SECURITY AND COMPLEXITY ANALYSIS OF LUT-BASED OBFUSCATION: A COMPREHENSIVE STUDY

by

Gaurav Kolhe

A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Computer Engineering

Committee:

_____	Dr. Houman Homayoun, Thesis Director
_____	Dr. Avesta Sasan, Committee Member
_____	Dr. Setareh Rafatirad, Committee Member
_____	Dr. Monson Hayes, Department Chair
_____	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering
Date:_____	Summer Semester 2018 George Mason University Fairfax, VA

Security and Complexity Analysis of LUT-based Obfuscation: A comprehensive Study

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

by

Gaurav Kolhe
Bachelor of Engineering
University of Nagpur, 2015

Director: Houman Homayoun, Associate Professor
Electrical and Computer Engineering Department

Summer Semester 2018
George Mason University
Fairfax, VA

DEDICATION

This thesis is dedicated to my loving parents for their endless love, support and encouragement.

ACKNOWLEDGEMENTS

There have been many people who have walked alongside me during the past couple years. They have guided me, placed opportunities in front of me and showed me the doors that might be useful to open. I would like to thank each one of them. I would especially thank Dr. Houman Homayoun and Dr. Avesta Sasan, with your encouragement I have achieved all the results and accomplished my research. A very big thank goes to my family, love and my roomies.

Thank you, Lord, for always being there for me.

This thesis is only a beginning of my journey

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Application Specific Integrated Circuit ..... ASIC
Avoiding Unintentionally Correct Key Generation ..... LC_NoGen
Boolean Satisfiability ..... SAT
Breadth First Search ..... BFS
Complementary metal–oxide–semiconductor ..... CMOS
Computer Aided Design ..... CAD
Conflict Clauses CC
Conflict Driven Clause Learning ..... CDCL
Conjunctive Normal Form ..... CNF
Control and Data Flow Graph ..... CDFG
Design for Testability ..... DFT
Design-for-trust ..... DFTr
D-FlipFlop ..... D-FF
Distinguishing Input ..... DI
Distinguishing Pattern ..... DIP
DI-Validation Circuit ..... DIVC
Field Programmable Gate Array ..... FPGA
Finite State Machine ..... FSM
Focused Ion Beam ..... FIB
Graphic Data System – II ..... GDSII
Integrated Circuit IC
Intellectual Property ..... IP
Key Differentiating Circuit ..... KDC
Key Programmable Circuit ..... KPC
Key Programmable Gate ..... KPG
Learned Clause Avoidance Circuit ..... LCAC
Look Up Table ..... LUT
Low Output Corruptibility LC
Magnetic Tunnel Junction ..... MTJ
Multiplexer ..... MUX
Network on Chip ..... NoC
Non-Volatile Latch ..... NV-Latch
Non-Volatile Look Up Table ..... NV-LUT
Power/Performance/Area/Security ..... PPA/S
Primary Input ..... PI
Primary Output ..... PO

Probably Approximately Correct ..... PAC
Random ..... RND
Register-Transfer Level ..... RTL
Reverse Engineer ..... RE
SAT Circuit ..... SATC
Scanning Electron Microscopy ..... SEM
Set of Candidate Keys ..... SCK
Set of Valid Keys ..... SVK
Single Probability Skew ..... SPS
Spin Transfer Torque ..... STT
Static Random-Access Memory ..... SRAM
System on Chip ..... SOC
Tenacious and Traceless Locking ..... TTL
Third Party Intellectual Property 3PIP

ABSTRACT

SECURITY AND COMPLEXITY ANALYSIS OF LUT-BASED OBFUSCATION: A COMPREHENSIVE STUDY

Gaurav Kolhe, M.S.

George Mason University, 2018

Thesis Director: Dr. Houman Homayoun

Logic locking and Integrated Circuit (IC) camouflaging are the most prevalent protection schemes that significantly thwart security threats, such as Intellectual Property (IP) piracy, hardware Trojans, reverse engineering, counterfeiting, and overproduction. However, the state-of-the-art attacks, including Boolean Satisfiability (SAT), Signal Probability Skew (SPS), and approximate-based attacks demonstrate the lack of having a comprehensive powerful defense scheme. Recent obfuscation schemes have employed reconfigurable logics, such as Look-up-Tables (LUTs) to prevent reverse engineering. However, existing LUT-based approaches focus on only a specific design factor such as replacement strategy or optimization metric such as SAT-hardness.

In this work, we study all proposed state-of-the art hardware obfuscation and attacks and forms a rationale for studying the LUT based obfuscation technique. We then propose a comprehensive analysis on LUT-based obfuscation based on all substantial

metrics that have considerable impact on design criteria, i.e. Power/Performance/Area (PPA) and Security (PPA/S). We performed a large design-for-security space exploration using four crucial factors for LUT-based obfuscation which has remarkable effect on PPA and security, namely (1) LUT technology, (2) LUT size, (3) number of LUTs, and (4) replacement strategy. Among these design parameters, the size of LUT is identified to have the most impact on making the obfuscation SAT resilient even for a weak random replacement strategy. A smarter replacement strategy helps to reduce the reliance on using large LUT to achieve SAT resiliency. Moreover, we found that while a clear trade-off exists between SAT resiliency, area and power overhead of LUT-based obfuscation, the delay trade-off can be substantially eliminated by using our proposed iterative security-driven design method which is non-disruptive to current standard ASIC design flow. Our experimental results indicate that for the studied designs, less than two iterations are sufficient to enhance the PPA/S along with eliminating the delay overhead with the proposed iterative security-driven PPA optimization. Our empirical results further demonstrate that increasing the size of LUTs from 2 to 8 provides SAT-resiliency with only less than 1% of gates replaced with LUTs.

INTRODUCTION

Hardware Security

Massive integration of billions of transistors on a single integrated circuit (IC) led to improved functionality, but at the cost of design complexity. In addition, increasing the manufacturing costs and heterogeneity in IC components are leading towards integrating multiple design units manufactured by different vendors onto one single IC [1]. Despite the cost-effectiveness, using designs from untrusted vendors for fabrication further exacerbates the security concerns [2,3] such as IC reverse engineering (RE) and insertion of hardware Trojans [4]. Such a security breach can occur at any phase of IC design and manufacturing process, such as during design, at foundry, SOC integration, or even at the end user. For example,

3PIP (Third Party Intellectual Property): Any rogue employee in 3PIP design house with access can sell, modify, overuse, or reverse engineer an IP as the design is open and visible in this phase.

SoC and DFT inserter: A malicious entity in SoC or DFT insertion phase with access to unencrypted IP can also sell, modify, or reverse engineer the design.

Untrusted foundry: Any adversary with access to the final GDSII file of the IC design might overproduce the design or sell it to a third party. They might reverse engineer the design to retrieve higher level description to exploit vulnerabilities.

Assembly, distributor, and user: An attacker in assembly and distribution stage or an end user does not have access to the original design. However, they might reverse engineer the fabricated IC. Hence, the active participation of various external agents in the design and manufacturing how has made the entire process highly vulnerable to various security threats.

Although IC reverse engineering 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 reverse engineer 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. [5]

IC Security Threats

A Figure 1 Threats on third party IP in supply chain demonstrate the various threats that exacerbates hardware security at various levels. Below we explain each of these threats.

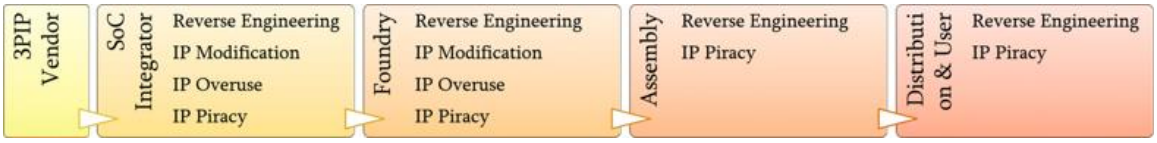


Figure 1 Threats on third party IP in supply chain

IC Piracy

This is a scenario where the IP is illegally used and/or copied without paying the lawful royalty to the IP vendor [6].

IC Overuse

In this scenario, the manufacturing fab illegally copies and reverse engineers the design database of an IC sent for fabrication to manufacture illegal copies (“clones") of the IC [7].

IC Modification

The design can be modified in the design house during the design phase or in the fab by malicious insertion, deletion or modification of circuits, referred to as Hardware Trojans, which cause the IC to deviate from its intended functional behavior during deployment [8, 9]. Typically, these Trojan circuits are stealthy by design, which makes it extremely challenging to detect them by traditional post-manufacturing testing [10].

Secret Information Leakage

Though the “deploy and monitor" step of the design is completely trustable, it has been shown that secret information can be extracted by an adversary from secure ICs with cryptographic functionality [11]. Such threats increase with increasing controllability and observability of the internal nodes of the circuit because of widespread adoption of “Design for Testability" (DfT) techniques in modern ICs. It is worth noting that the “design" stage

itself is designated as one of the partially insecure stages of the entire flow. This takes into consideration the possible presence of untrusted personnel in the design house with access to the design, who might sabotage the design to serve other interests.

Concept of Obfuscation

Obfuscation is a technique that makes understanding or reverse engineering of a design difficult. To protect hardware IP from these threats, the design needs to be unintelligible, even in decrypted form. Hardware obfuscation provides the option to effectively hide and disable the design, but still facilitate structural testing and static/dynamic parameter analysis [1,12,13]. This convenience makes obfuscation a desirable method for security and an active field of research.

As shown in Figure 2 Hardware obfuscation obscuring logic function to prevent hardware attacks, obfuscation methods facilitate to create look-alike logic, uncertain logic (until post-fabrication configuration), and key-controlled logic. An attacker who does not have sufficient knowledge of the applied specific obfuscation procedure could misunderstand the logic function of the module of interest. For instance, the NOR gate is recovered as a NAND gate; an XOR function is recognized as an addition.

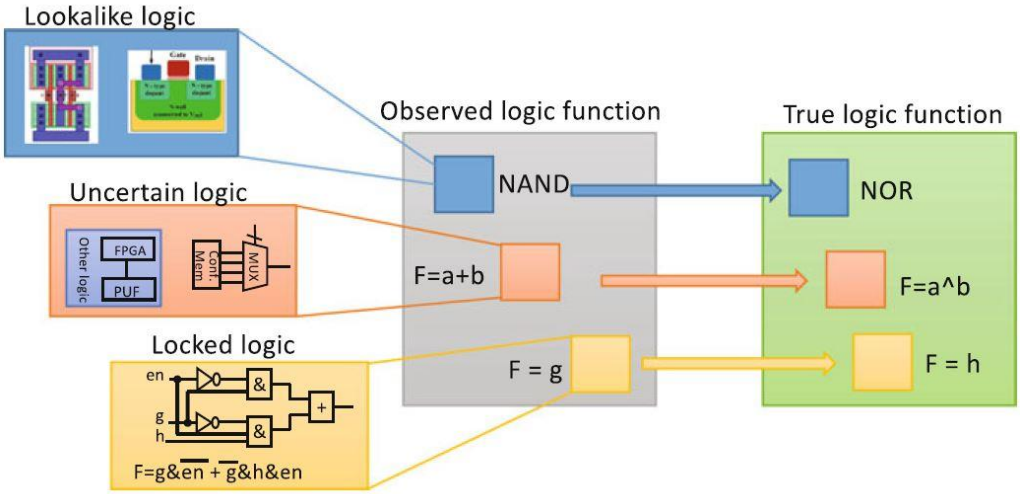


Figure 2 Hardware obfuscation obscuring logic function to prevent hardware attacks

The application of hardware obfuscation has been demonstrated at different abstraction levels in integrated circuits and systems. Hardware obfuscation techniques can also be classified based on whether they are combinational or sequential in nature. In this work, we focus on combinational obfuscation. This type of hardware obfuscation is realized by adding combinational components only to the combinational parts of a hardware design. We summarize different hardware obfuscation at various level in Figure 3 Hardware Obfuscation at different abstraction levels. The following subsections introduce the key idea of the methods at each level in details.

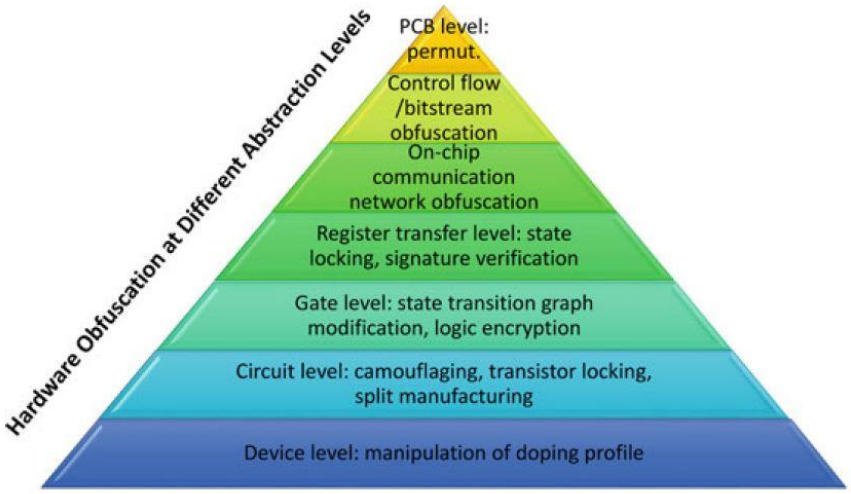


Figure 3 Hardware Obfuscation at different abstraction levels

Chip-Level Obfuscation:

Chip-level obfuscation includes the hardware modification that we make on the device, circuit, gate, and register-transfer levels to confuse attackers who intend to insert hardware Trojans.

Device-Level Obfuscation

Device-level obfuscation is to disguise the real function of a device by introducing controllable faults, such as stuck-at and delay faults. Without knowing the exact details of the device implementation procedure, a reverse engineer may misinterpret the device (and the associated circuits) function based on the superficial understanding from, for instance, the SEM images.

