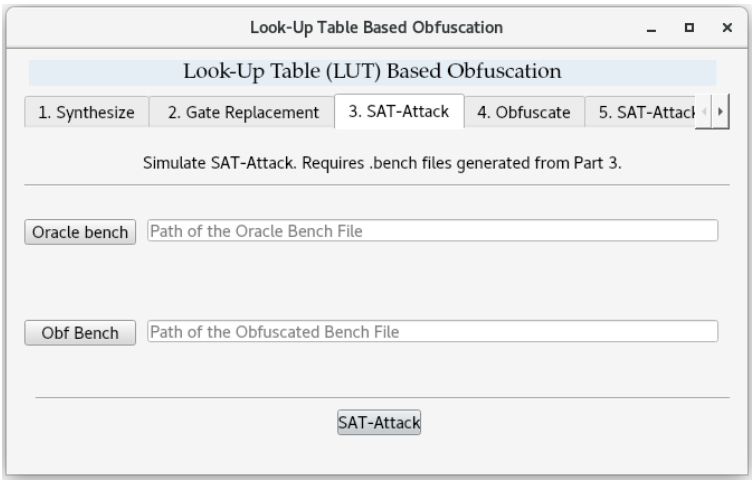


Figure 11.5. SAT-Attack Module (Module 3)

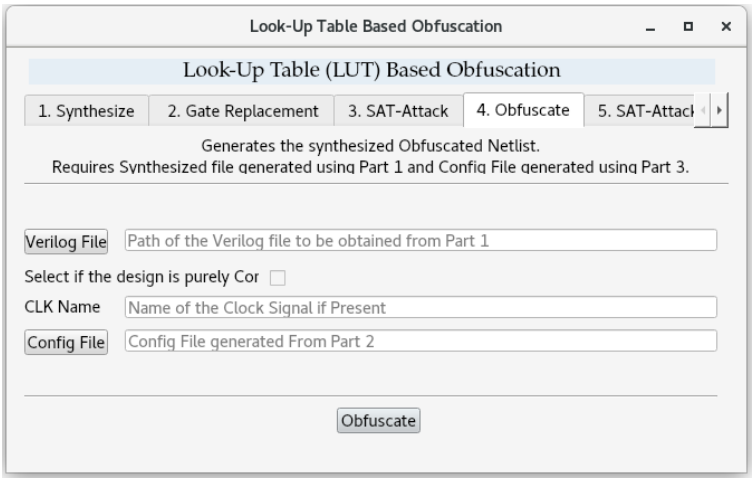


Figure 11.6. Final Obfuscation Module (Module 4)

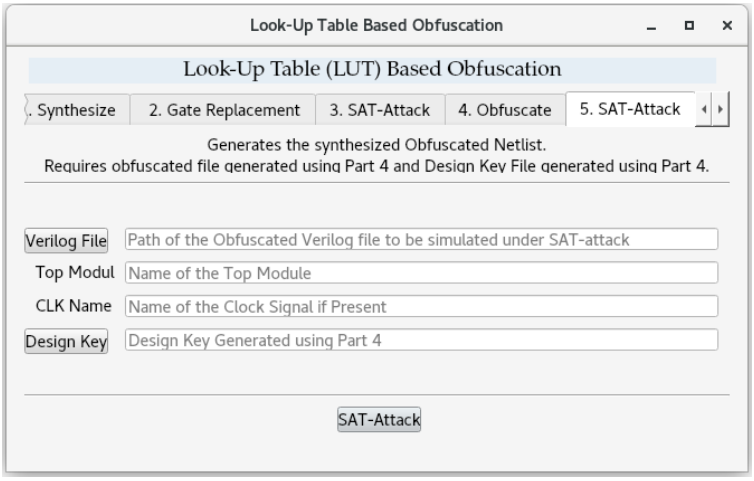


Figure 11.7. SAT-attack post obfuscation (Module 5)