Circuit-Level Obfuscation

The circuit-level obfuscation techniques are divided into two categories: camouflaging layout and transistor locking. Camouflaging at circuit layout is used to thwart the reverse engineers from successfully reading the layout and thus recovering the transistor-level schematic for that circuit. The camouflaging techniques have been applied to two-dimensional (2D) and three-dimensional (3D) ICs. Transistor locking is another type of obfuscation (in a general sense), which changes the logic gate functionality by muting some transistors in the schematic [14,15].

Gate-Level Obfuscation

The gate-level obfuscation methods hinder the hardware Trojans from affecting the module under protection by modifying the state transition function of a circuit such that the attacker is not able to reach the real power-up state. Thus, the inserted hardware Trojan is either in the states belonging to the authentication mode or in the normal operation mode. As the attacker does not know the correct obfuscation key, the Trojans in the latter mode will never be triggered. There are many Gate-level obfuscation which have been proposed both for combinational and sequential circuit. We have focused mainly on combinational gate level obfuscation and various obfuscation methods are discussed further in other section.

Register Level Obfuscation

The basic idea is to transform the RTL core into control and data flow graph (CDFG) and then integrate a well obfuscated finite state machine (FSM) of special structure, referred as “Mode-Control FSM”, into the CDFG in a manner that normal functional behavior is enabled only after application of a specific input sequence. [16]

On-Chip Network Communication Obfuscation

The hardware Trojan mitigation method in [17] aims to detect and mitigate the hardware Trojan attacks that (1) modify the flit type, (2) change the legal packet destination address to an unauthorized one, and (3) sabotage the integrity of a packet. The main consequence of the hardware Trojan targeted in [17] is the NoC bandwidth depletion. Hence, to overcome such type of attacks, the NOC modules are obfuscated.

Other Methods

Split manufacturing methods [18, 19, 20, 21, 22] obfuscate the design by dividing a circuit into multiple tiers, which are sent to different foundries for fabrication. As each foundry does not have the complete design, attackers from the untrusted foundry have limited understanding on the entire design and may not be able to insert effective hardware Trojans to the design portion they have. In recent years, researchers have realized that split manufacturing may not be as necessarily secure as expected [23, 24 ,25, 26].

After, the obfuscation re-synthesis is necessary. Among the methods that we have looked, there are some obfuscation methods which are done after the first synthesizing step,

where one of the example being layout obfuscation. Figure 4 Integration of hardware obf. in all levels of abstraction through design flow shows the integration of these different obfuscation techniques at different abstraction levels and the modified design flow. Verification is necessary to be carried out before and after obfuscation, to ensure the implementation has been properly done and the original functionality is preserved when unlocked.

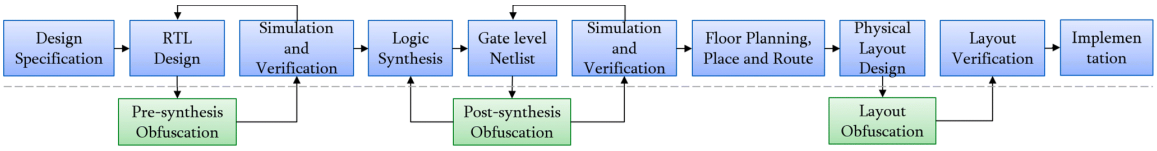


Figure 4 Integration of hardware obf. in all levels of abstraction through design flow

Recent Work

Hardware design-for-trust (DFTr) mechanisms water marking, IC metering, IC camouflaging, split manufacturing, and logic locking [27,28,29,30,31], camouflaging and logic locking are the techniques which have shown a better resiliency to many of the existing potential hardware-design stealing or attacking techniques [32]. Figure 5 Security threats in a modern IC life-cycle, and corresponding solution shows the security threats at various stages of the IC design lifecycle along with different obfuscation solutions that have been proposed.

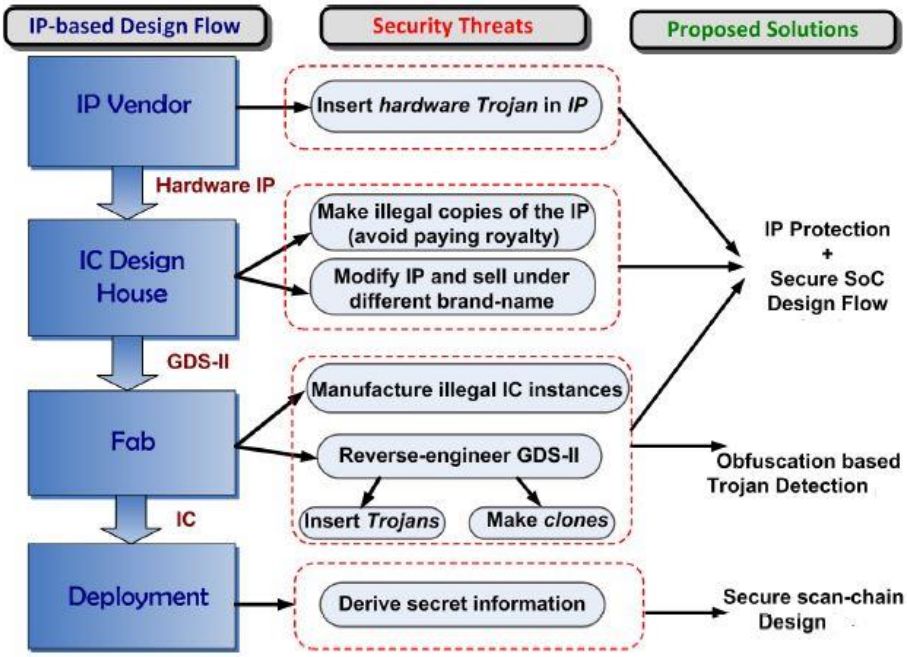


Figure 5 Security threats in a modern IC life-cycle, and corresponding solution

Increasing interest of the research community in logic locking and camouflaging persuaded a major CAD tool provider, i.e. Mentor Graphics, to release Trust Chain which is a CAD framework that supports logic locking and camouflaging [33]. Numerous logic locking, and camouflaging schemes have been proposed to thwart these security concerns. Due to the invent of Boolean Satisfiability (SAT) based attack, which is one of the powerful de-obfuscation/de-camouflaging techniques, the correct key can be extracted from most of the logic locking and camouflaging schemes in few minutes [34, 35]. Hence, this is still an active area of the research.

Outline of Proposed Research

Defense vectors are defined at various level, but Gate level obfuscation is an effective way to curb many attack threats such as overproduction, IC piracy, Trojan insertion etc., However, A Boolean Satisfiability (SAT) based attack breaks all the existing combinational logic locking techniques, and hence forms the priority for addressing first.

Some of these schemes use reconfigurable logics based on hardware reconfiguration and/or transformation, such as Look-Up-Tables (LUTs) [36, 37, 38]. The obfuscation using LUT also serve the purpose of hindering various IC threats while having significant resiliency against the SAT attack and the removal Attack. The LUT also has an interesting aspect of reconfigurability & hence replacing a gate with LUT means there exist 2^{2^N} possibilities the LUT can be configured and this increases the search space for the SAT Solver. Hence, we evaluate the LUT Based obfuscation in this paper and efficacy of integrating it into the circuit.

The prior LUT-based obfuscation schemes have used diverse approaches like increasing the number of LUTs [38] or using different replacement strategies [39], however, we demonstrate that despite considering all substantial and effective factors for LUT-based obfuscation, security can be compromised.

In this work, we comprehensively explore the design space of LUT-based obfuscation to show how it is possible to achieve the most powerful resiliency against state-of-the-art attacks like SAT attack, while trading power-performance (delay)-area (PPA). Based on our design space exploration, we demonstrate that four factors play a key role for security enhancement and PPA optimization in LUT-based obfuscation. These four factors

consist of (1) technology of the LUT, (2) the number of LUT inputs (LUT size or scale up), (3) the number of cells replaced by LUTs (LUT count or scale out), and (4) the replacement strategy. With the aid of design space exploration (i.e. investigate all possible combinations for these factors), we provide a LUT-based obfuscation, which is not only resilient against SAT attack but also has permissible PPA overhead. In contrast to [39], which uses combination of number of LUT factor and replacement strategy to achieve the resiliency, we show that the LUT size has significantly more impact on SAT solver execution time. In fact, our investigations show that the LUT size (number of LUT inputs) plays a crucial role in enhancing the resiliency of the obfuscation scheme, even in presence of weak replacement policy such as random placement of LUTs.

An iterative design flow for PPA optimization towards reducing the delay and enhancing security is introduced in this work and showed be to non-disruptive to standard CMOS ASIC design flow which is important for the designability and manufacturability of the solution.

The main contributions of this work are outlined in five-fold manner, as follows:

(1) **Studying Existing literature:** Studying the existing obfuscation & attack techniques to evaluate robustness of existing obfuscation scheme.

(2) **Performing Design Space (factors) Exploration:** We explore the impact of LUT-based design parameters on the security resiliency and PPA metrics. Identifying the Size of LUTs (scale up) as the Most Important Factor for SAT Resiliency

(3) **Increasing the Size of LUTs (scale up) as the Most Important Factor in case of SAT Resiliency:** We show that using enlarged LUTs with extra arbitrary inputs,

called dummy inputs, is the most effective factor for amplifying the resiliency of LUT-based obfuscation against SAT attack, even for weak replacement policy such as random insertion of LUTs.

(4) **Proposing Iterative Security-driven Design Flow:** Regardless of chosen LUT obfuscation factors, we introduce a simple iterative security-driven PPA optimization design flow, which allows maximizing the efficiency for any type of LUT-based obfuscation with several factors.

(5) **Proposing SAT-resilient LUT-based obfuscation Solutions:** Leveraging comprehensive design space exploration studied in this work, we demonstrate what combinations of the four factors provide SAT resiliency, and which solution is the best among all SAT-resilient combinations.

A sub-optimal solution, however better than existing approaches is shown with the `proposed STT-LUT based obfuscation and iterative security-driven design flow. While we found that a clear trade-off exists between SAT resiliency, area and power overhead of LUT based solutions, the delay trade-off can be substantially eliminated by using our proposed iterative optimization methodology enabled by the state-of-the-art industry class synthesis tools optimization engine. As a panacea for high resiliency and optimal PPA, a customized or camouflage customized LUT architecture can be deployed.

BACKGROUND

An overview of logic obfuscation, SAT attack, and challenges are presented here. We'll present the defense and the corresponding attack.

Logic Obfuscation and De-Obfuscation

In logic obfuscation process, the functionality of the design is concealed by inserting additional logic gates including key programmable XOR/XNOR gates, key-programmable MUXes for interconnections, avoiding netlist extraction after delayering by adding ambiguity. The strength of traditional logic obfuscation schemes is based on the location of inserted/replaced gate according to gate selection algorithm [31, 40, 41]. Below are some of the obfuscation techniques that were proposed.

Strong Logic Obfuscation

Strong logic obfuscation hinges on inserting key-gates with complex interferences among them. Logic obfuscation is weak when the inserted key-gates are isolated, or their effect can be muted. If mutable gates are employed, then 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. In such insertions when the key size is greater than 100, it will take several years for an attacker to determine the key bits. The Figure 6 Ideology of

Strong Logic Obfuscation shows the Strong Logic Obfuscation. The attacker cannot propagate the effect of key bits K1 and K2 individually to the outputs. Hence, the attacker has to brute force to determine the values of K1 and K2. [40]

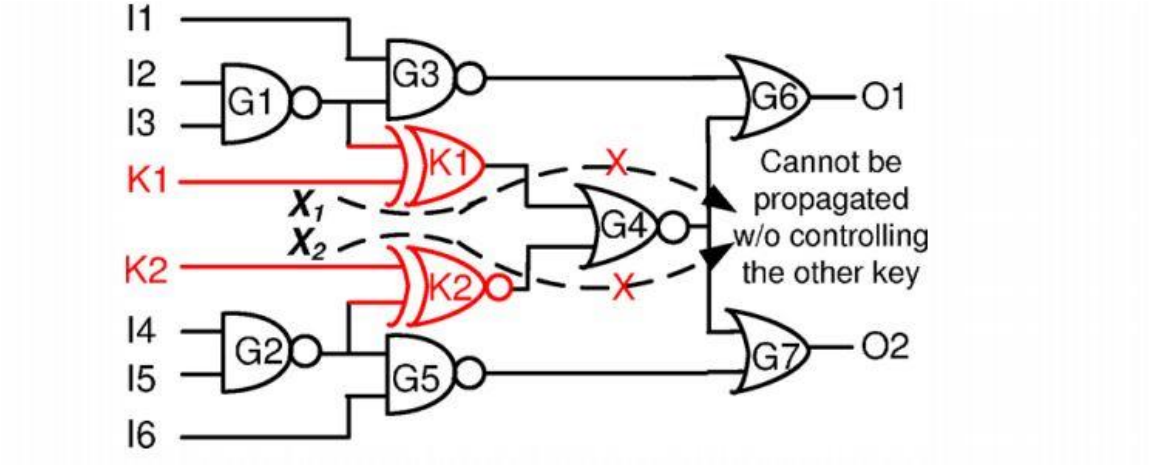


Figure 6 Ideology of Strong Logic Obfuscation

Camouflaging:

IC Camouflaging in layout obfuscation technique. 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 7 Standard Cell Versus Camouflaged Cell the layout of regular NAND cell Figure 7(a) and NOR Figure 7 (b) cell look different and are hence easy to reverse engineer. However, the layout of camouflaged NAND cell Figure 7(c) and NOR cell Figure 7(d) look identical and are difficult to differentiate [42, 43, 44, 45]. When deceived into incorrectly interpreting the functionality of the camouflaged gate, the attacker may obtain a reverse engineered netlist that is different from the original. The

netlist obtained by an attacker is the deceiving netlist where the functionality of the camouflaged gates are arbitrarily assigned.

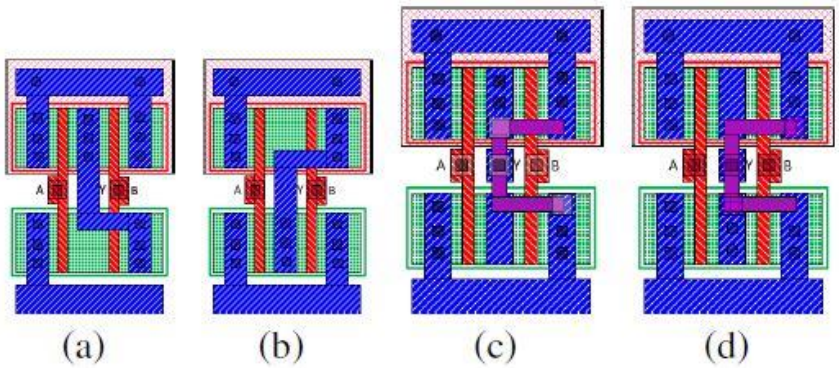


Figure 7 Standard Cell Versus Camouflaged Cell

Figure 8 Generic Camouflaged Layout shows the layout of a camouflaged cell that can function as either 2-input XOR, NAND, or NOR. The sets of true and dummy contacts to implement distinct functions with the camouflaged gate are listed in List of true and dummy contact to realize different function using the camouflaged layout shown in figure 8Table 1. Similarly, one can design a camouflaged cell that can function as either 2-input XNOR, NAND or NOR. [55]

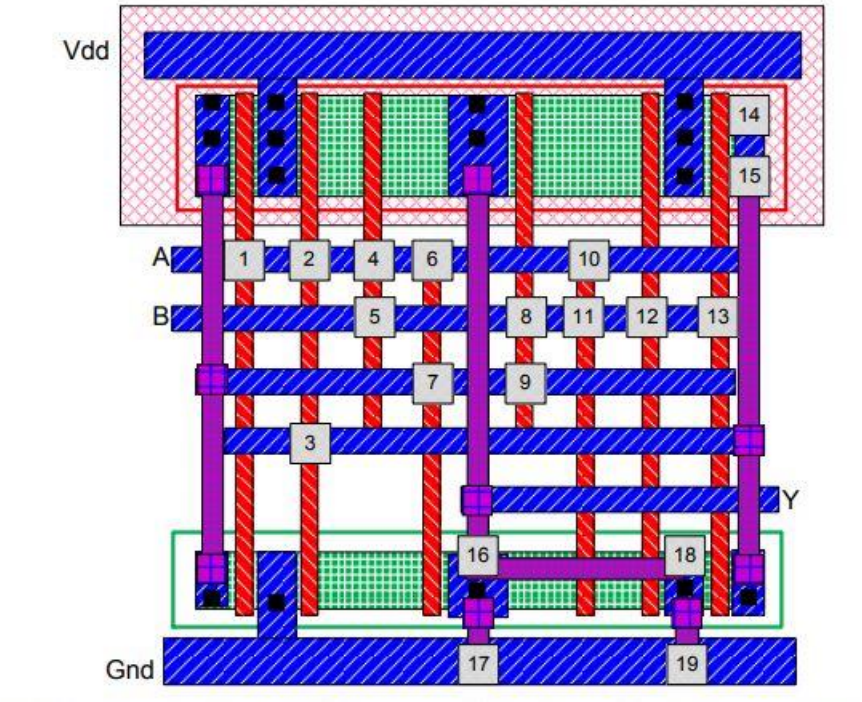


Figure 8 Generic Camouflaged Layout

Table 1 List of true and dummy contact to realize different function using the camouflaged layout shown in figure 8

Function	Contacts	
	True	Dummy
NAND	2, 4, 6, 8, 11, 12, 16, 17	1, 3, 5, 7, 9, 10, 13, 14, 15, 18, 19
NOR	2, 5, 6, 11, 12, 18, 19	1, 3, 4, 7, 8, 9, 10, 13, 14, 15, 16, 17
XOR	1, 3, 4, 7, 9, 10, 12, 13, 14, 15, 18, 19	2, 5, 6, 8, 11, 16, 17

SAT Attack

In computer science, the Boolean satisfiability problem (sometimes called propositional satisfiability problem and abbreviated as SATISFIABILITY or SAT) is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. The concept of SAT was introduced to defeat the obfuscation schemes that were discussed so far.

SAT attack iteratively eliminates the incorrect keys based on specific input patterns, called Distinguished Input Patterns (DIPs) [46, 47]. SAT attack finds one DIP in each iteration which produces two different outputs for two different keys. As its name implies, each DIP can distinguish between keys. Based on this iterative based structure SAT attack can find and eliminate all incorrect keys within few minutes, even for large circuits.

A SAT solver takes a Boolean function in Conjunctive Normal Form (CNF) as input and finds a valid assignment for input variables to satisfy the function. To attack an obfuscated netlist using a SAT solver, a working copy of the chip and its obfuscated netlist is required. The adversary can acquire the working chip after it is unlocked by the manufacturer and shipped to the market and could gain access to the obfuscated netlist by means of RE. In case of supply chain adversary, the obfuscated netlist is readily available to the attacker. Then, the obfuscated netlist should be transformed into a circuit SAT problem. This process is explained next:

Let us refer to the functional black-box copy of the obfuscated circuit as C_F . The C_F is used to find the correct output for any given input. When using K keys, random assignment of key could create at most 2^K instances of a circuit. Similar argument applies

to camouflaged cells, where each of K camouflaged gates could assume one of the M different possibilities (for simplicity, let us consider M = 2). Let us denote obfuscation scheme obtained by means of using K keys or obfuscated gates by K-obfuscation. A circuit C with N_x inputs that is subjected to K-camouflaging could be represented with an equivalent C_K circuit with N_X + K inputs. Let us denote the circuit C with input X and output Y by C (X, Y) and its K-obfuscated netlist by C (X, K, Y). If the correct set of keys $\widehat{K}$ = (k_0, k_1, ... , k_{K-1}) is applied to the obfuscated circuit, for every input the obfuscated circuit reduces to the original circuit C(X, $\widehat{K}$, Y_K) , ≡≡ C(X, Y).

For a SAT attack the key signals in C (X, K, Y) should be available as input. Hence, obfuscation cells should be represented as Key-Programmable Gate (KPG), where insertion of the correct key converts them to the correct gate. The cells used for obfuscation could be divided into two categories: (1) key-controlled gates [31, 48] in which the key is an input signal (e.g. XOR, MUX based obfuscation). (2) keyless-gates [39, 46] where functionality is hidden in the ambiguous structure or by use of internal memory elements (e.g. camouflaged gates and LUTs). When using key-controlled gates, the key is stored in an internal memory or a burned fuse. Hence, in a reverse-engineered netlist the key inputs could be identified by tracking their connectivity to memory/fuse elements. To prepare the C (X, K, Y) netlist, the memory/fuse element is removed, and key inputs are connected to input port(s).

When using keyless-gates, the gate must be transformed to a key-programmable gate before invoking a SAT attack. For a L-input LUT, the number of functional possibilities is 2^{2^N}. To build a KPG for a LUT, the circuit illustrated in Figure 9 is

deployed. The inputs to the LUT are connected to the select lines of the S-MUX and keys are the select lines of B-MUXes. Then, each key is connected to an input port adding 2^N keys to the C (X, K, Y).

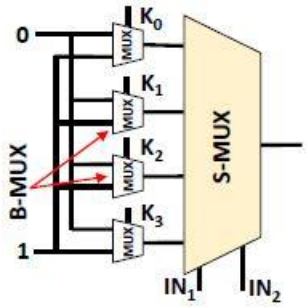


Figure 9 Converting a LUT to a KPG

A camouflaged cell relies on hiding the gate functionality by keeping the structure of several gates similar. Even in the best camouflaging cells, the number of gate possibilities is limited, and it could be treated similarly to programmable cells, where the camouflaged cell is replaced by a MUX and each of the gate possibilities is fed to a different input of the MUX, while using the select lines of the MUX as key inputs that are routed to the input pins of the C (X, K, Y).

Before invoking the SAT solver, every key input combination is considered as a candidate key. Let's denote the Set of Candidate Keys by SCK. If we 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 is denoted as a Discriminating Input (DI) [35]. This is because the selected input can prune the SCK and find at least one incorrect key that is removable from SCK. In

addition, each time a new DI is found, the SCK search space for function FDI should be updated. This could be achieved by forcing the FDI to check each pair of new keys K1 and K2 against all previously founds DIs. A Complete-DI-set is a set of DI inputs that reduces the SCK to the Set of Valid Keys (SVK). SCK reduces to SVK when we no longer can find a DI using the updated FDI. At this point if a key is valid across the Complete-DI-Set, it is the correct key for all other inputs [35]. As suggested in Figure 10.b, a reverse-engineered netlist, where all obfuscated cells are replaced with KPG cells, is denoted by Key-Programmable Circuit (KPC). To build the FDI, 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 K1 and K2 are different. This circuit, as suggested in Figure 10.c is denoted as Key-Differentiating Circuit (KDC). The candidate keys in the SCK can produce the correct output for all DIs that have previously been discovered and tested on the KPC circuit. To test the keys for one DI, the circuit in Figure 10.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 DI and two key values. This circuit is denoted as DI-Validation Circuit (DIVC). To test the keys for all DIs, as illustrated in Figure 10.e, the DIVC circuit is duplicated D times, with D being the number of current DIs tested, and the output of all DIVC circuits ANDed together. The resulting circuit is a validation circuit for SCK set denoted as SCKVC. If two keys K1 and K2 produce the correct output for all previously tested DIs (SCKVC evaluates to true), but produce different results for a new input X_{test}, then X_{test} is a DI that further prunes the SCK. This, as illustrated in Figure 10.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 K1 and K2 that produce two different results for an input X_{test}, and SCKVC circuit has confirmed that both K1 and K2 belong to SCK set. Hence, the input X_{test} is yet another DI. Each time a new DI is found, the SCKVC should be updated by adding yet another DIVC circuit for testing the newly discovered DI. 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 a correct key for the circuit. On the SAT solver side, every time the SAT solver is executed, it learns a new set of conflict clauses. It is essential to store the learned clauses and use them in the next invocation of the SAT solver to prevent SAT solver from re-learning these clauses. Hence, as illustrated in Figure 10.f a Learned-Clause Avoidance Circuit (LCAC) is added to the SATC to check for the occurrence of learned conflict clauses.

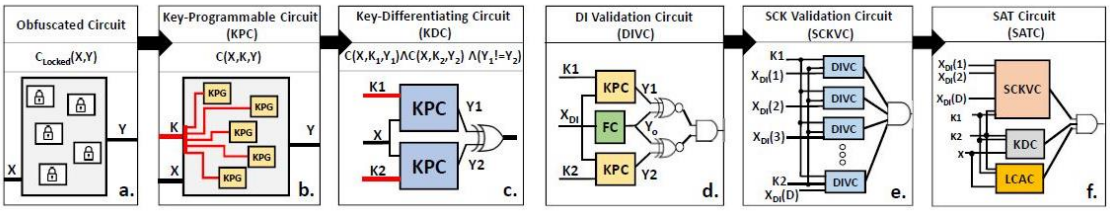


Figure 10 SAT Attack Process

The SAT attack, as illustrated in Algorithm for SAT Attack, follows the SATC construction process explained previously. In the first iteration, the SCKVC circuit does

not contain any logic, since there is no previously tested DI. Hence, it is set to 1 (true). The KDC circuit is simply built based on its definition by using the equation in Figure 10.c. The SATC circuit is constructed by using an ANDing the KDC and SCKVC circuits. SATF function is a call to SAT solver. Considering the to-be-assigned variables in SATC circuit are X, K1 and K2, the SAT solvers return an assignment to these variables and a list of conflict clauses (CC) learned during SAT execution. SATF return 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 DI is found. Hence, the SATC circuit should be modified (lines 7-10). The parts of SATC circuit that is updated are the SCKVC and LCAC. After finding each DI, an additional DIVC is added to SCKVC to validate the keys generated in the next invocation of SAT solver with respect to the newly found DI. In addition, the newly 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 DI. Finally, the SATC circuit is formulated at line 10 for the next invocation of SAT solver. The while loop is executed until no other DI is found. At this point, any key in the SCK set is a 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 satisfy all previous constraints imposed by previously found DIs. A simple call to a SAT solver at this point returns a correct key assignment. If the SAT solver does not return a valid key, it means the obfuscation, locking, or camouflaging technique is invalid. Note that the SAT attack in each iteration, as explained in Algorithm for SAT Attack and illustrated in Figure 12, 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. This way, SAT solver efficiently breaks all the obfuscation schemes. [49]

1: $KDC = C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge (Y_1 \neq Y_2)$;
2: $SCKVC = 1$;
3: $SATC = KDC \wedge SCKVC$
4: $LCAC = 1$
5: **while** $((X_{DI}, K_1, K_2, CC) \leftarrow SAT_F(SATC) = T)$ **do**
6: $Y_f \leftarrow C_F(X_{DI})$;
7: $DIVC = C(X_{DI}, K_1, Y_f) \wedge C(X_{DI}, K_2, Y_f)$;
8: $SCKVC = SCKVC \wedge DIVC$;
9: $LCAC = LCAC \wedge CC$
10: $SATC = KDC \wedge SCKVC \wedge LCAC$;
11: **end while**
12: $KeyGenCircuit = SCKVC \wedge (K_1 = K_2)$
13: $Key \leftarrow SAT_F(KeyGenCircuit)$

Figure 11 Algorithm for SAT Attack

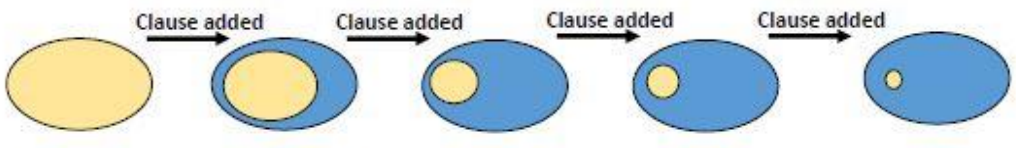


Figure 12 Reduction in SCK Set as SAT Progresses

SARLock

There was a need to thwart the SAT attack and hence SAT-Attack Resistant Logic Locking (SARLock) technique was introduced. The proposed technique adds only a few XOR/XNOR gates; however, the attack effort increases exponentially with the number of key bits. Figure 13 defines the SARLock mechanism. Only when the input matches the applied key which is incorrect, the circuit produces the wrong answer, else the circuit works

just like the oracle, as shown in Figure 14. Hence, in each SAT iteration only 1 key is eliminated, making the attack exponential with the key size, because SAT solver has to exhaustively search all DIPs.