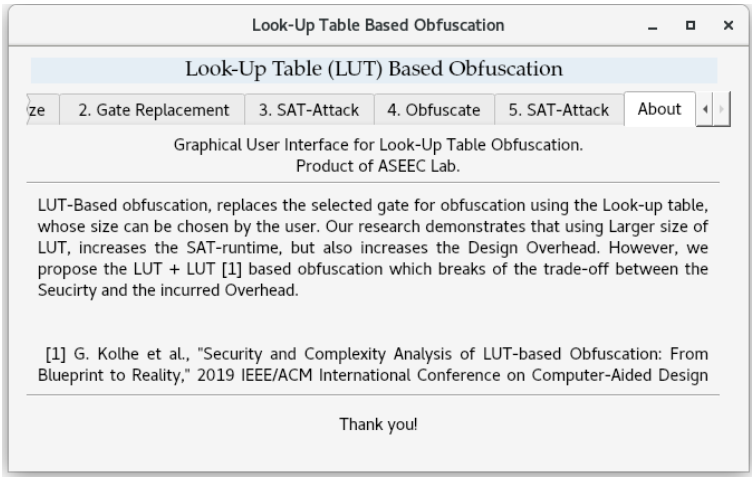


Figure 11.8. About Page

provides its usage and the files that are required. The modular design of the GUI gives the user to understand the obfuscation process and repeat the experiment if necessary, without the need to start from the beginning.

Acronyms

BL	Bit Line.
SE	Sense Enable.
SE	Scan Enable.
WE	Write Enable.
ASIC	Application-Specific Integrated Circuit.
ATPG	Automatic Test Generation Pattern.
BM	Boltzmann Machines.
CA	Civilian Acquisition.
CAD	Computer-aided design.
CB	Configuration Bit.
CC	conflict clauses.
CDCL	Conflict Driven Clause Learning.
CEP	Common Evaluation Platform.
CLE	Configurable Logic Elements.
CM	Contract Manufacturers.
CMOS	Complementary Metal-Oxide Semiconductor.
CNF	conjunctive normal form.
CNN	Convolutional Neural Network.

DCNN Diffusion Convolutional Neural Network.

DIP Distinguishing Inputs.

DIVC Distinguishing Input Validation Circuit.

DL Deep Learning.

DNN Deep Neural Network.

DoD Department of Defense.

DPLL Davis-Putnam-Logemann-Loveland.

DSE Design Space Exploration.

EDA Electronic Design Automation.

FIB focused ion beam.

FPGA Field-Programmable Gate Array.

GCN Graph Convolutional Network.

GUI Graphical User Interface.

HD Hamming Distance.

IC Integrated Circuit.

ILP Integer Linear Programming.

IP Intellectual Property.

KDC Key-Differentiating Circuit.

KPC Key-Programmable Circuit.

KPG key-programmable.

LCAC Learned-Clause Avoidance Circuit.

LCO Logical Cone Output.

LOCK & ROLL Deep-Learning Power Side-Channel Attack Mitigation using Emerging Reconfigurable Devices and Logic Locking.

LUT Look-Up Table.

MC Monte Carlo.

MESO Magneto-Electric Spin-Orbit.

ML Machine Learning.

MOS Metal-Oxide Semiconductor.

MRAM Magnetoresistive Random-Access Memory.

MRAM-LUT Magnetoresistive Random-Access Memory based Look-up Table.

MTJ Magnetic Tunnel Junction.

OCM Original Chip Manufacturer.

OEM Original Equipment Manufacturers.

P-SCA Power Side-Channel Attack.

PG Pass Transistors.

PO primary output.

PPA Power-Performance-Area.

PV Process Variation.

RBF Radial Basis Function.

RBM Restricted Boltzmann Machines.

RE Read Enable.

RE Reverse Engineering.

RTL Register-Transfer Level.

SAT Satisfiability.

SAT-attack Boolean Satisfiability Attack.

SCK Set of Candidate Keys.

SCKVC Validation Circuit for Set of Candidate Keys.

SEM scanning electron microscopy.

SEPIANN Security Evaluation Platform for Hardware Logic Obfuscation using Intelligent Artificial Neural Net.

SFLL Stripped Functionality Logic Locking.

SMT Satisfiability Module Theories.

SOM Scan-Enable Obfuscation Mechanism.

SRAM Static Random Access Memory.

SSE Mean Squared Error.

STT Spin Transfer Torque.

STT-LUT Spin Transfer Torque-based Look-up Table.

STT-MRAM Spin Transfer Torque Magnetoresistive Random Access Memory.

SVK Set of Valid Keys.

SVM Support Vector Machine.

SyM-LUT Symmetric Magnetoresistive Random-Access Memory-based Look-up Table.

TG Transmission Gates.

US United States.

VMO Verilog Module Object.