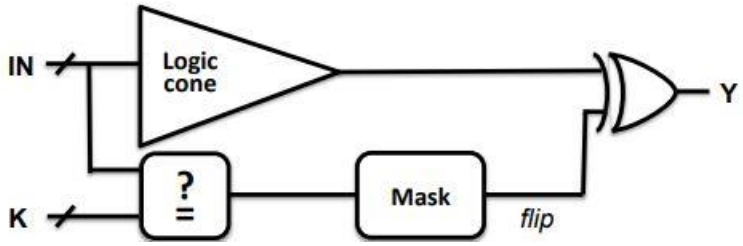


Figure 13 SARLock circuit

					Output Y for different key values							
No.	a	b	c	Y	k0	k1	k2	k3	k4	k5	k6	k7
0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0	0
3	0	1	1	1	1	1	1	0	1	1	1	1
4	1	0	0	0	0	0	0	0	1	0	0	0
5	1	0	1	1	1	1	1	1	1	1	1	1
6	1	1	0	1	1	1	1	1	1	0	1	1
7	1	1	1	1	1	1	1	1	1	1	1	0

Figure 14 Resisting the SAT attack by Controlling discriminating ability of input patterns.

SARLock can be used in conjunction with existing logic locking techniques to protect against a wide spectrum of attacks. [54]

Anti-SAT

Another type of attack called the Anti-SAT was designed to enhance the security of existing logic locking techniques against the SAT attack. Figure 15 a and b, illustrate two configurations of the proposed Anti-SAT block, referred to as type-0 Anti-SAT and type-1 Anti-SAT. They consist of two logic blocks g and g´, which share the same set of inputs $\hat{X}$ = ($X_1 \ldots X_n$). The functionalities of g and g´ are complementary. A set of keygates (XORs) are inserted at the inputs of two logic blocks, denoted as $\hat{K}_{11}$ = ($K_1 \ldots K_n$) and $\hat{K}_{12}$ = ($K_{n+1} \ldots K_{2n}$). Hence the key-size is 2n. The output of g and g´ are fed into an AND2 gate (for Fig. 4(a)) or an OR2 gate (for Figure 15 b) to form the final single-bit output Y. As a result, we have Y = g ($\hat{X} \oplus \hat{K}_{11}$) ^ g ($\hat{X} \oplus \hat{K}_{12}$) for type-0 Anti-SAT and Y = g ($\hat{X} \oplus \hat{K}_{11}$) ∨ g ($\hat{X} \oplus \hat{K}_{12}$) for type-1 Anti-SAT. Type-0 Anti-SAT always outputs 0 if key values are correct while Type-1 Anti-SAT always outputs 1 if key values are correct. Figure 15 c shows the integration of Type-0 Anti-SAT block into a circuit.

one basic property of Anti- SAT block is that when the key vector is correctly set, the output Y is a constant. Specifically, given a correct key, Y always outputs value 0 for type-0 Anti-SAT (Figure 15 a) and always outputs value 1 for type-1 Anti-SAT (Figure 15 b). Otherwise, when a wrong key is given, Y can output either 1 or 0 depending on the inputs ~X. This property enables it to be integrated into the original circuit. [52]

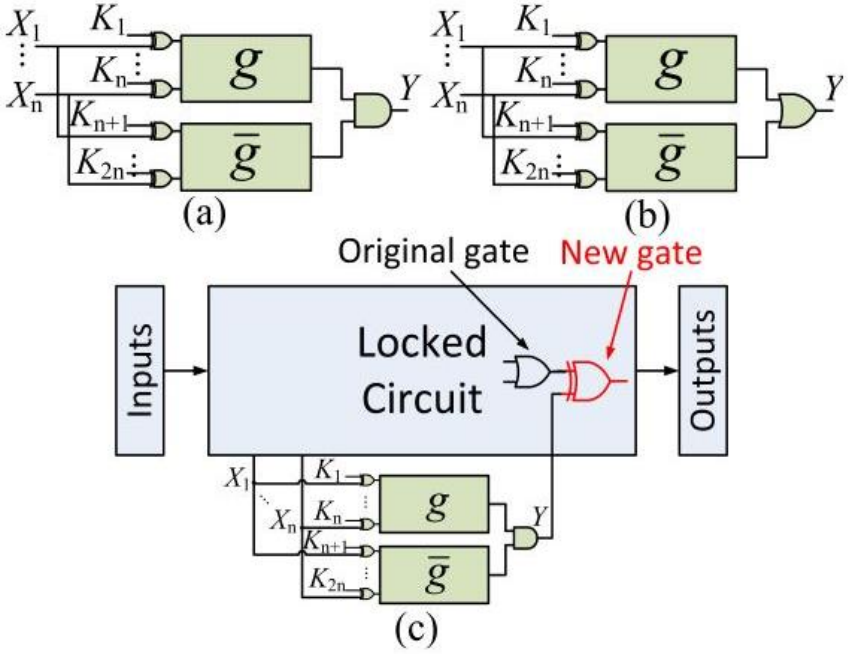


Figure 15 Anti-SAT block configuration

In the Anti-SAT obfuscation technique, the number of iterations needed by the SAT attack to decipher the correct key is lower bounded by 2^n, where n is the length of key.

Signal Probability Skew

As an attack to the above proposed scheme, signal probability skew (SPS) attack was developed that not only defeats Anti-SAT but also conquests over SARLock within minutes. SPS attack is highly scalable to large circuits. This attack is more effective with increasing key size.

The main vulnerability of Anti-SAT is that it is incorporated into the netlist at a single point, where its output Y is XORed with an internal net. Therefore, Anti-SAT defense relies on various obfuscations that make the identification of the block and its

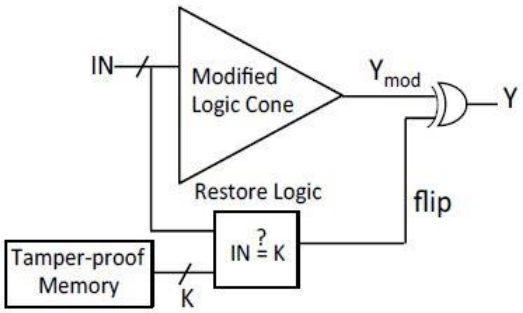
output difficult. At the same time, SAT attack resilience is ensured by having a skewed p value irrespective of all the structural and functional obfuscations. This basic construction principle inevitably leads to structural traces that help identify Anti-SAT block output in a given netlist i.e. The two complementary blocks of Anti-SAT produce oppositely skewed signals that converge at a gate, whose output is Anti-SAT output Y that is integrated into the netlist.

Absolute value of probability skew close to 1 indicates that the two inputs of the gate G exhibit highest skews with opposite polarity. This property of gate G distinguishes it from the rest of the gates in Anti-SAT block. SPS attack on a logic encrypted circuit with Anti-SAT block comprises computing the SPS of all the gates in the circuit. The gate with the highest SPS, i.e., a gate with oppositely skewed inputs is the suspect gate G, the output gate of Anti-SAT block. After finding this gate, Anti-SAT block can be removed from the circuit.

In SARLock circuit, shown in Figure 13, the original logic cone is implemented intact without any modifications, which makes it vulnerable to removal attacks. An attacker must isolate the protection circuitry comprising of an XOR, comparator and mask block; he/she can then remove the protection circuitry and extract/pirate the original IP. The comparator is functionally composed of XNOR gates and an AND tree, which can be easily identified using existing AND-tree identification algorithms, or the SPS. Upon the removal of the protection logic, the original function O = F(I) is retrieved. Due to the invent of SPS attack the hardware security was again challenged to propose better obfuscation schemes. [53]

Tenacious and Traceless Locking:

TTL is an extended version of the SARLock. TTLock modifies the original logic cone by inverting the response to one protected input pattern, while an additional inversion introduced by TTLock restores the correct functionality only for the correct key. Even though the TTLock logic can be identified via a signal-tracing attack, its removal will still leave the remaining logic different than the original one, thwarting removal attacks. Figure 16 shows the TTL implementation. At each iteration, when input is equal to the wrong key, we observe the wrong output, whereas when the correct input pattern is applied, all outputs are wrong for any given key. The restore logic then corrects the output for the correct key. Hence when k6 is applied, you get the correct output.



Y_{mod}	IN	k0	k1	k2	k3	k4	k5	k6	k7
✔	0	✖	✔	✔	✔	✔	✔	✔	✔
✔	1	✔	✖	✔	✔	✔	✔	✔	✔
✔	2	✔	✔	✖	✔	✔	✔	✔	✔
✔	3	✔	✔	✔	✖	✔	✔	✔	✔
✔	4	✔	✔	✔	✔	✖	✔	✔	✔
✔	5	✔	✔	✔	✔	✔	✖	✔	✔
✖	6	✖	✖	✖	✖	✖	✖	✔	✖
✔	7	✔	✔	✔	✔	✔	✔	✔	✖

Figure 16 Proposed TTLock architecture and it's working

SAT attack takes exponential time to find the key, while the SPS and other removal attack fails when TTL obfuscation is used. [56]

AppSAT

AppSAT attack is an approximate de-obfuscation algorithm which is based on the SAT attack and random testing models. It uses probably-approximately-correct (PAC) settings with SAT attack. The SAT attack begins by satisfying the miter circuit with a DI, X_0. Subsequently, the oracle is queried with X_0 and a constraint is added to the SAT-formula. Whereas for the exact SAT attack the algorithm continues to find DIs until no more DIs can be found while the approximate SAT(AppSAT) attack terminates the attack in any early step, i. AppSAT attack randomly queries input patterns to estimate error rate after every d DI queries, and the attack is terminated if error rate stays below a threshold for more than a certain number of times (settlement threshold).

Figure 17 shows the overall AppSAT algorithm flow. The SAT is used to find the DI and at the same time DI is used to query oracle. If certain number of queries are not achieved then those queries are stored as a DI, to further prunes SCK. Otherwise, error rate is measured with Random Queries and if the error rate is below certain limit, AppSAT terminates.

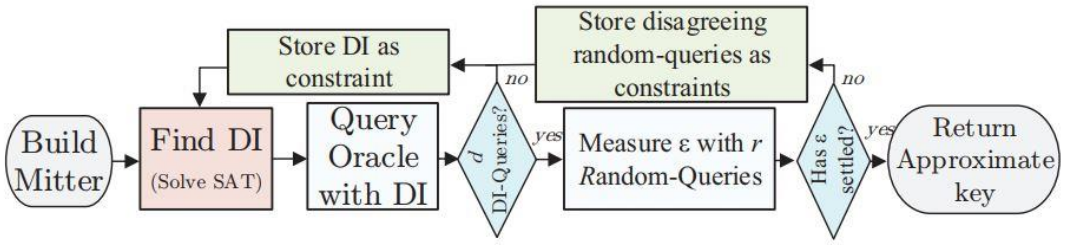


Figure 4: The overall AppSAT algorithm flow.

Figure 17 AppSAT algorithm flow

Low output corruptibility of point-function obfuscations is susceptible to removal attack hence, these methods are often combined with high corruptibility obfuscations to form the compound obfuscation scheme. Adding a high corruptibility obfuscation to the point-function schemes does not contribute to the overall security. The AppSAT attack is capable of deobfuscating the traditional portions of the compound obfuscation as if the point-function scheme was not present. This observation is key to utilizing the AppSAT attack to defeat various flavours of compound schemes such as SARLock, AntiSAT, TTL etc., [57]

Double DIP

Double DIP is another SAT-based decryption technique. It excludes at least two wrong keys each iteration, ensuring wrong keys in the part of traditional logic encryption being excluded without taking exponential iterations. SARLock minimizes the efficiency of SAT attack by exponentially increasing the required number of distinguishing input patterns, and only one incorrect key can be pruned each iteration. To avoid exponential

iterations, Double DIP only consider incorrect keys causing more than one incorrect input-output pairs, which disables the SARLock and gets the correct key for traditional logic encryption. [58]

Other Attack Vectors

Among the discussed attack and defense strategies there exist other methods to which are somewhat similar. For example, Camoperturb [59] is based on the SARLock but is a layout level obfuscation. Stripped Function Logic Level [32] is another obfuscation which is based on TTL. Few obfuscation schemes are based on adding cycles in the circuit, especially stateful/oscillating cycles [48, 50]. When SAT attack encounters these cycles, two states can happen: (1) SAT solver returns wrong key or (2) SAT solver will be stuck in an infinite loop. In fact, SAT solver cannot extract the correct key in cyclic-based approaches with stateful/oscillating cycles. However, CycSAT and SRCLock [30,28] shows that by considering some conditions and having some pre-processing, cyclic-based schemes can be decrypted. The Figure 18 below summarizes all the defense and attack in a gist.

Figure 18 Various Defense and Attacks proposed at Gate Layout

LUT-based Obfuscation

As mentioned previously, some obfuscation schemes use reconfigurable logic to prevent reverse engineering [36, 38, 39]. From the SAT attack perspective, each LUT is replaced with a (2+)-level MUX. Since each n-input LUT can provide all 2^{2^N} possibilities, replacing by MUXes lead to a $\log_2(n)$-level MUX-based structure. This indicates that

having large LUTs increase the depth of MUX structures, consequently improving the resiliency against diverse types of attacks. However, increasing the size of large LUTs imposes larger area and performance overheads. Another reason for considering the LUT based obfuscation is that it is resilient to removal attack as well. LUT based obfuscation strips the functionality in more abstract manner and prevents attacker from inferring the functionality that LUT has using the visual approach.

The existing LUT-based obfuscation schemes though have shown some resiliency against attacks, they did not consider all the effective factors for enhancing power-performance (delay)-area, security (PPA/S), which is crucial for hardware design. Furthermore, this also leads to having a sub-optimal solution. The work in [36] introduces gate selection policies for security and delay purposes. However, their proposed selection policy can be broken easily by SAT attack [35]. The [38] demonstrates that using spin-transfer torque (STT)-based LUTs is the best technology for employing LUTs in a design for PPA optimization. However, it has no significant security strength against state-of-the-art attacks. Also, LUT-lock [39] focuses on the number of LUTs, which has near to exponential hardness against SAT attacks even in some specific designs with topological structure. In contrast, the main aim of this work is to explore design space of LUT-based obfuscation based on different influential factors to show the effectiveness of each factor on PPA/S. Also, we show that the most effective factor that provides resiliency is the LUT size, even in presence of weak LUT replacement policy such as random. However, PPA overhead simply that for realizing LUT-based obfuscation, new customized LUT architecture equipped with camouflaging is required to significantly lower the overhead.

COMPREHENSIVE (PPA/S) ANALYSIS ON LUT-BASED OBFUSCATION

As the primary factors affecting the PPA are: (1) the technology of LUTs, (2) the number of LUTs replaced, (3) the number of inputs per LUT (LUT size), and (4) replacement strategy, we investigate the influence of these factors on PPA/S. These factors are influencing the PPA as well as security metric. Increasing the size of LUTs (scaleup), increasing the number of LUTs (scale out), and replacement strategies impose PPA overheads.

Impact of LUT Technology

Design and Integration of STT-Based LUT

As STT-based LUTs have shown higher PPA efficiency [14], we consider STT based LUT design and obfuscation in this work. STT technology not only can provide incredible features like (1) higher integration density than SRAMs, (2) high endurance and retention time, (3) near zero leakage, and (4) soft error resilience, it is also highly integrative in CMOS fabrication process [38]. Additionally, it provides on-die reconfigurability which enables to achieve high performance and security.

In addition, for STT-based LUTs since reconfigurable bits are stored in magnetic tunnel junction (MTJ) inserted between metal layers, the stored bits are highly susceptible to be lost during reverse engineering delayering process.

In LUT-based obfuscation, the design is partially mapped to LUTs. This results in a design implementation that is a hybrid of custom (ASIC) and programmable (FPGA) styles. The custom part of the design is implemented using the standard cell-based ASIC design flow. Since the ASIC standard cells are implemented in the static logic style, the resulting designs are static. This imposes a limit on the LUT design to have a static type interface for connection with the static ASIC standard cells. Also, the existing STT-LUT design styles in which a dynamic circuit such as a dynamic sense amplifier resides between the LUT inputs and the output is not suitable for this application [51]. Contrasting, we propose an STT-LUT design concept in which the path from the LUT inputs to the LUT output is a MUX, as shown in Figure 19(a). The MUX of the LUT is a 2^n to 1 (2^n : 1) CMOS MUX implemented in static style, that can be written as a synthesizable RTL code for automatic implementation and optimization by the logic synthesizer tool in the process of design compilation.

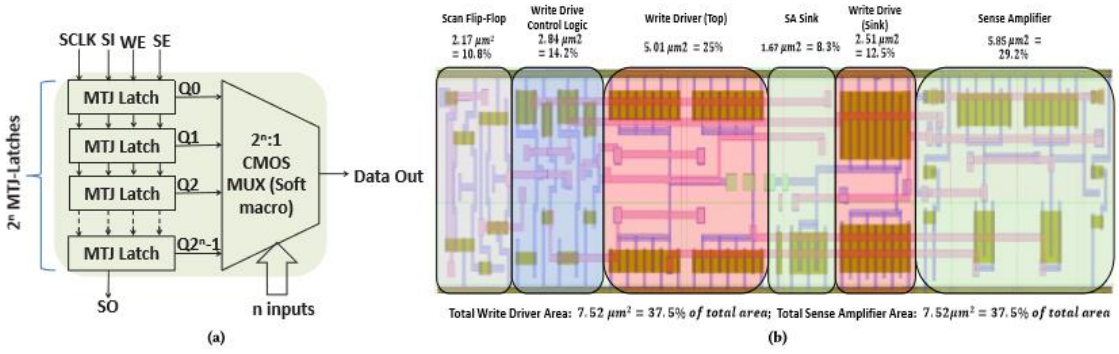


Figure 19 STT-LUT and Full Custom layout of MTJ Latch in Standard Cell Format

Each configuration bit is stored by a magnetic tunnel junction (MTJ) latch with scan chain programmability as shown in Figure 20. 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 scheme for parallel write to 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 VDD 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 covert 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 is 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.

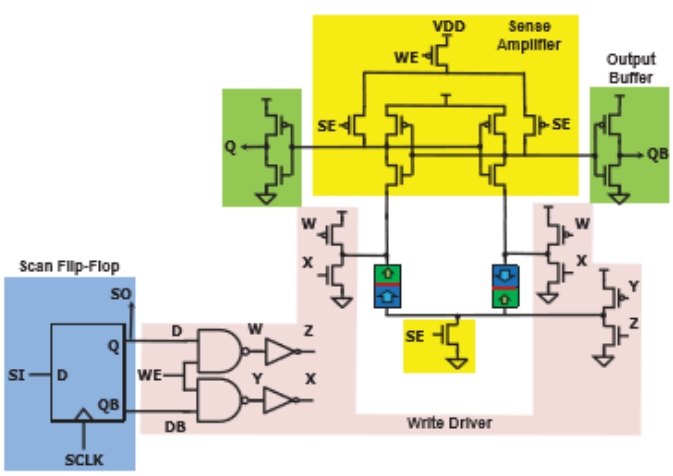


Figure 20 MTJ-Latch with Scan Chain Programming

The MTJ-latch is designed in a full-custom manner and needs to be optimized for sensing reliability and area and delivered as a standard cell for integration to the ASIC design flow. The full custom design and optimization of the one-bit MTJ latch cell is performed in the Synopsys generic 28nm process. The write drivers tend to require large transistor to produce sufficient current needed for MTJ write. The write transistors need to be optimized so that the write operation can succeed under process variations. We have performed a statistical transistor sizing optimization on the write driver for achieving near zero (less than 0.1%) write failure rate. 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 to be 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 20 shows the full-custom layout of the one-bit MTJ-latch designed in the format of a standard cell layout (fixed height). Most of the layout area

(37.5%) is occupied by the write drivers since the MTJ write current is still fairly large. Notice that the MTJ devices are stacked on top of this layout between two metallization layers (assuming M3 and M4) and hence do not occupy 2D area. M3 pins are placed for connection to the MTJ layers.

STT-Based LUT versus CMOS-based LUT

Figure 21 shows the comparison of the area of the MTJ latch, STT-LUTs, and areas of other standard cells in 28nm. The MTJ latch area is 6× to 15× that of basic logic gates, and 3× larger than SRAM based D flip-flop (FF). The MTJ-latch, however, shows much less leakage power. It has 7× to 11× less leakage power compared to basic CMOS logic gates, and 20× smaller compared to SRAM based D-FF.

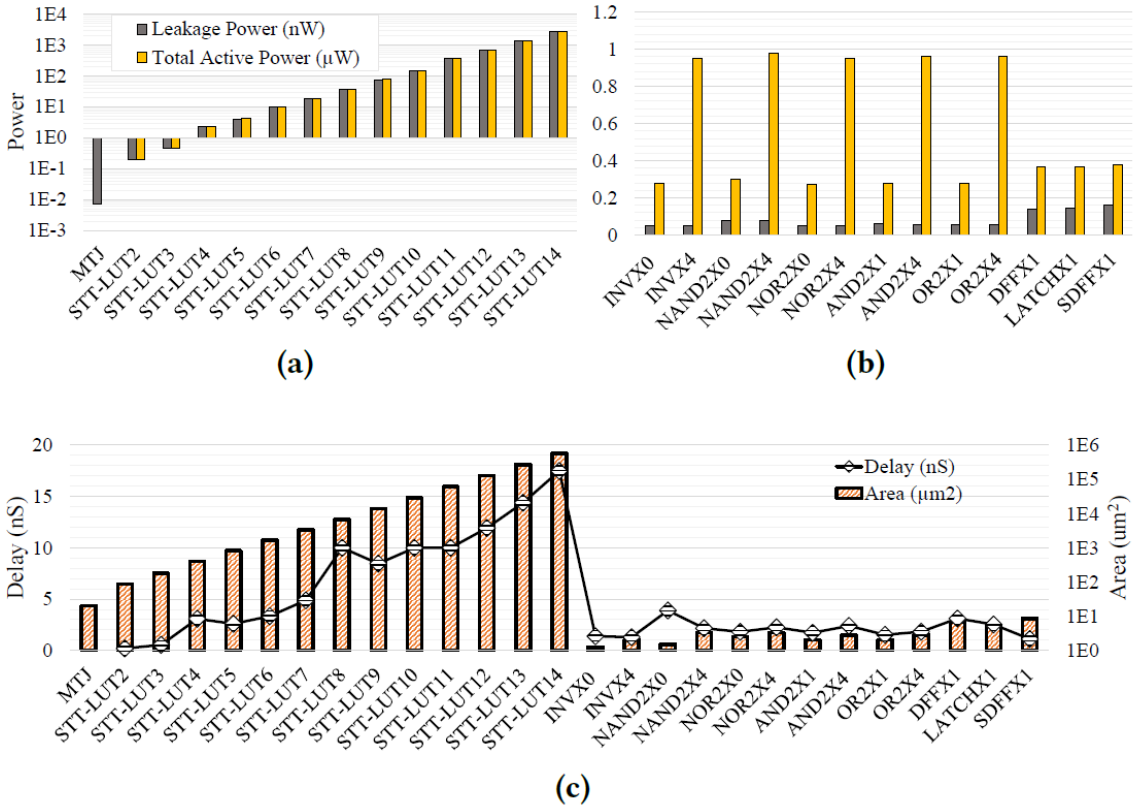


Figure 21 Comparison of (a, b) Power, (c) Delay, and Area of STT-LUT and Standard Cells in 28nm

The delay and active mode power of the STT-LUT is determined by the multiplexer part of the LUT which is optimizable by the logic synthesizer. Figure 21 presents the comparison of 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 large MTJ latch area, the areas of LUTs are noticeably higher 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.

LUT Size versus Number of LUTs

One of the straight forward approaches for LUT-based obfuscation is to increase the number of LUTs to enhance the security of the approach against SAT attack. In this scheme, each candidate gate will be replaced with same-input LUT. For instance, 2-input gates will be replaced with LUT2s, and 3-input gates with LUT3. However, this not only imposes non-negligible PPA overhead, it also cannot provide the highest resiliency against SAT attacks [35]. Despite increasing the number of same-input LUTs in design, which is not perfectly resilient against SAT attack, it is possible for a designer to use enlarged LUTs. As an instance, instead of using a LUT n for n-input gate, a LUT_n+ (i.e. LUT_{n+1}, LUT_{n+2}, ...) is used. Thus, by increasing the size of the LUTs, SAT attack replaces them with more deeper MUX trees, and consequently, the de-obfuscation time gets worse to exploit the value of keys for LUTs, making them resilient.

In addition, SAT attacks works based on Conflict-Driven Clause Learning (CDCL) for finding DIPs. Consequently, due to symmetric structure of MUX tree model, there is no short leaf in the equivalent logic to find the cut-off (conflicts) in the logic tree. Accordingly, enlarging the size of LUTs (scale up) increases the depth of this symmetric tree which makes it harder against SAT solver to find conflicts.

Figure 22 shows an example of using enlarged LUTs in the design. As it can be seen, two 2-input NOR gate ($nor2_1$) and 3-input NAND gates ($nand3_1$) are replaced by LUT4s. Using larger LUTs provide some extra inputs for each LUT, called dummy inputs. For example, $LUT4_1$ and $LUT4_2$ have 2 and 1 extra inputs, respectively. Although different

assignment strategies can be applied and evaluated for feeding these extra inputs, to acquire the most secure solution, we arbitrarily fed these inputs from primary inputs [52].

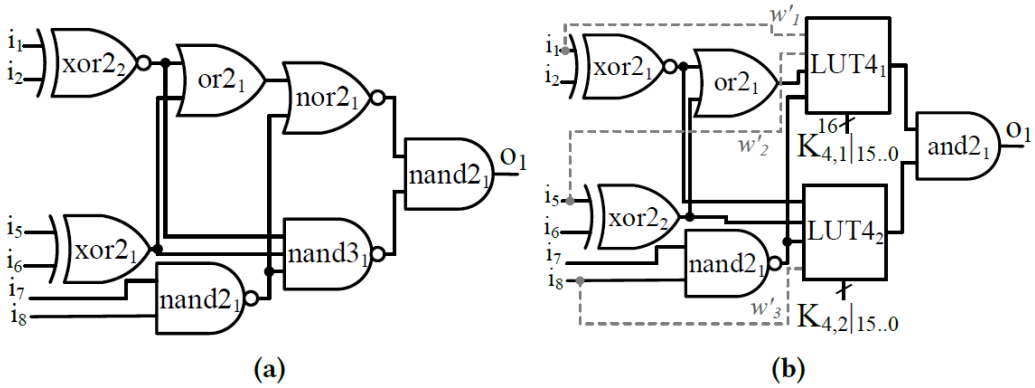


Figure 22 Using Enlarged LUTs for obfuscation, (a) A Sample Circuit, (b) Camouflaged with Enlarged LUTs with Primary Inputs as dummy Inputs

Replacement Strategies

Replacement strategy can potentially be considered as a promising optimization parameter for obfuscation, especially after the introduction of SAT attack. There are several conditions an effective replacement strategy need to meet to provide higher resiliency against SAT attacks. Two most important conditions are (1) low corruptibility, and (2) avoiding unintentionally correct key generation. By considering these conditions, we introduce a replacement strategy and compare it with random strategy [31]. To have better evaluation regarding the impact of each condition, we compare three different strategies in this work as follows.

Random Selection (RND)

As the baseline replacement strategy, we implement random selection/replacement, as opposed to the Independent Selection in [38]. As its name implies, the gates are selected randomly. However, since we use enlarged LUTs, if the LUT size is larger than the candidate gate, dummy inputs will be connected to primary inputs.

Low Output Corruptibility (LC)

As mentioned previously, SAT attacks work based on Conflict-Driven Clause Learning (CDCL). Based on CDCL, SAT attack is looking for conflict clauses to make cut-off for learning clauses. Finding the conflict clauses depends on the comparison between two different outputs for two different keys. Consequently, higher hamming distance of two outputs, finding the DIP is easier. If obfuscation strategy influences different outputs by applying each input, the probability of hamming distance > 1 will be increased drastically, which lead to faster de-obfuscation. Therefore, the higher the corruptibility, the easier de-obfuscation by SAT attack is. So, it is evident that the optimal solution is to minimize the hamming distance to 1. In fact, maximum one output must be different when two different keys and DIP are applied. This is called low corruptibility if fewer (the best is 1) outputs are different after applying different keys and DIP. To have the lowest output corruptibility, we need to employ a custom Breadth First Search (BFS) on the logic design based on a corruption matrix which indicates the corruptibility of visited gates. While traversing from outputs toward the inputs, a dictionary with different corruption matrix will be created, which allows us to check the list of the gates with lowest

output corruptibility. After traversing, based on the number of gates targeted for obfuscation, the dictionary provides a list with the targeted number of gates, which have the lowest output corruptibility.

Avoiding Unintentionally Correct Key Generation (LC_NoGen)

Due to the capability of LUTs for implementing all 2^{2^N} possibilities for a LUT_n, connecting LUTs directly to each other may generate some extra correct keys. This scenario provides additional options for SAT solver to find the key, which results in decreasing the execution time of SAT solver drastically. Also, increasing the number of LUTs which are directly connected to each other significantly increases the number of correct possibilities in a design. As a simple example, connecting two inverters directly to each other has the same functionality with connecting two buffers directly to each other. Hence, since previous work use many gates to be replaced with LUTs, the probability of choosing directly connected gates are extremely high. However, since in this work our result indicates that the LUTs scale-up is the most effective factor on SAT execution time, we show that only a few numbers of LUTs are enough for SAT-resiliency in case of using large LUTs. Consequently, we show that this condition effectively increases execution time. By considering dictionary-based gate selection and avoiding unintentionally correct key generation, the proposed LUT-based replacement strategy has been illustrated in Figure 23. We illustrate that like increasing the number of LUTs, applying this algorithm provides considerable security enhancement impact. However, as we will show later in this work the results indicate that this solution is not as effective as increasing the size of LUTs.

The aim of the algorithm is to maintain low output corruptibility while avoiding correct key generation. The graph of the netlist is created and BFS is applied starting from Primary Output (PO) towards the Primary Input (PI). While traversing each Primary Cone, we tag the gate present in that cone with the PO's name. After traversing each cone, we have dictionary of gates with PO's name to which they contribute. A key is generated from this value and the gate is added next to that key in a dictionary. This gives us a dictionary which gives us a list of gates when queried with any combination of Primary Output. This dictionary is then processed to avoid correct key generation. For example, if Gate A, contribute to Output G1 and G2, then gate A will be tagged with output G1 and G2. So, if you query dictionary with G1, gate A will pop up. This dictionary will help us in finding the Output gate which has maximum gate coverage.

Now, the next aim is to select minimum number of Primary Output to keep less corruptibility while having maximum number of gate coverage. This problem is of minimizing one thing while maximizing the other. Once such primary outputs have been found, the algorithm replaces the gate with the LUTs. It may happen that we want to encrypt x gate, but if algorithm finds y gates where $y > x$, then gates can be sorted on various gate properties.

```
1:  for each (PO in outputs_list) do                    ▷ PO: Primary Outputs
2:      gate_list = BFS(PO);                            ▷ current PO Fanin Cone Gates
3:      for each (gate in gate_list) do
4:          gate.listPOs = find_affected_POs(gate)
5:  for each (gate in circuit) do
6:      for each (PO in gate.listPOs) do
7:          tag_key(PO)
8:          if isExist(tag_key(PO)) then
9:              dictionary.add(gate)
10:         else
11:             dictionary.add(gate)
12:             dictionary.addtag((tag_key(PO))
13: Sort(gates, logic_level, descending)
14: for each (gate in dictionary) do
15:     if isExist(Parent(gate)) then
16:         dictionary.delete(Parent(gate))
17: for each (gate in circuit) do
18:     if tag_key(gate) > target_no then list_key.add(tag_key)
19: Sort(list_key, length, ascending)
20: Replace_LUT(gates, target_no)
```

Figure 23 Algorithm for LC_NoGen to have less corruptibility and to avoid Correct key Generation

ASIC Iterative Security-driven Design Flow

The STT-LUT is defined as a Verilog module in which it has instances of the MTJ-Latch (or NV-latch) cell and the RTL code of the multiplexer. Notice that the NV-latch is an empty module as it is a new primitive non-synthesizable standard cell. The RTL of the multiplexer gets synthesized and implemented by logic gates in the process of logic synthesis. The proposed logic obfuscation needs a gate level netlist. The result of the obfuscation is a new net list in which identified gates for logic obfuscation are replaced with NV-LUTs. Since the NV-LUTs are firm macros which contain RTL code of a

multiplexer, the net list with NV-LUT inserted needs to be re-synthesized and optimized, given their impact on timing constraint. To obtain the best PPA results for design space exploration and finding the best solution for LUT-based obfuscation, we introduce an iterative-based design flow, which not only does not trade PPA with security, but also operates iteratively to the best available PPA solution.

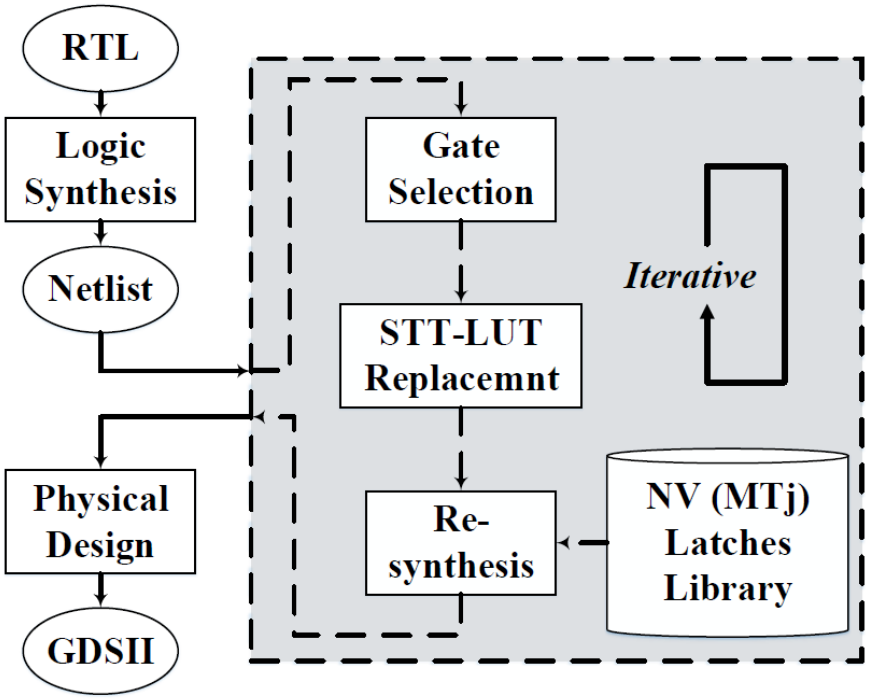


Figure 24 Iterative-based Security-driven Design Flow with Maximizing PPA

Figure 24 shows the proposed iterative security driven ASIC design flow optimization. The logic library of the NV-latch is used in the re-synthesized step and the multiplexer of the LUTs and the rest of the net list is further re-optimized to meet the original timing. If the design constraints cannot be met, the gate selection, replacement and

re-synthesis processes need to be iteratively performed until the PPA constraints as well as the security constraints are satisfied. After the final logic synthesis that meets the timing and area/power budget and the security requirements, the physical design steps follow to generate layout. While for our studied circuit benchmarks, two level of iteration was found to be sufficient to meet security and design parameter constraints, for larger circuits more iteration might be required.

Since the most effective factor is the size of LUTs (scale up), we demonstrate that replacing with only a few LUTs can bring resiliency against SAT solver. Therefore, based on the conditions we consider for replacement strategy, i.e., less corruptibility and avoiding unintentionally key generation, various choices are available per design which are the same in case of security against SAT attack. Consequently, this allows us to get the best choice for PPA optimization.

EXPERIMENTAL SETUP

To explore all design space including the impact of LUT size, the number of LUTs, the replacement strategy, we used a cluster computing environment which has 53 Dell computing nodes, each with dual Intel Xeon CPUs. The total number of cores ranging from 16 to 24 with RAM varying from 64GB to 512GB. We employed ISCAS-85 benchmarks for revaluation, illustrated in table 1. For security evaluation, we employed SAT attack described and developed by Subramanyan et al. [35], which utilizes Lingeling as its SAT solver.

We comprehensively measure and explore the SAT solver execution time by sweeping three out of four factors, i.e. (1) increasing the number of LUTs from 1 to 3 percent of the total circuit gates, (2) increasing the size of LUTs from 2 to 14 by using dummy inputs fed via primary inputs, and (3) replacement strategy, to demonstrate the impact of each factor for security design. Also, for SAT attack, a run time limit of 5 days (432×10^3 seconds) is set to demonstrate time out states. To account for run-to-run variations in performance, we ran the SAT solver 5 times for each obfuscated benchmark.

According to the security-driven PPA optimization, the benchmarks are first synthesized in Synopsys generic 28nm technology using Synopsys's Design Compiler. After the first synthesis and determining different solutions based on replacement strategy, an iterative-based re-synthesis will be started to find the optimal solution. In all iterations,

we use STT-LUT as the technology of LUT replacement, and accordingly, the circuit PPA overhead will be evaluated.

RESULTS AND DISCUSSION

Figure 25 illustrates the overall impact of the discussed three effective factors on execution time. As it can be seen, even for random insertion strategy, for LUT size larger than 10, obfuscating ~1% of each circuit is sufficient to provide SAT resiliency. Therefore, scaling up LUTs (increasing the size of LUTs) significantly increases the hardness of obfuscation regardless replacement strategy and number of LUTs. Also, LC_NoGen, which has both conditions considered in replacement strategy, remarkably increases SAT execution time, which shows its effectiveness on SAT resiliency.

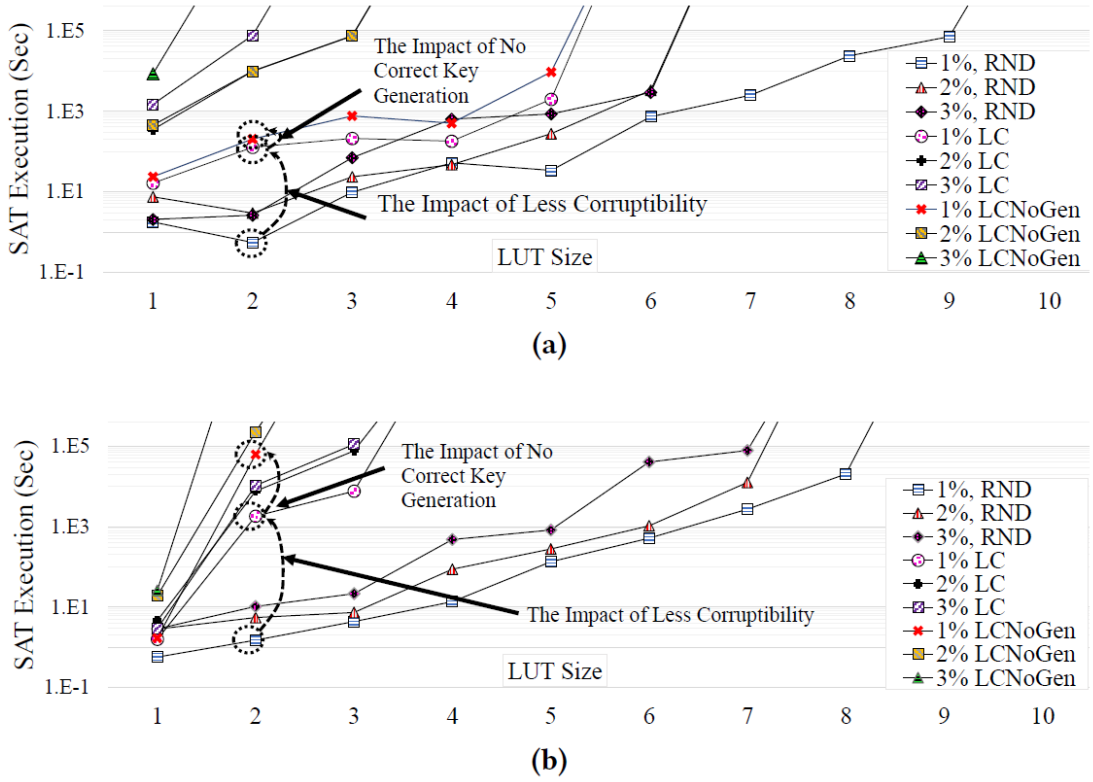


Figure 25 SAT Attack Execution Time for Different (1) Replacement Strategy, (2) LUT Size (scale up), and (3) Number of LUTs (scale out) in (a0 ISCAS-85 c2670, (b) ISCAS-85 c3540

Figure 26 demonstrates SAT execution time with more details on ISCAS-85 C7552 for smaller size of LUTs, and different number of LUTs. Since we increase the number of LUTs or LUT size in Figure 26, some variations, which are high in some cases can be observed. It should be noted that the variations (ups and downs in the execution time) are a result of the number of outputs selected for current obfuscation. Since our dictionary-based algorithm for choosing candidate gates minimize the number of affected outputs, after a target number there is no choice to select a candidate which affects a new output. Therefore, increasing corruptibility in outputs results in variation in SAT execution time.

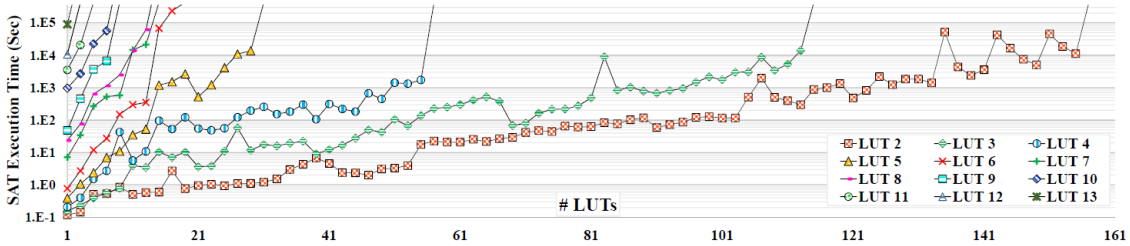


Figure 26 SAT Attack Execution Time on Synthesized ISCAS-85 C7552 with Different Number of LUTs and Different LUT Sizes

Table 2 SAT-attack Execution Time on Synthesized-Camouflaged ISCAS-85 Benchmarks for Different LUT Size and Different Number of LUTs.

Table 2: SAT-attack Execution Time on Synthesized-Camouflaged ISCAS-85 Benchmarks for Different LUT Size and Different Number of LUTs.

LUT Size	c432			c499			c880			c1355			c1908			c2670			c3540			c7552		
	1%	2%	3%	1%	2%	3%	1%	2%	3%	1%	2%	3%	1%	2%	3%	1%	2%	3%	1%	2%	3%	1%	2%	3%
2	0.035	0.039	0.065	0.079	0.099	0.099	0.059	0.185	0.202	0.117	0.751	1.453	0.302	1.689	1.155	23.59	449.1	∞	1.665	19.04	5.617	7.139	26.84	67.89
3	0.039	0.071	0.184	0.128	0.359	0.397	0.194	1.009	6.596	0.846	2.642	8.264	1.961	15.14	36.63	203.9	9777	∞	61813	∞	∞	∞	∞	∞
4	0.055	0.079	2.214	0.246	0.632	2.720	0.390	2.072	359.4	1.267	4.260	23.54	6.545	412.7	335.1	758.0	73165	∞	∞	∞	∞	∞	∞	∞
5	0.280	2.566	4.216	0.373	4.547	10.14	1.593	28.12	17685	4.208	77.74	93.61	196.9	13431	∞	499.5	∞	∞	∞	∞	∞	∞	∞	∞
6	0.554	14.70	65.62	1.264	21.47	66.76	6.177	50.52	∞	5.506	263.1	320.3	379.5	∞	∞	9347	∞	∞	∞	∞	∞	∞	∞	∞
7	2.112	72.11	274.5	12.64	67.31	1204	77.77	1876	∞	83.03	1711	45236	16846	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	11.89	3490	9534	56.26	553.0	4571	226.4	13786	∞	5501	71683	∞	81861	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	20.21	7074	18129	905.6	6333	10653	2641	88870	∞	13450	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	415.3	26468	48474	2354	25678	65812	23717	∞	∞	37122	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
11	1109	45632	∞	54381	79675	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
12	7868	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
13	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
14	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

∞: *Timeout*

Table 2 demonstrates the execution time for different benchmarks when 1%, 2%, and 3% of the gates are obfuscated. In addition, the size of LUTs is in the range between 2 to 14. Note that, like Figure 26 SAT Attack Execution Time on Synthesized ISCAS-85 C7552 with Different Number of LUTs and Different LUT Sizes, LC_NoGen is deployed as it is the most effective replacement strategy. As it can be seen, the SAT execution time increases nearly exponentially in both dimension, i.e. scale up (increasing the size of LUTs) and scale out (increasing the number of LUTs). However, for a fixed number of LUTs, scale up (increasing the size) is more effective rather than scale out (increasing the number

of LUTs). For instance, for LUT14, only replacing a single gate with a LUT14 is sufficient to make the design perfectly resilient against SAT attack.

To understand and compare the impact of LUT scale up vs. scale out on SAT execution time, we use a regression model to demonstrate the relationship between SAT execution time with respect to these two parameters. Figure 27 provides two different scenarios to accurately model the relationship between SAT and size of LUTs, as well as the number of LUTs. As it can be seen in Figure 27 (a), one factor is fixed in each curve. In one of them, LUT size is set to 5, and the number of LUTs is swept from 1 to 29. In another curve, the number of LUTs is set to 13, and 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 to observe that LUT scale-up have significantly more influence on SAT execution time compared to LUT scale out. Figure 27 (b) shows another analogous situation that proves that LUT scale up is more effective than LUT scale out. In addition, according to a multi-dimensional linear regression, the impact coefficient of the number of LUTs on SAT execution time is 72.347. However, impact coefficient of LUT size is 1969.25. In addition, these factors have an intercept coefficient by −8325.47. This regression coefficient demonstrates that LUT size is the most crucial factor for security purposes rather than the number of LUTs and replacement strategies.

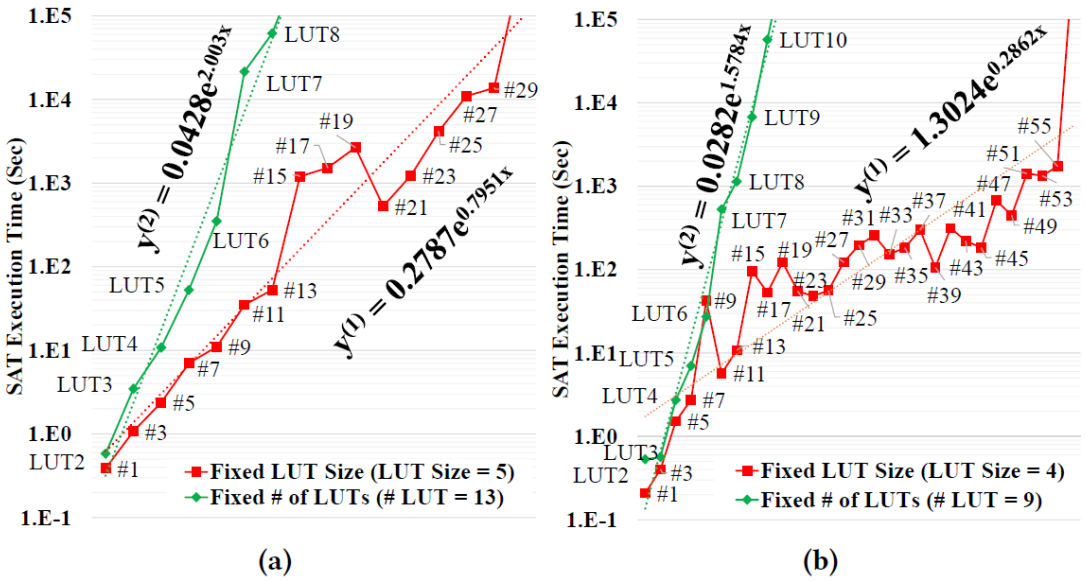


Figure 27 LUT scale up vs. scale out: Comparison between the Impact of LUT size and Number of LUTs on SAT Execution Time

Although our evaluations show that LUT scale-up is the most straight forward approach for LUT-based obfuscation with high resiliency against state-of-the-art attacks, investigation on PPA optimizations how that precipitous increases in the number of LUTs and the size of LUT can be inefficient solutions. In other words, while we identified design points that are resilient against current attacks, yet incur low PPA overheads, to guarantee resiliency against future complex attacks, just increasing the number of LUTs (scale up) and the size of LUT (scale out) are not effective solutions in terms of PPA overhead. Figure 28 (a) depicts normalized delay overhead for the different number of LUTs and LUT sizes. As it can be seen, in some cases the delay overhead will be increased up to 2×. For instance, for LUT size = 14, although only one gate replacement is enough to make it SAT resilient, 2× delay overhead is not negligible especially for designers who have tight timing constraint. Note that number of LUTs are swept in Figure 28 (a), and after each increase in

LUT size, one or more columns corresponding to a specific LUT size are eliminated. Eliminating any column means that for that situation SAT attack could not find the correct key and it returned timeout. In fact, all results are related to all configurations before SAT timeout.

Although delay may impose up to 2x overhead in the best SAT resilient solutions, the area and power overheads for most SAT resilient solutions are much higher. As it can be seen in Figure 28 (b), and Figure 28 (c), the frontiers Pareto optimal curves are SAT resilient solutions with minimum area/power overhead for the shown configuration. Based on these figures, it is clear to observe that for large LUT sizes (more than10 inputs) we can impose even more than 100× area/power overhead. Note that all results are based on 28nm integrative STT-based LUTs, which is a competitive technology for area/power/performance constraints. This implies that while for today attacks hybrid STT-LUT is an effective solution with respect to security and PPA design constraints, for future complex attacks, we cannot rely on scaling up (larger LUT) or scaling out (more LUT) LUTs for resiliency.

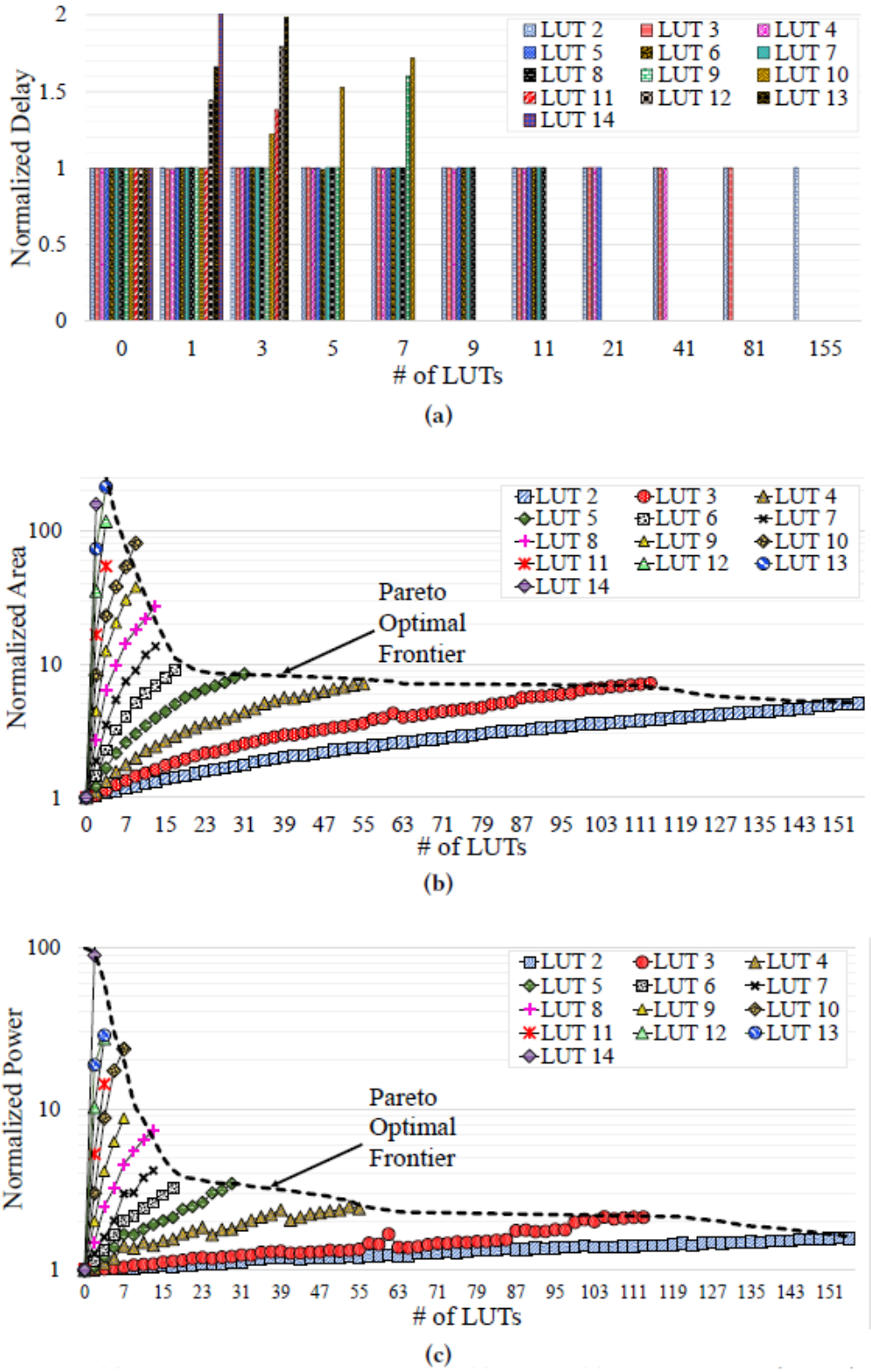


Figure 28 The Impact of Increasing # of LUTs and LUT size on Circuit (C7552) Design Properties: (a)Delay, (b)Area), (c)Power.

This leads towards designing and implementing new customized and area/power optimized LUTs to keep LUT-based obfuscation a promising solution. For instance, an STT-LUT where internally has only 4 MTJs accommodating 4 configuration states while externally is augmented with 10 additional inputs can deliver resiliency of a 12 input LUT while at the same time meeting PPA constraints of a 2 input LUT. A challenge in this case is that the type of customized LUT design can be detectable via delayering and the attacker can identify the discrepancy between the number of inputs to the LUT and the number of MTJ memory cells inside the LUT. To prevent this, the designer can employ camouflaging for implementing customized LUTs to not only meet all PPA constraints, but also to guarantee the security against reverse engineering. This will be an important topic of research for future work.

CONCLUSION

In this work, we comprehensively investigate the four crucial factors of (1) the technology of LUT, (2) LUT size, (3) number of LUTs, and (4) replacement strategy which are influencing PPA optimization and security enhancement of LUT-based obfuscation. Unlike prior work which mostly focus on replacement policy, we show that while replacement policy can be effective, it is not the most important parameter for security guarantees. Our experimental results show that the size of LUT is the most influential and straight forward factor in SAT resiliency, even for a weak random replacement strategy, however, it introduces PPA design overhead. While the delay overhead can be substantially eliminated using our proposed iterative solution which is none disruptive to standard ASIC design flow, the power and area overhead need attention. The PPA results indicate that for today state-of-the-art attacks STT-LUT is an effective solution with respect to security and PPA design constraints, however for future evolving attacks, we cannot rely on scaling up (larger LUT) or scaling out (more LUT) solutions for resiliency, due to the large impact on power and area. To mitigate this impact, customized LUT design or camouflaged customized LUTs must be deployed to realize LUT-based obfuscation with permissive PPA overheads and guaranteed security.

REFERENCES

1. Defense Science Board Task Force On High-Perf Microchip Supply. (n.d.). doi:2005
2. Karri, R., Rajendran, J., Rosenfeld, K., & Tehranipoor, M. (2010). Trustworthy Hardware: Identifying and Classifying Hardware Trojans. *Computer,43*(10), 39-46. doi:10.1109/mc.2010.299
3. Xiao, K., Forte, D., Jin, Y., Karri, R., Bhunia, S., & Tehranipoor, M. (2016). Hardware Trojans. *ACM Transactions on Design Automation of Electronic Systems,22*(1), 1-23. doi:10.1145/2906147
4. Rostami, M., Koushanfar, F., & Karri, R. (2014). A Primer on Hardware Security: Models, Methods, and Metrics. *Proceedings of the IEEE,102*(8), 1283-1295. doi:10.1109/jproc.2014.2335155
5. Amir, S., Shakya, B., Xu, X., Jin, Y., Bhunia, S., Tehranipoor, M., & Forte, D. (2018). Development and Evaluation of Hardware Obfuscation Benchmarks. *Journal of Hardware and Systems Security,2*(2), 142-161. doi:10.1007/s41635-018-0036-3
6. Bhatkar, S., DU Varney, D. C., & Sekar, R. (n.d.). Address Obfuscation: An Efficient Approach to Combat a Broad Range of Memory Error Exploits.
7. Fujiwara, H., & Brgles, F. (n.d.). Https://www.researchgate.net/publication/239562831_A_neutral_netlist_of_10_combinational_circuits_and_a_target_translator_in_fortran.
8. Berkeley Logic Synthesis and Verification Group (2004) ABC: A system for sequential synthesis and verification. (n.d.). http://www.eecs.berkeley.edu/alanmi/abc/
9. Brglez F, Bryan D, Kozminski K (1989). Combinational profiles of sequential benchmark circuits. Proceedings of IEEE International symposium on circuits and systems, 1989, vol 3, pp 1929–1934. https://doi.org/10.1109/ISCAS.1989.100747
10. Business Wire (2017) Inside secure unveils industry's first root-of-trust solution based on RISC-V processor. https://www.businesswire.com/news/home/20171114006581/en/Secure-Unveils-Industry
11. Yang, B., Wu, K., & Karri, R. (2005). Secure scan: A design-for-test architecture for crypto chips. *Proceedings. 42nd Design Automation Conference, 2005.*doi:10.1109/dac.2005.193787
12. Rahman, M. T., Forte, D., Shi, Q., Contreras, G. K., & Tehranipoor, M. (2014). CSST: An Efficient Secure Split-Test for Preventing IC Piracy. *2014 IEEE 23rd North Atlantic Test Workshop*. doi:10.1109/natw.2014.17
13. Roy, J. A., Koushanfar, F., & Markov, I. L. (2008). EPIC: Ending Piracy of Integrated Circuits. *2008 Design, Automation and Test in Europe*. doi:10.1109/date.2008.4484823
14. Juretus, K., & Savidis, I. (2016). Reduced Overhead Gate Level Logic Encryption. *Proceedings of the 26th Edition on Great Lakes Symposium on VLSI - GLSVLSI 16*. doi:10.1145/2902961.2902972
15. Liu, B., & Wang, B. (2015). Reconfiguration-Based VLSI Design for Security. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems,5*(1), 98-108. doi:10.1109/jetcas.2014.2372431

16. Chakraborty, R. S., & Bhunia, S. (2010). RTL Hardware IP Protection Using Key-Based Control and Data Flow Obfuscation. *2010 23rd International Conference on VLSI Design*. doi:10.1109/vlsi.design.2010.54
17. Frey, J., & Yu, Q. (2017). A hardened network-on-chip design using runtime hardware Trojan mitigation methods. *Integration, the VLSI Journal,56*, 15-31. doi:10.1016/j.vlsi.2016.06.008
18. Jagasivamani, M., Gadfort, P., Sika, M., Bajura, M., & Fritze, M. (2014). Split-fabrication obfuscation: Metrics and techniques. *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. doi:10.1109/hst.2014.6855560
19. Otero, C. T., Tse, J., Karmazin, R., Hill, B., & Manohar, R. (2015). Automatic obfuscated cell layout for trusted split-foundry design. *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. doi:10.1109/hst.2015.7140237
20. Vaidyanathan, K., Das, B. P., Sumbul, E., Liu, R., & Pileggi, L. (2014). Building trusted ICs using split fabrication. *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. doi:10.1109/hst.2014.6855559
21. Xiao, K., Forte, D., & Tehranipoor, M. M. (2015). Efficient and secure split manufacturing via obfuscated built-in self-authentication. *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. doi:10.1109/hst.2015.7140229
22. Yang, P., & Marek-Sadowska, M. (2016). Making split-fabrication more secure. *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD 16*. doi:10.1145/2966986.2967053
23. Magaña, J., Shi, D., & Davoodi, A. (2016). Are proximity attacks a threat to the security of split manufacturing of integrated circuits? *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD 16*. doi:10.1145/2966986.2967006
24. Rajendran, J. (., Sinanoglu, O., & Karri, R. (2013). Is Split Manufacturing Secure? *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*. doi:10.7873/date.2013.261
25. Vaidyanathan, K., Das, B. P., & Pileggi, L. (2014). Detecting Reliability Attacks during Split Fabrication using Test-only BEOL Stack. *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference - DAC 14*. doi:10.1145/2593069.2593123
26. Wang, Y., Chen, P., Hu, J., & Rajendran, J. (. (2016). The cat and mouse in split manufacturing. *Proceedings of the 53rd Annual Design Automation Conference on - DAC 16*. doi:10.1145/2897937.2898104
27. Alkabani, Y., & Koushanfar, F. (2007). Active Hardware Metering for Intellectual Property Protection and Security. Proceedings of 16th *USENIX Security Symposium*,1-20.
28. Kahng, A. B., Lach, J., Mangione-Smith, W. H., Mantik, S., Markov, I. L., Potkonjak, M., . . . Wolfe, G. (1998). Watermarking Techniques for Intellectual Property Protection. *Design and Automation Conference (DAC)*,776-781.
29. Imeson, F., Emtenan, A., Garg, S., & Tripunitar, M. V. (2013). Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation. *USENIX Conference on Security*,495-510.

30. Ronald P. Cocchi and Lap Wai Chow and James P. Baukus and Bryan J. Wang, "Method and apparatus for camouflaging a standard cell based integrated circuit with micro circuits and post processing," in US Patent, 2013
31. Roy, J. A., Koushanfar, F., & Markov, I. L. (2008). EPIC: Ending Piracy of Integrated Circuits. *2008 Design, Automation and Test in Europe*. doi:10.1109/date.2008.4484823
32. Yasin, M., Sengupta, A., Nabeel, M. T., Ashraf, M., Rajendran, J. (., & Sinanoglu, O. (2017). Provably-Secure Logic Locking. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS 17*. doi:10.1145/3133956.3133985
33. Skudlarek, J. P., Katsioulas, T., & Chen, M. (2016). A Platform Solution for Secure Supply-Chain and Chip Life-Cycle Management. *Computer,49*(8), 28-34. doi:10.1109/mc.2016.243
34. Massad, M. E., Garg, S., & Tripunitara, M. (2015). Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes. *Proceedings 2015 Network and Distributed System Security Symposium*. doi:10.14722/ndss.2015.23218
35. Subramanyan, P., Ray, S., & Malik, S. (2015). Evaluating the security of logic encryption algorithms. *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. doi:10.1109/hst.2015.7140252
36. Baumgarten, A., Tyagi, A., & Zambreno, J. (2010). Preventing IC Piracy Using Reconfigurable Logic Barriers. *IEEE Design & Test of Computers,27*(1), 66-75. doi:10.1109/mdt.2010.24
37. Liu, B., & Wang, B. (2014). Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks. *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*. doi:10.7873/date2014.256
38. T. Winograd, H. Salmani, H. Mahmoodi, K. Gaj , & H. Homayoun (2016). "Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks," in ACM/EDAC/IEEE Design Automation Conference (DAC), 1-6
39. H. M. Kamali, K. Z. Azar, K. Gaj, H. Homayoun, & A. Sasan(2018). "LUT-Lock: A Novel LUT-based Logic Obfuscation for FPGA-Bitstream and ASIC-Hardware Protection," in IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 1-6
40. Rajendran, J., Pino, Y., Sinanoglu, O., & Karri, R. (2012). Security analysis of logic obfuscation. *Proceedings of the 49th Annual Design Automation Conference on - DAC 12*. doi:10.1145/2228360.2228377
41. J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu & R. Karri (2015). "Fault Analysis-Based Logic Encryption," in IEEE Transactions on Computers, vol. 64, no. 2, 410-424.
42. SypherMedia, “Syphermedia library circuit camouflage technology,” http://www.smi.tv/solutions.htm.
43. J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, & B. J. Wang, (2012). “Building block for a secure cmos logic cell library,” US Patent no. 8111089.
44. J. P. Baukus, L. W. Chow, and W. Clark (2002). “Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide,” US Patent no. 20020096776.

45. J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, & B. J. Wang, (2012). “Building block for a secure cmos logic cell library,” US Patent no. 8111089.
46. Li, M., Shamsi, K., Meade, T., Zhao, Z., Yu, B., Jin, Y., & Pan, D. Z. (2016). Provably secure camouflaging strategy for IC protection. *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD 16*. doi:10.1145/2966986.2967065
47. Eén, N., & Sörensson, N. (2004). An Extensible SAT-solver. *Theory and Applications of Satisfiability Testing Lecture Notes in Computer Science*,502-518. doi:10.1007/978-3-540-24605-3_37
48. S. Roshanisefat, H. Mardani Kamali, & A. Sasan (2018). “SRCLock: A SAT Resistant Cyclic Logic Locking for Protecting the Hardware,” in Proceedings of the on Great Lakes Symp. on VLSI 2018.
49. Roshanisefat, S., Thirumala, H., Gaj, K., Homayoun, H., & Sasan, A. (2018). Benchmarking the Capabilities and Limitations of SAT Solvers in Defeating Obfuscation Schemes. *IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*,.
50. Shamsi, K., Li, M., Meade, T., Zhao, Z., Pan, D. Z., & Jin, Y. (2017). Cyclic Obfuscation for Creating SAT-Unresolvable Circuits. *Proceedings of the on Great Lakes Symposium on VLSI 2017 - GLSVLSI 17*. doi:10.1145/3060403.3060458
51. A. Attaran, T. Sheaves, P. Mugula, & H. Mahmoodi (2018). "CycSAT: SAT-based attack on cyclic logic encryptions," in Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI), 1-6.
52. Xie, Y., & Srivastava, A. (2018). Anti-SAT: Mitigating SAT Attack on Logic Locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*,1-1. doi:10.1109/tcad.2018.2801220
53. Yasin, M., Mazumdar, B., Sinanoglu, O., & Rajendran, J. (2017). Security analysis of Anti-SAT. 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC). doi:10.1109/aspdac.2017.7858346
54. Yasin, M., Mazumdar, B., Rajendran, J. J., & Sinanoglu, O. (2016). SARLock: SAT attack resistant logic locking. 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). doi:10.1109/hst.2016.7495588
55. Rajendran, J., Sam, M., Sinanoglu, O., & Karri, R. (2013). Security analysis of integrated circuit camouflaging. Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security - CCS 13. doi:10.1145/2508859.2516656
56. Yasin, M., Mazumdar, B., Rajendran, J. J., & Sinanoglu, O. (2017). TTLock: Tenacious and traceless logic locking. 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). doi:10.1109/hst.2017.7951830
57. Shamsi, K., Li, M., Meade, T., Zhao, Z., Pan, D. Z., & Jin, Y. (2017). AppSAT: Approximately deobfuscating integrated circuits. 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). doi:10.1109/hst.2017.7951805
58. Shen, Y., & Zhou, H. (2017). Double DIP. Proceedings of the on Great Lakes Symposium on VLSI 2017 - GLSVLSI 17. doi:10.1145/3060403.3060469
59. Yasin, M., Mazumdar, B., Sinanoglu, O., & Rajendran, J. (2016). CamoPerturb. Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD 16. doi:10.1145/2966986.2967012

BIOGRAPHY

Gaurav Kolhe is a master's student at George Mason University in Computer Engineering department. He graduated from Rajiv Gandhi College of Engineering and Research, Nagpur, in 2015. His research interest is in hardware-level functional safety and security. Gaurav is a recipient of Richard Newton Young Fellowship at Design Automation Conference, 2018.